ASPECTOS CONCEITUAIS SÔBRE CONCORRÊNCIA EM BANCO DE DADOS

Paulo Renato Bastos Pinto

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:

Prof. Gernot Richter
(Presidente)

Prof. Nelson Maculan Filho

Prof. Rubens Nascimento Melo

RIO DE JANEIRO, RJ - BRASIL

JULHO DE 1979

PINTO, PAULO RENATO BASTOS

Aspectos Conceituais Sobre Concorrência em Banco de Dados (Rio de Janeiro)

X, 160p. 29,7cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1979)

Tese - Univ.Fed. Rio de Janeiro. Fac. Engenharia

1. Banco de Dados I.COPPE/UFRJ II. Aspectos Conceituais Sobre Concorrência em Banco de Dados.

À minha mãe.

AGRADECIMENTOS

Ao Prof. Gernot Richter, meu orientador, pela dedicação com que me acompanhou durante a elaboração desta tese;

Aos Prof. Nelson Maculan Filho e Rubens Melo por terem participado da banca de tese;

À Suely pelo trabalho datilográfico.

RESUMO

Este trabalho apresenta um tratamento de concorrência em Sistemas de Banco de Dados de forma conceitual. Os conceitos desenvolvidos foram baseados nos conceitos já desenvolvidos de IMC (Information Management Concepts), e a interface escolhida para a realização deste trabalho, foi a interface LOBAN (Linguagem de Operação de BANco de Dados) , desenvolvida no projeto MINIBAN.

São apresentados também dois métodos de tratamento de concorrência tendo em vista esta interface de referência. Podendo no entanto, os conceitos desenvolvidos neste trabalho, serem aplicados a qualquer interface para sistema de Banco de Dados.

Zusammenfassung

In dieser Arbeit wird das Konzept der Konkurrenz in Datenbank-systemen auf bregriffsmaessiger Basis behandelt.

Die Begriffe, die wntwickelt wurdern, beruhen auf schon vorhandenen Festlegungen des IMC (Information Management Concepts), und die fuer die Verwirklichung dieser Arbeit ausgesuchten Begriffe sind jene, die fuer das Projekt MINIBAN entwickelt wurden und die unter dem Namen LOBAN (Linguagem de Operação de BANco) bekannt sind.

Ausserdem werden zwei Behandlungsmethoden der Konkurrenzprobleme presentiert und zwar mit Bezug auf die erwaehnten Wechselbeziehunger, die als Referenz dienen. Es soll aber ausdrücklich festgestellt bleinben, dass die Begriffe dieser These auf jegliche Wechselbeziehungern in Datenbanksystem angewendet werden koennen.

A B S T R A C T

This work presents a concurrent treatment of Data Base Management Systems in a conceptual way. The developed concepts were based in the IMC(Information Mangement Concepts), and the chosen interface was LOBAN, developed within MINIBAN project.

Two methods of concurrent control are presented by seeking this reference interface, although the developed concepts may be applied to any interface for Database Management Systems.

ÍNDICE

Páginas

CAPÍTULO I

ASPECTOS SOBRE CONCORRÊNCIA EM SISTEMAS DE GERÊNCIA DE BASE DE DADOS

Daremos a seguir uma idéia do fluxo de informação, bem como do processo de comunicação entre os vários usuários e um SBD, por meio de um funciograma. Antes introduziremos alguns conceitos necessários.

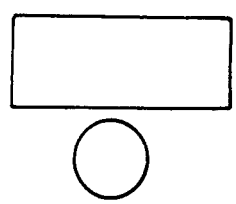
1.1. Estações e Canais

Estações e Canais são unidades funcionais componentes de um sistema. A estação desempenha o papel da unidade funcional que pode agir, isto é, processar informações. O canal desempenha no sistema o papel da unidade funcional que pode guardar informações, isto é, pode assumir estados que representam informações.

Quanto a representação de estações e canais, para a especificação de um funciograma , que é a representação por meio de estações e canais de um dado sistema, utiliza-se

ESTAÇÕES →

CANAIS →



Na figura 1.1, encontra-se o funciograma de um SBD, onde as estações com nomes $U_1, U_2, \ldots, U_n$ são unidades funcionais que submetem ao SBD, instruções, que denominaremos doravante por "usuários". Os canais denominados por $C_1, C_2, \ldots, C_n$,

são os canais utilizados para a comunicação entre os vários usuários e o SBD.

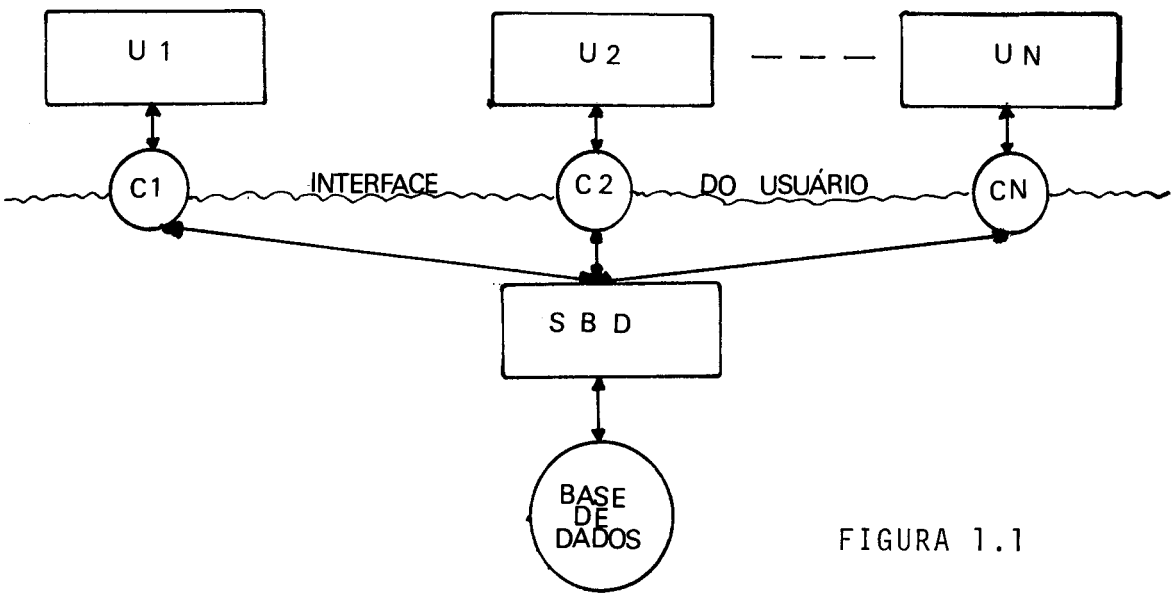


FIGURA 1.1

A concorrência no SBD é introduzida para permitir que os usuários possam operar na Base de Dados o mais independentemente possível. Sem a permissão de concorrência os usuários teriam que ser sequencializados, isto é, primeiro operaria U1, depois U2, etc... . Com a possibilidade de concorrência, existe um melhor aproveitamento dos recursos do sistema, como também uma menor espera por parte dos usuários. No entanto, para que o sistema possa permitir concorrência é necessário que haja um controle sobre esta concorrência devido principalmente a dois aspectos.

- A manutenção da integridade das informações na base de dados
- A não interferência entre usuários

Explicaremos a seguir estes dois aspectos de uma forma simplificada, e no decorrer deste trabalho as explicações mais detalhadas serão abordadas

1.2. Integridade

Para nosso objetivo, adimitimos que em um SBD, é permitido aos usuários definirem ligações ou asserções em informações, e desde que as asserções são mantidas, dizemos que o sistema se encontra em estado integro. Por exemplo, tomemos um arquivo que possue vários registros com informações a cerca de funcionários em uma dada empresa, e que nesta empresa, nenhum funcionário percebe mais do que seu gerente respectivo. Por meio então das facilidades permitidas na interface do usuário, poderíamos ter a seguinte asserção.

> "Nenhum funcionário ganha mais do que seu gerente"

Desta forma definiríamos uma ligação entre as informações dos salários dos funcionários e o salário do gerente desses funcionários. Em caso de uma alteração ser efetuada no salário de um funcionário de forma a torná-lo maior do que o salário de seu gerente, diríamos que o sistema estaria em estado não integro.

Com a finalidade da manutenção da integridade, não poderia o sistema permitir que dois usuários, pudessem ao mesmo tempo estarem; um alterando salários de funcionários, e outro alterando salário de gerentes. Porque além do risco da perda de um estado integro, teríamos também que nem os usuários têm possibilidade de recuperar a integridade e nem o sistema, como veremos posteriormente.

1.3. Interferência

Havendo concorrência, evidentemente, existe a possibilidade de dois usuários requererem a mesma informação, por exemplo, ambos querem alterar o salário de um dado funcionário. Assim, para que um usuário não desfaça a alteração realizada por outro usuário, até que este outro usuário termine de submeter suas instruções ao SBD, o SBD, deve garantir que não haja uma interferência entre os vários usuários.

1.4. Concorrência em sistemas operacionais(SO) e SBD

Para caracterizar o aspecto de concorrência em SBD, e a dificuldade para o seu tratamento, podemos tomar como referência o tratamento desse mesmo aspecto em S.O. Enquanto que os "recursos" (unidades de fita, UCP, etc...) em SO, são tratados somente sob o ponto de vista "físico", os "recursos" (informações manipuladas pelo SBD) para os SBD, são encarados sob os pontos de vista lógico, (devido as referências que os usuários realizam), como também sob o ponto de vista físico(devido ao mapeamento da estrutura lógica para a estrutura física). Assim, os problemas como deadlock, distribuição eficiente, dos "recursos" etc..., ocorrem em SBD, tanto sob o ponto de vista físico como sob o ponto de vista lógico.

Este trabalho realiza um levantamento dos aspectos relevantes para o tratamento de concorrência em SBD. Para realização deste tratamento utilizamos a interface LOBAN |[22]|, desenvolvida no projeto MINIBAN.

CAPÍTULO II

CONSTRUÇÕES DE INFORMAÇÕES E OPERAÇÕES PARA SEU PROCESSAMENTO

2.1. Composição de Construções

Aqui deve-se ressaltar bem a distinção entre a informação e sua representação. A informação está no nível de abstração, enquanto a representação está no nível meio físico. Ao usuário não é necessário o conhecimento da forma de representação das informações na base de dados, isto é, como elas são armazenadas internamente pelo Banco de Dados. Ao usuário interessam apenas as informações e o enfoque, vista ou organização que ele tem da informação regidas pelas regras da interface. É sob este último ponto de visto, isto é, da organização da informação que a base de dados será descrita a seguir. A cada estrutura referenciável pelo usuário implícita ou explícitamente, chamamos de construção.

As construções se compõem de outras construções e assim sucessivamente, até chegar ao nível mais elementar, ou seja ao nível denominado atômico.

Esta forma de composição será descrita a seguir, em suas linhas gerais, deixando-se de mencionar construções secundárias para não obscurecer a visão global que se pretende transmitir.

O conteúdo total da base de dados é chamado de acervo que é composto de vários tipos de construções, sendo os principais deles chamados arquivos. Em um acervo podem

coexistir dois tipos de arquivos, que são arquivos relacionais e os ligacionais. Aos arquivos são atribuidos nomes e o usuário pode fazer referência nominal a qualquer deles dentro de um acervo. Na figura 2.1 está representado um acervo com alguns arquivos, cujos nomes são ARQ-1, ARQ-2,...,ARQ-M:

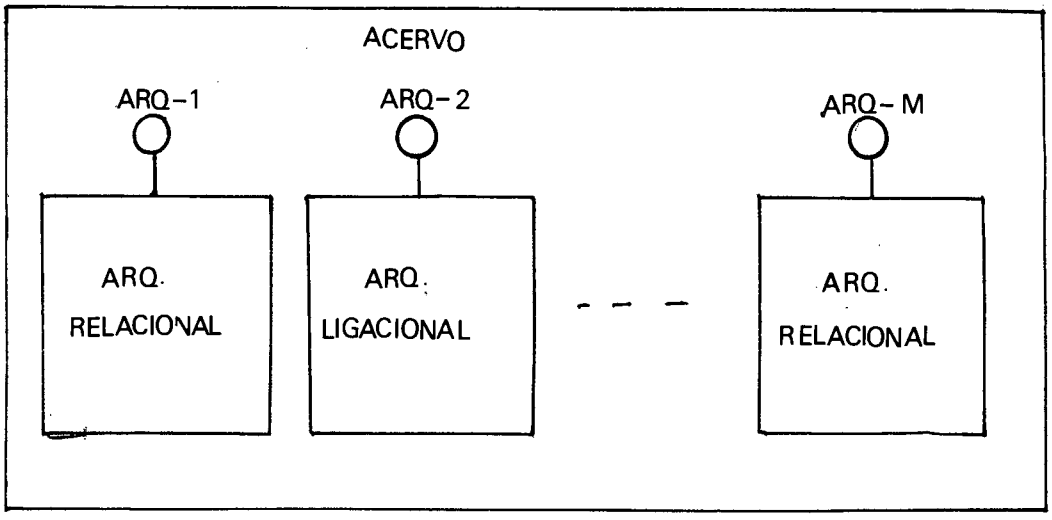


FIGURA 2.1

Um arquivo relacional contém uma tabela, entre outros componentes, que é uma tabela relacional, em que cada linha é uma tupla, no sentido da terminologia relacional, ou um registro, na terminologia corrente de processamento de dados. Cada componente de uma tupla, na terminologia relacional, corresponde aproximadamente a um campo, na terminologia corrente de processamento de dados. Assim, cada um dos componentes de uma linha podem ser identificados por um nome. A representação de uma tupla pode ser vista no diagrama da figura 2-2 onde os componentes aparecem sob nomes A, B, C,..., Z

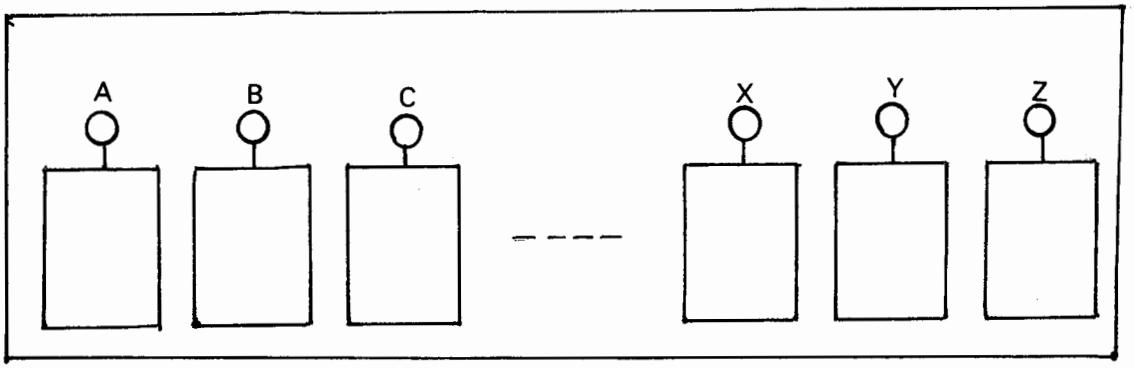


FIGURA 2.2

Um componente de uma tupla pode ser um atomo ou uma outra tupla. Um átomo é considerado indivisível constituindo a construção de mais baixo nível. Há diversos tipos de átomos,que contém valores de naturezas diversas, como por exemplo, numéricos e alfanuméricos.

Pode-se assim aplicar cada nome a todos os componentes de uma mesma coluna da tabela. A referência a uma linha pode ser feita pela posição relativa dentro da tabela, se for estabelecida uma ordem sobre as linhas, ou por critérios a serem satisfeitas pelos seus componentes. Uma forma de representar a tabela do arquivo relacional, ou tabela relacional, é a mostrada no diagrama da figura 2.3, sendo A, B, C,..., Z os nomes dos componentes de cada linha. O componente sob nome C é uma subtupla, novamente, e seus componentes tem os nomes C1, C2 e C3.

Um arquivo ligacional contém entre outros componentes uma tabela ligacional, que pode ser vista como constituida por registros especiais, chamados de ligações.

Cada uma destas ligações é contituida por uma tupla e uma tabela relacional. Uma ligação é representada na figura 2.4 onde, sob nome L, se encontra a tupla e, sob nome

T, a tabela relacional.

A	B	C			— — —	Y	Z
		C1	C2	C 3			
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

FIGURA 2.3

Desta maneira é possível estabelecer a ligação entre um registro de informações, o ligante, e um conjunto de registros de uma tabela, chamada tabela ligada, onde cada registro é chamado ligado, que assim ficam ligados ao ligante.

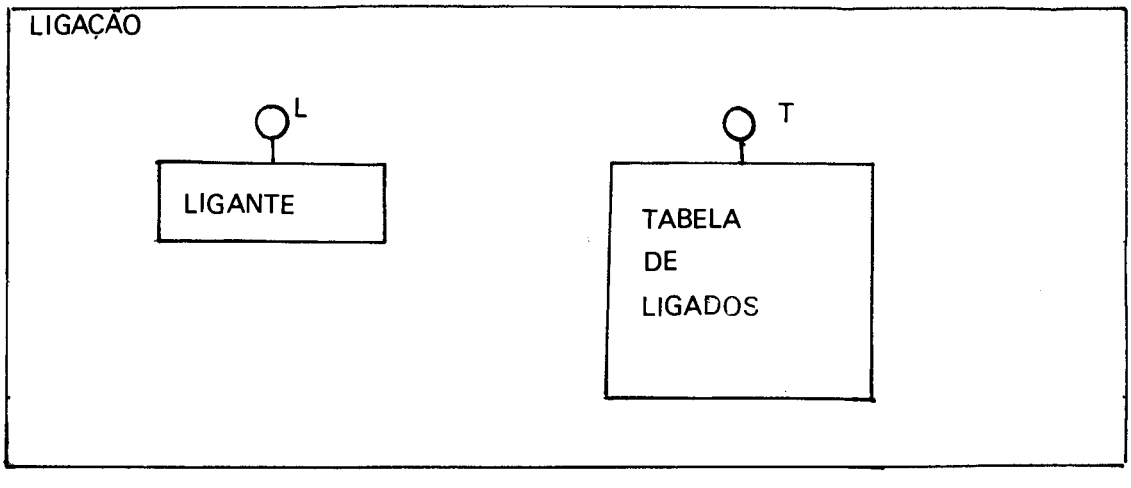


FIGURA 2.4

A referência a uma ligação pode ser feita pela sua posição relativa dentro da Tabela Ligada se for estabelecida uma ordem sobre as ligações, ou por condições a se-

rem satisfeitas pelo ligante. É possível referenciar, dentro de uma ligação, o ligante ou cada um dos ligados.

Portanto, em relação a estes, também é possível referenciar cada um dos seus componentes, átomos e tuplas.

Uma tabela deste tipo, de um arquivo ligacional, é chamado de tabela ligacional. Na figura 2-5 é representada uma tabela ligacional, contendo as ligações.

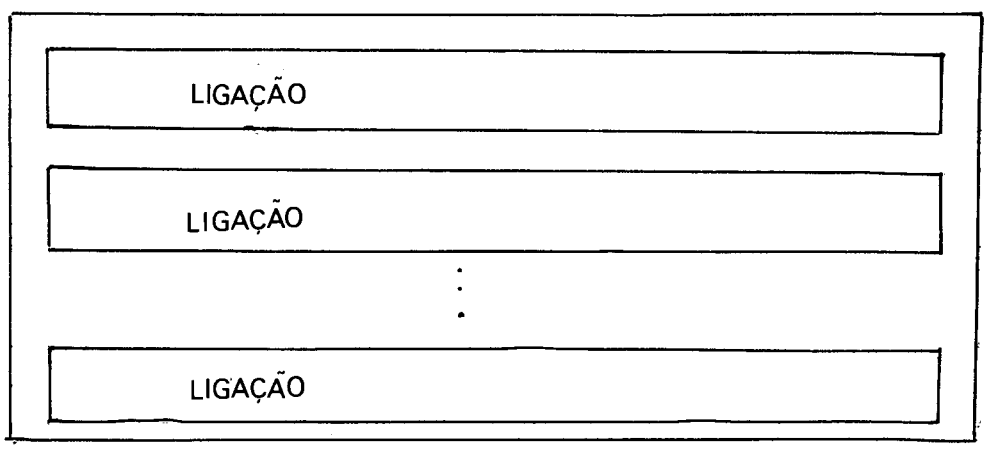


FIGURA 2.5

Tem-se assim, com estes dois tipos de arquivos, duas maneiras de guardar registros de informações, bem como de estabelecer conexão entre elas. Desta maneira o usuário pode obter informações sob os enfoques ou organizações dos tipos relacional, hierárquico ou de redes.

2.2. Conceitos Básicos

Alguns conceitos e termos são fundamentais para a compreensão do presente trabalho. Serão apresentadas noções acerca dos principais deles.

2.2.1. Pretipos de Construção

Um pretipo é um tipo padrão do Banco de Dados, isto é, pré-estabelecido. Um pretipo e um tipo podem ser compreendidos como conceitos equivalentes ao de conjunto matemático.

Um pretipo é um conjunto de construções, não necessariamente finito nem componente do acervo. É um conjunto ideal, definido por seus predicados. Portanto, todos os seus elementos, isto é, ocorrências, são possuidores dos mesmos predicados comuns ao pretipo. A condição de pertinência de um elemento ao pretipo é satisfazer aos predicados do pretipo.

Os predicados de um pretipo são, todavia, bastante genéricos para permitir a definição de tipos, ou seja, subconjuntos dos pretipos. Estes predicados estão predefinidos, embutidos no Banco de Dados, fazendo parte da interface, como uma das convenções que permitem a comunicação usuário/Banco de Dados.

Portanto, os pretipos são tipos bastante gerais, já previamente definidos (como tipos primitivos), parte integrante das convenções a nível da interface que permitem ao usuário definir novos tipos, de acordo com suas necessidades. Como exemplo de pretipos referimo-nos àqueles descritos no parágrafo 2.1.

2.2.2. Tipos de Construção

Um tipo de construção é um conjunto de cons-

truções que possuem os mesmos predicados. Um predicado necessàriamente comum às construções de um tipo é sua pertinência ao mesmo pretipo. A definição de um tipo de construção é feita pelo usuário. As regras para definir um tipo de construção dependem do pretipo de construção.

Portanto, ao se definir um tipo de construção, isto é, os predicados de um conjunto de construções, será declarado o seu pretipo. Desta maneira, todos os predicados do pretipo serão atribuidos ao tipo. Além destes, serão declarados os demais predicados que permitirão distinguir este subconjunto, o tipo, dentre os demais subconjuntos, outros tipos, pertencentes ao conjunto envolvente, o pretipo. A figura 2.6 mostra alguns tipos definidos dentro de um pretipo, em forma de diagrama.

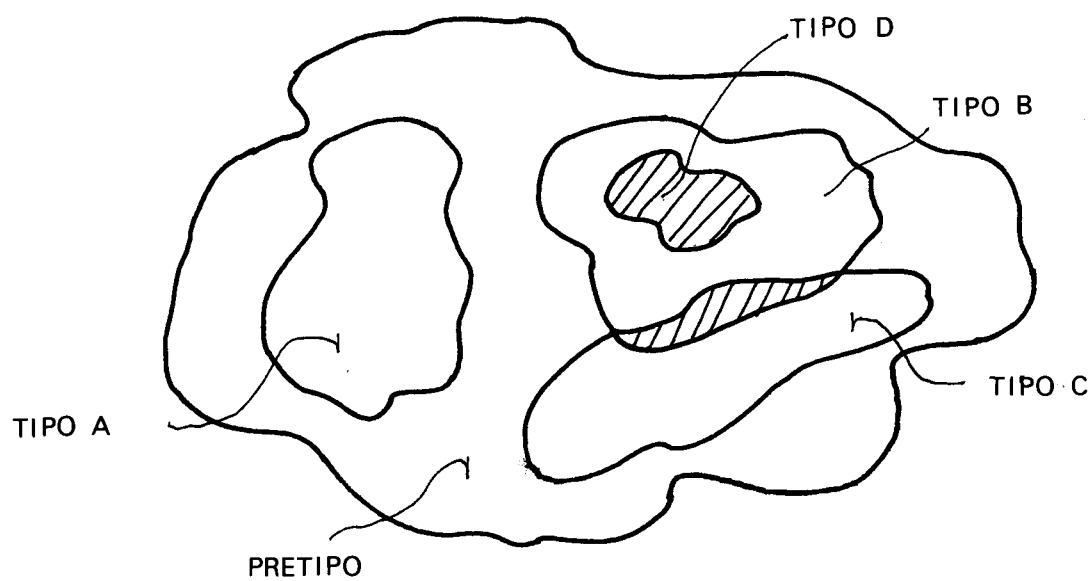


FIGURA 2.6

Neste diagrama, na área hachuriada, estão ocorrências comuns aos tipos B e C ou B e D. Considerando-se os tipos A e B ou A e C, eles constituem conjuntos disjuntos e não há ocorrencias comuns entre eles. Deve-se ainda observar que o tipo D é um subconjunto do tipo B, mais abrangente.

2.2.3. Consistência

O conceito de consistência está ligado aos conceitos de composição e de conexão.

Ao apresentar a descrição geral do acervo na seção 2.1, ficou evidenciada a maneira de descrever os tipos de construções, através dos tipos dos seus componentes, que também são tipos de construções e assim sucessivamente.

Portanto, cada tipo de construção composta é definida por composição, isto é, pelos seus componentes imediatos, em termos de tipos de construção. A descrição dos tipos de construção mais abrangentes começa pela descrição dos componentes de mais baixo nível.

Normalmente são descritos inicialmente os tipos de átomos a partir de pretipos atômicos. Compondo-se os tipos de átomos, obtem-se tipos de tuplas, e a partir destes, novos tipos de tuplas. Compondo-se tipos de tuplas obtem-se tipos de tabelas relacionais.Com tipos de tuplas e tipos de tabelas relacionais, compõe-se tipos de ligações. Com tipos de ligações, compõem-se tipos de tabelas ligacionais.

Com tipos de relações de ordens e tipos de tabelas, compõem-se tipos de arquivos. Os tipos de arquivos são relacionais ou ligacionais, correspondendo ao tipo de tabela utilizada na definição da composição, isto é, relacional ou ligacional. Com os tipos de arquivos relacionais ou ligacionais é definida a composição de tipos de acervos.

Além disso, em cada nível de definição de um tipo de construção, são definidas conexões, isto é, relações entre os componentes. Estas relações são, em geral, condições a serem obedecidas.

Por exemplo, para um tipo de tupla, pode-se estabelecer que a soma de dois átomos componentes não exceda um determinado valor limite.

Pode-se perceber que a composição e a conexão, são dois aspectos de uma mesma coisa, a consistência.

Quando qualquer tipo de construção é definido apenas por composição, a conexão dentro dos componentes imediatos ainda é mantida. Cada tipo componente traz para a construção composta as conexões que foram definidas para os componentes de uma maneira implícita.

A consistência, em última análise, é a pertinência ao tipo. Portanto, as condições de consistência são aquelas condições que permitam determinar a pertinência ou não de uma construção considerada ao tipo definido.

Coerência

O conceito de coerência está ligado aos conceitos de consistência, já apresentado, e ao conceito de persistência.

A persistência é a qualidade de uma transição observada em um lugar obedecendo a regras.

Cada regra de transição é um par formado por uma construção e um conjunto de construções do tipo, que é o conjunto de sucessores da construção. O conjunto de sucessores é um subconjunto do tipo. Em geral, quando não houver outra regra de transição o conjunto de sucessores é o próprio tipo.

Portanto, para garantir a persistência, as transições devem ser feitas de acordo com regras que são chamadas regras de transição. As regras de transição permitem fazer alterações nas construções e garantir a sua persistência.

Exemplo de uma regra de transição é só admitir que um átomo contendo o valor 'CASADO' possa ser alterado para 'VIUVO', mas não para 'SOLTEIRO', porque não é possível a um casado tornar-se solteiro, no caso da informação se referir a uma pessoa.

Pode-se portanto, garantir a coerência de uma construção quando são garantidas sua consistência e persistência.

2.2.4. Coleção

Uma coleção é uma construção em que os componentes imediatos são construções sem nome. A representação gráfica de uma coleção está na figura 2-7.

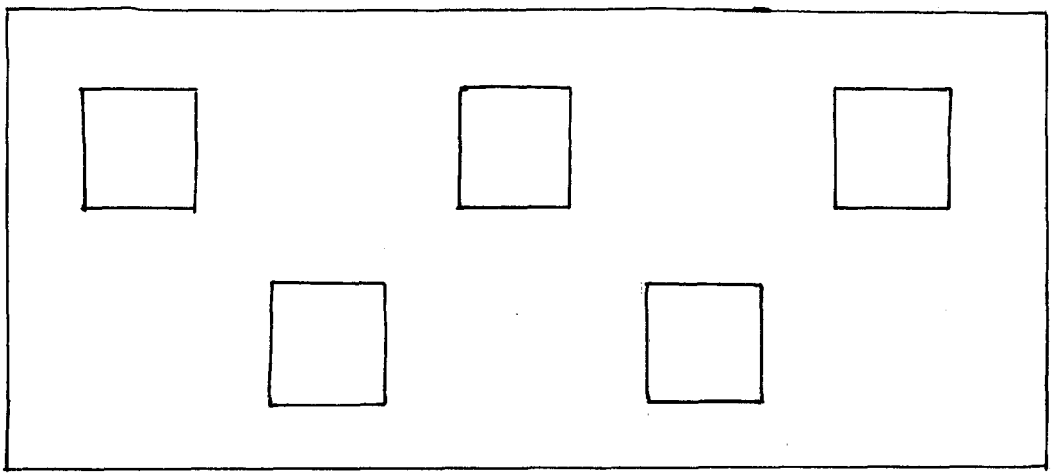


FIGURA 2.7

2.2.5. Nominação

Uma nominação é uma construção em que os componentes imediatos são construções sob nomes. O conceito de nominação corresponde ao de função matemática, isto é, a nominação é definida por pares (nome, componente imediato) . Portanto a um dado nome está associado um componente imediato. Em geral, vários nomes podem estar associados um único componente imediato. Todavia, o inverso não é válido, isto é, não pode haver vários componentes imediatos associados ao mesmo nome. A representação gráfica de uma nominação está na figura 2.7.1.

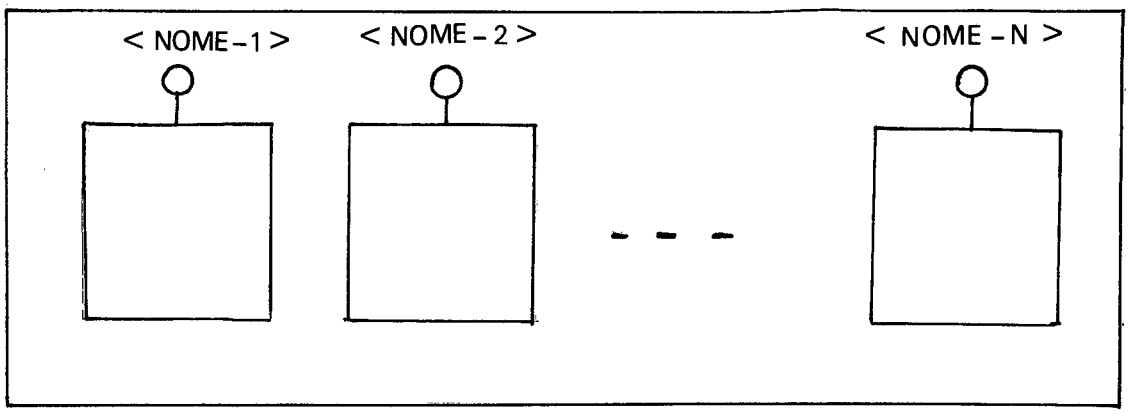


FIGURA 2.7.1

2.2.6. Coletivo

Um coletivo é um caso especial de coleção. É, portanto, uma coleção, ou seja, seus componentes imediatos são construções sem nome. Todavia, distingue-se das demais coleções porque os componentes imediatos são nominações que possuem nomes comuns. Portanto, além de cada componente imediato ser uma nominação, é necessário que os nomes dos componentes imediatos de cada nominação pertençam ao mesmo conjunto de nomes. A figura 2.8 representa graficamente um exemplo de Coletivo.

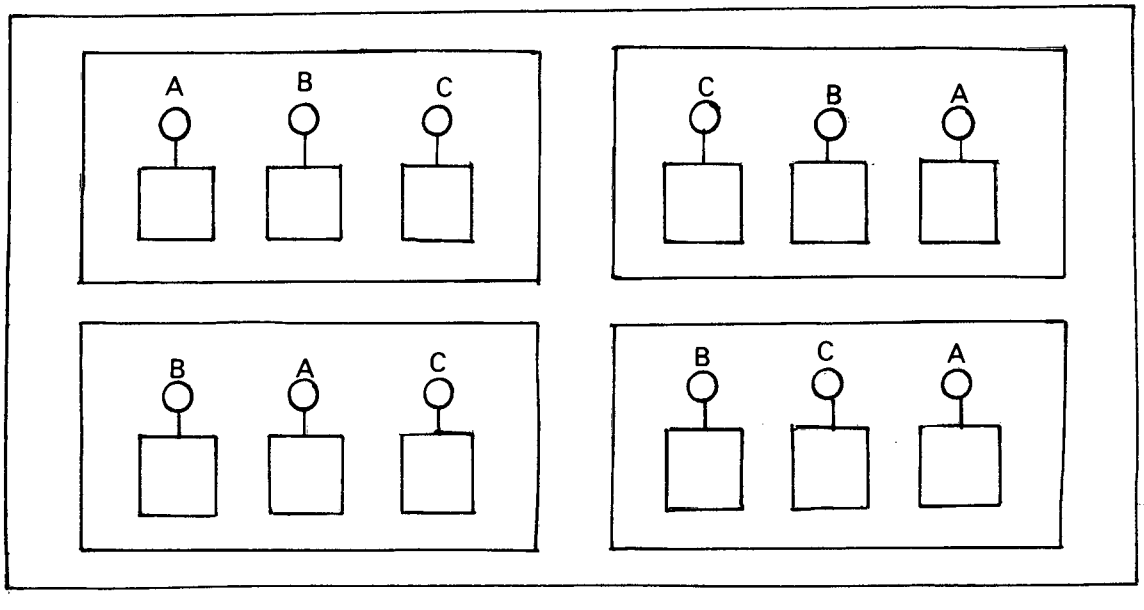


FIGURA 2.8

2.3. Átomos e Tuplas

2.3.1. Átomo

2.3.1.1. Representação Gráfica

A representação gráfica de um átomo está na figura 2.9.

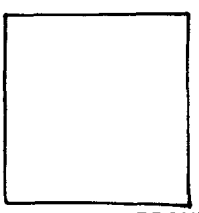


FIGURA 2.9

2.3.1.2. Descrição Verbal

Átomo é a designação usada para o conjunto de todas as construções indecomponíveis (construção representada acima).

A designação padrão do pretipo átomo é AT ou ÁTOMO.

2.4. Tupla

2.4.1. Representação Gráfica

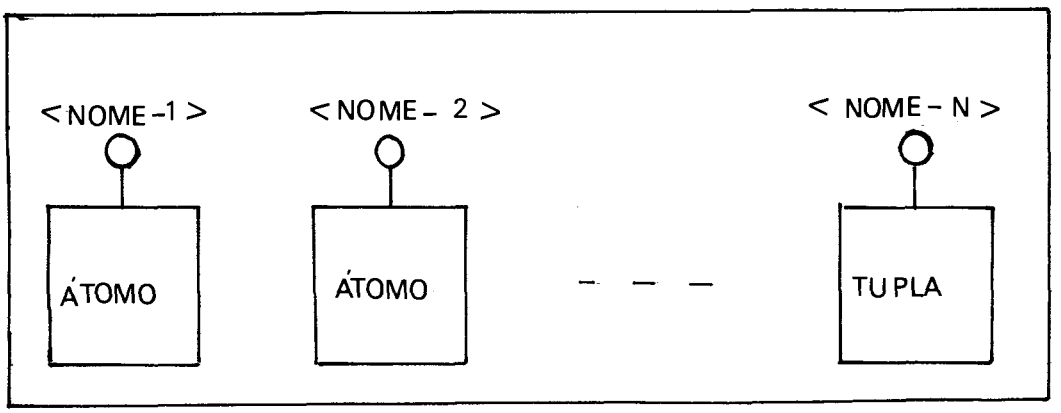


FIGURA 2.10

2.4.1.1. Descrição Verbal

Uma tupla é uma nominação de nomes escolhidos pelo usuário, sobre átomos ou tuplas. É permitida a definição de tuplas, cujos componentes sejam outras tuplas (subtuplas).

A designação padrão do pretipo tupla é TUP ou TUPLA.

2.5. Ligações e Tabelas

2.5.1. Tabela

2.5.1.1. Representação Gráfica

Uma tabela é representada graficamente como na figura 2.11, a seguir.

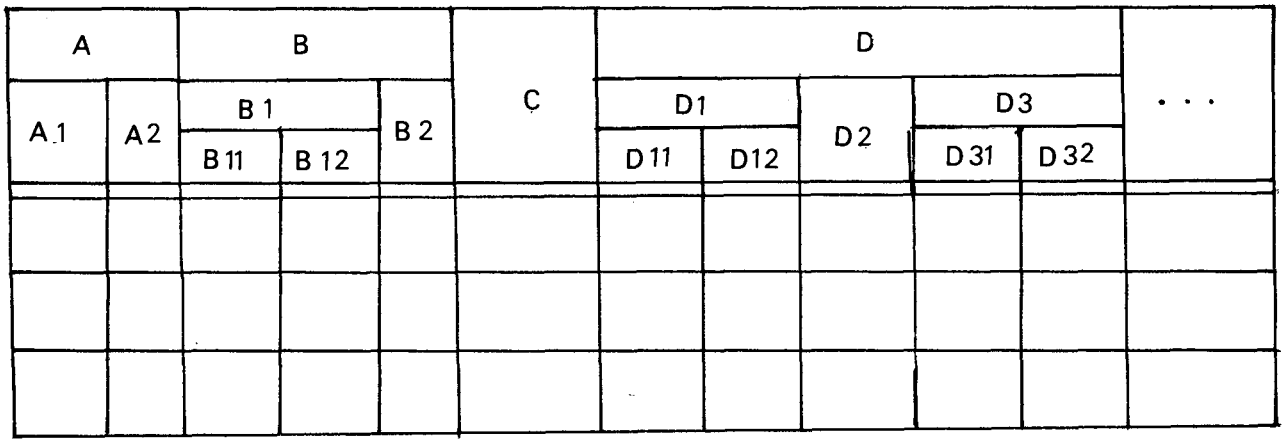


FIGURA 2.11

2.5.1.2. Descrição Verbal

Uma tabela é uma coleção. Significa que os seus componentes imediatos são construções sem nome. Todavia uma tabela não é uma coleção qualquer.

Além de serem coleções, as tabelas são também coletivos por dois motivos: primeiro, seus componentes imediatos, ou tuplas ou ligações, são nominações; segundo, estas nominações têm nomes comuns, ou seja, pertencentes ao mesmo conjunto de nomes.

Uma tabela de tuplas do mesmo tipo é uma tabela relacional. Uma tabela de ligações do mesmo tipo é uma tabela ligacional. Os nomes dentro dos componentes de uma tabela se aplicam a todas as construções de uma mesma coluna.

É possível,numa representação fatorizar estes nomes,isto é,colocá-los em evidência,para aplicá-los a todos os elementos de uma coluna.

Na representação gráfica da figura 2.11 está o diagrama de uma tabela, onde todos os nomes estão fatorizados, sobre a linha dupla.

Aparecem, além dos nomes dos componentes imediatos, também os nomes dos componentes mais internos se forem comuns, em vários níveis, constituindo uma hierarquia.

A uma sequência de nomes obedecendo a uma destas hierarquias, que permitem associar uma coluna ou conjunto de colunas, chama-se de atributo.

No exemplo da figura 2-11, são atributos, por exemplo, tanto as sequências de nomes A.A2 e D.D3.D32 como as sequências de nomes A, B.B1 e C.

No caso das tabelas em que ocorram como componentes numerações de caracteres, os números (que são os nomes) não farão parte dos atributos, pois como é variável o número de componentes de uma numeração, não é possível a fatorização de nomes.

No caso de uma tabela ligacional, em que cada ligação é constituida por uma tupla sob nome L os atributos relativos aos componentes da tupla incluirão sempre o nome L.

No caso da tabela relacional sob nome T só é possível fatorizar este nome. Assim, por exemplo, são atributos de uma tabela ligacional L.A.B.X e T.

A designação padrão do pretipo tabela é TAB ou TABELA.

2.6. Tabela Relacional

2.6.1. Representação Gráfica

A representação gráfica de uma tabela relacional é feita na figura 2.12.

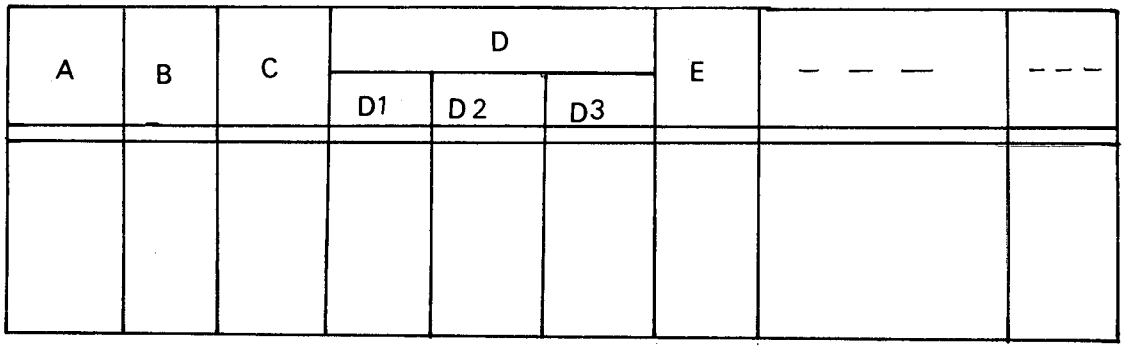


FIGURA 2.12

2.6.1.1. Descrição Verbal

Uma tabela relacional é um coletivo de tuplas do mesmo tipo. A figura 2-12 é um diagrama de uma tabela relacional.

Cada tupla incluida em uma tabela relacional chama-se uma linha da tabela, ou seja, a linha é um ponto onde aparece um componente imediato da tabela relacional. Assim, os atributos são sequências de nomes comuns a todos os elementos de uma coluna, aplicando-se em cada linha. Aplicam se à tabela relacional todas as observações já feitas para tabelas, em geral.

A designação padrão do pretipo tabela relacional é TAREL ou TABELA RELACIONAL.

2.7. Ligação

2.7.1. Representação Gráfica

A representação gráfica de uma ligação é apresentada na figura 2.13.

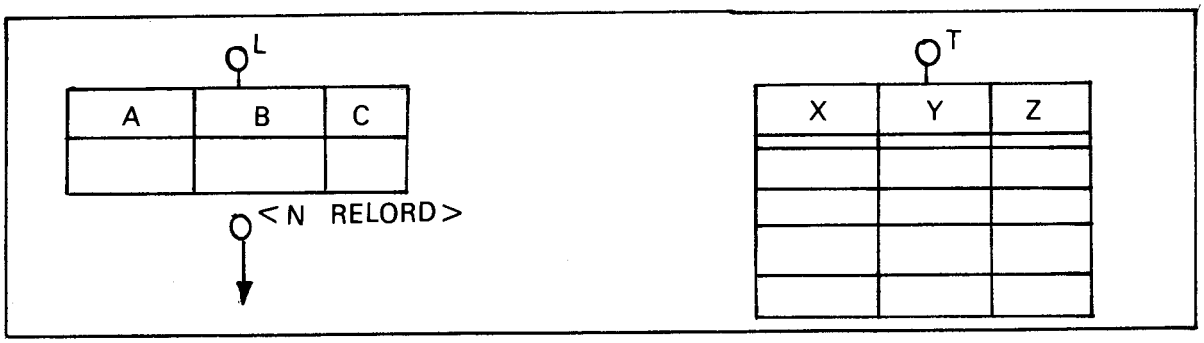


FIGURA 2.13

2.7.1.1. Descrição Verbal

Uma ligação é uma nominação sobre uma tupla, uma tabela relacional e uma relação de ordem. Os nomes da tupla, da tabela relacional e da relação de ordem são, respectivamente, L(de LIGANTE) e T(de TABELA LIGADA) e <n relord>.

A tabela relacional sob o nome T é uma tabela ligada à tupla sob nome L.

Cada linha de uma tabela ligada é chamada de ligado.

A designação padrão do pretigo ligação é LIG ou LIGAÇÃO.

2.8. Tabela Ligacional

2.8.1. Representação Gráfica

Veja representação gráfica na figura 2.14.

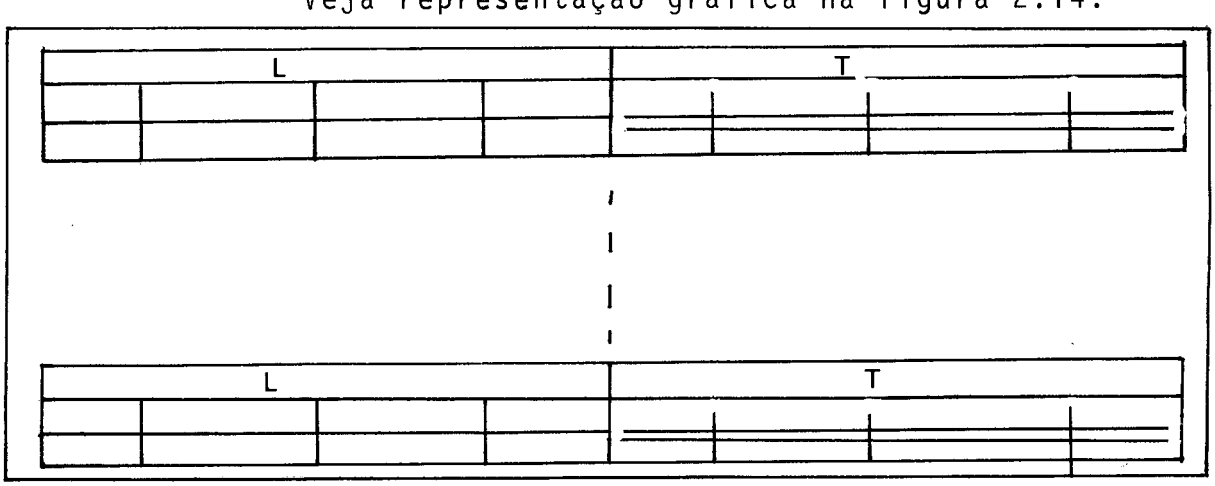


FIGURA 2.14

2.8.1.1. Descrição Verbal

Uma tabela ligacional é um coletivo de ligações do mesmo tipo.

Aplicam-se à tabela ligacional todas as observações já feitas para tabelas em geral.

A designação padrão do pretipo tabela ligacional é TALIG ou TABELA LIGACIONAL.

2.9. - Operação

Além das estruturas de informação o usuário tem a possibilidade de inserir, retirar e atualizar as informações no acervo. Descreveremos sumàriamente estas instruções cuja sintaxe encontra-se no Anexo-1.

- INSERIR CONSTRUÇÕES

A instrução de inserir construções (INSERIR), aplica-se somente para construções do acervo que são conjuntos, isto é, para tabelas. Em outras palavras a instrução serve para:

- inserir uma tupla em uma tabela relacional ou
- inserir uma ligação em uma tabela ligacional.

Além disso, a instrução de inserir permite inserir uma construção em várias tabelas simultaneamente.

- RETIRAR CONSTRUÇÕES

Com a instrução de retirar construção, conse-

gue-se o contrário da instrução de inserir construção, ou seja, ela serve para:

- retirar uma tupla de uma tabela relacional ou
- retirar uma ligação de uma tabela ligacional.

Uma única instrução permite retirar várias tuplas, e/ou ligações simultaneamente. Esta instrução não se aplica para construções portadoras de nomes.

- SUBSTITUIR CONSTRUÇÕES (nomes)

A instrução de substituir construção, aplica-se tanto para construções sob nomes (componentes imediatos de uma nominação) quanto para construções sem nome (componentes imediatos de coleções, isto é, de uma tabela relacional ou ligacional). A instrução de substituir nome é uma instrução bem restrita. Ela serve para dar um novo nome a uma construção que já tem nome,o único componente de um acervo que pode receber um novo nome é um arquivo.

- MODIFICAR CONSTRUÇÕES

A instrução de modificar construção, aplica-se, somente para tuplas, e serve para substituir os componentes imediatos que corresponde a superposição da nominação da tupla com a subnominação fornecida na instrução.

CAPÍTULO III

CONSTRUÇÕES DE INFORMAÇÃO NO SEU CONTEXTO

3.1. Informação a nível da interface do usuário e a nível de representação no computador

Para efeitos de um tratamento de controle de acesso concorrente em SBD, neste trabalho, é de nosso interesse as estruturas de informação a nível da interface do usuário. Os problemas decorrentes da implementação do sistema e em particular da parte do sistema que gerencia o controle de acesso concorrente, fogem ao assunto deste trabalho. Para caracterizar de uma forma clara, os aspectos da informação a nível da interface do usuário, e a representação desta informação em computadores, tomemos por exemplo, o conceito de tabela relacional. A nível da interface do usuário uma tabela relacional, nunca possui duas tuplas identicas, porém sob o ponto de vista de implementação de tabelas relacionais, pode ser vantajosa a criação de duplicatas, isto é, duas representações para a mesma tupla, com o objetivo de tornar mais eficiente o acesso no disco ou na memória principal a representação desta tabela relacional.

É claro que a nível de representação das informações em computadores, os endereços destas representações ficam definidos, como por exemplo pelo endereço relativo ou absoluto no disco magnético ou na memória principal, como já é bastante conhecido. Porém, a nível de estrutura de infor-

mação, carece-se de uma forma de endereço, pelos quais possamos endereçar construções. Isto é um endereço de informações pois o usuário não conhece ou não é desejável a ele o conhecimento de endereço de representações. Este endereço deve satisfazer alguns requisitos como:

- Em que nível de agregação as nominações ou coleções estão disponíveis.
- Que propriedades podem ser utilizadas para o endereçamento de construções (independentemente de sua representação)
- Como endereçar as mesmas construções em contextos diferentes.

Para estas questões introduziremos a conceitos de ponto, que vem suportar estas e outras questões concernentes a identificação de construções.

3.2. Ponto

Um ponto é definido como uma sequência de pares, (nome,construção). O primeiro par da sequência definidora, consiste do nome vazio e a construção de referência, a qual o ponto esta sendo considerado. No caso de coleções o nome vazio é inserido na posição nome do par, da sequência definidora. Veremos a seguir uma série de exemplos que tornaram mais claro este conceito.

EXEMPLO 3.1.

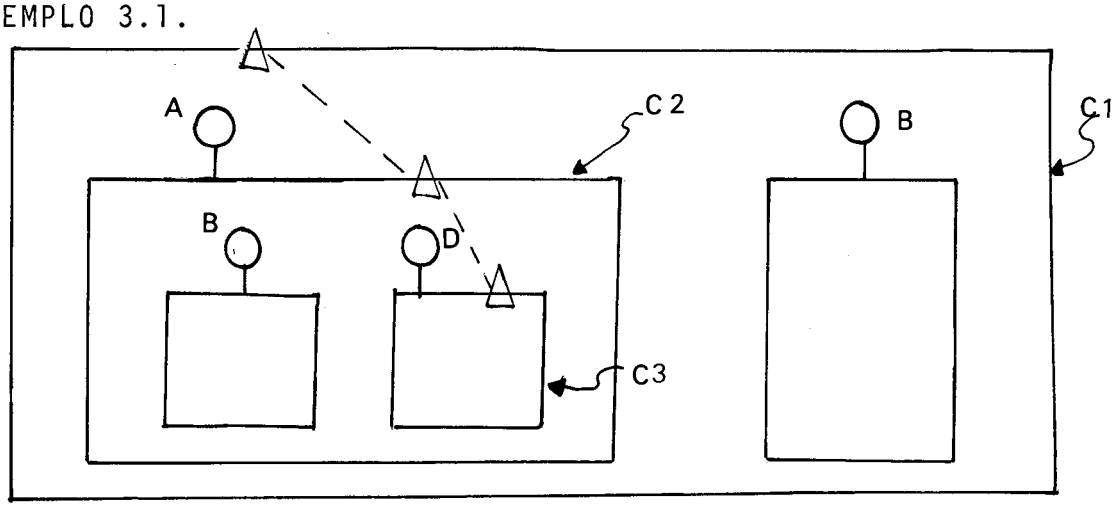


FIGURA 3.2

Convenções: Δ - simboliza o ponto da construção

---- - sequencia definidora

O ponto onde aparece a construção sob nome D é definido como:

$$P = (-,C_1)(A,C_2)(D,C_3)$$

C_1, neste exemplo é a construção de referência.

Utilizamos C1, C2 e C3, somente para efeitos de referência na sequência definidora do ponto em questão.

EXEMPLO 3.1 - Tomando como construção de referência ao invés da construção C1, a construção C2, a identificação na figura 3.2 da construção sob nome D seria dada da seguinte forma:

$$P' = (-, C_2)(D, C_3)$$

Observamos, que embora a construção C_2 tenha um nome, a saber A, este nome não aparece quando esta é a construção de referência; o que evidencia que a identificação de um ponto não leva em consideração construções fora do contexto de referência.

EXEMPLO 3.2.: Este exemplo visa motivar a necessidade da criação de uma correspondência entre pontos, que será vista quando tratarmos sobre pontos correspondentes entre vários acervos. O conceito de ponto é estático, no sentido de que dada uma alteração na construção de referência, a definição de um ponto P antes da modificação, já não tem mais efeito, isto é, já não corresponde a definição de um ponto válido na construção de referência modificada, como poderemos ver através deste exemplo:

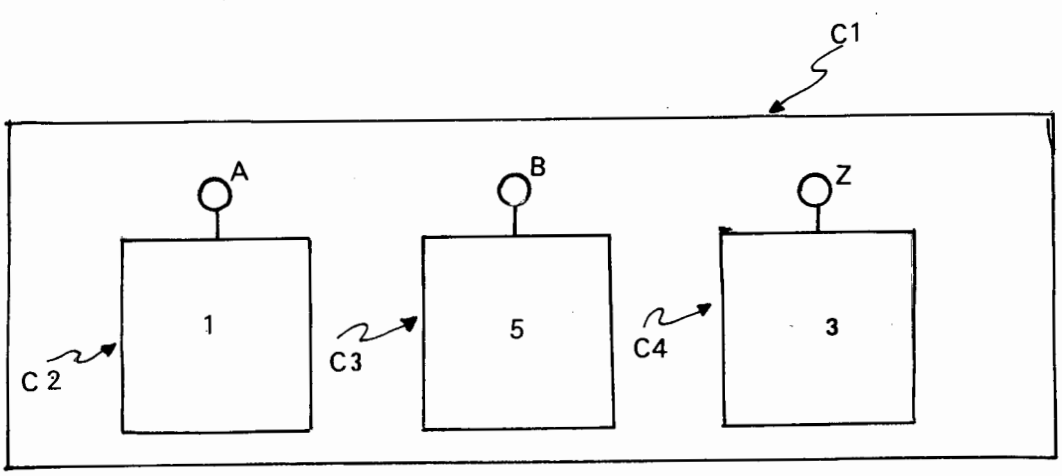


FIGURA 3.3

se uma modificação, for realizada na construção C_2 da figura 3.3 transformando-a da construção 1 para a construção 7, teríamos o representado na figura 3.4

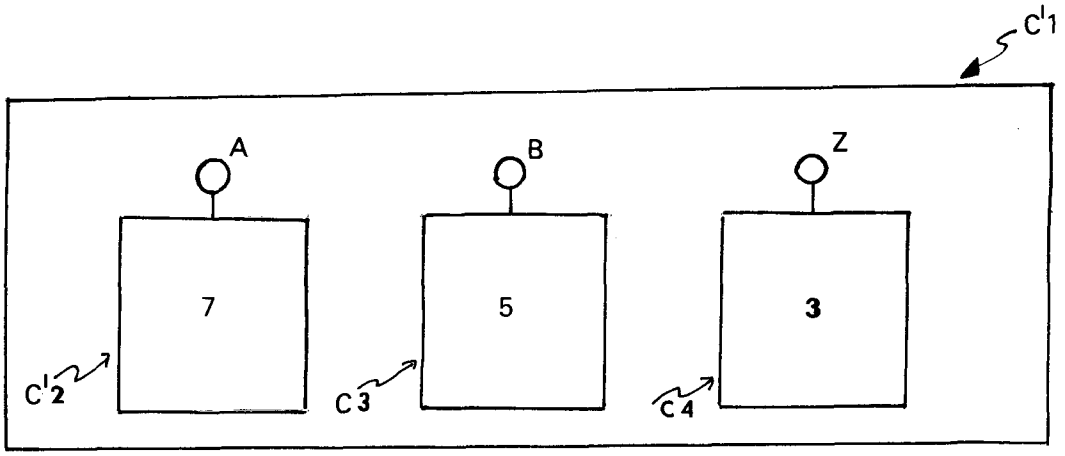


FIGURA 3.4

Assim o ponto onde na figura 3.3, fica a construção C_2 é dado por:

$$P_1 = (-, C_1)(A, C_2)$$

e o ponto onde fica, a construção sob nome A na figura 3.4, ou seja a construção C_2' é dado por:

$$P_1' = (-, C_1')(A, C_2')$$

podemos observar que as sequências definidoras de P_1 e P_1' são diferentes, em razão da modificação realizada.

Se tomarmos a construção sob nome Z, das figuras 3.3 e 3.4, podemos verificar também que o ponto onde fica esta construção na figura 3.3 é diferente do ponto onde fica esta construção na figura 3.4. Senão vejamos.

Para figura 3.3 temos:

$$P_2 = (-, C_1)(Z, C_4)$$

e

Para figura 3.4 temos:

$P_2' = (-, C_1')(Z, C_4)$

como a construção C_1 é diferente da construção C_1', pois duas construções são iguais somente quando são identicas, vemos então que P_2 e P_2' são pontos diferentes, embora a construção sob nome Z não tenha sofrido qualquer modificação.

Posteriormente definiremos uma regra de correspondencia entre pontos de acervos dérivados. Isto é uma regra que associe pontos de um dado acervo A, com pontos do acervo A modificado.

Apresentaremos a seguir, algumas definições e observações com o objetivo de auxiliar no tratamento de áreas para proteção.

3.3. Árvore de Pontos

Uma árvore de pontos é o conjunto de todos os pontos de uma construção, representado através de uma árvore.

Sob as regras de formação de uma árvore de pontos a partir de uma construção, introduziremos através de exemplos.

EXEMPLO 3.3 : Apresentaremos uma construção e sua árvore de pontos correspondentes.

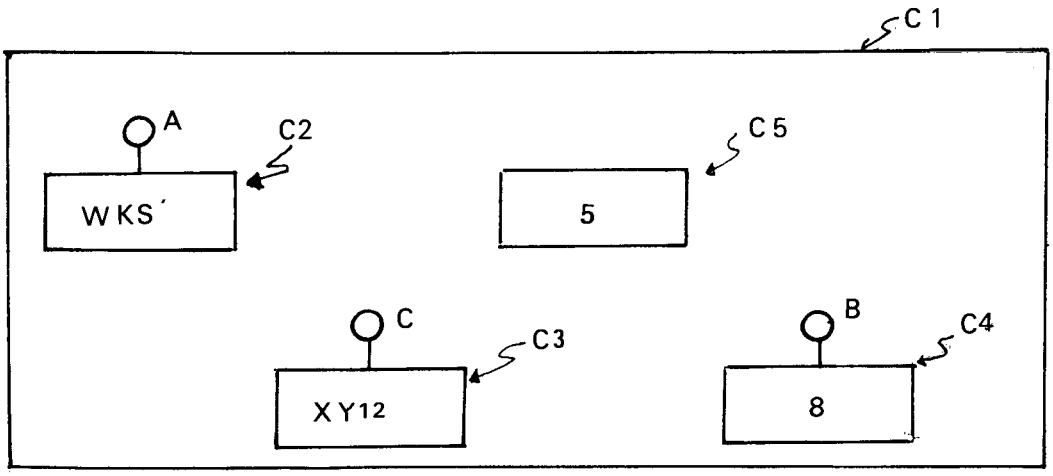


FIGURA 3.5

Árvore de pontos correspondente a construção de referência C_1, da figura 3.5.

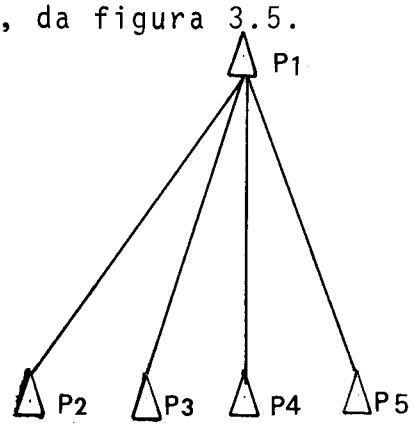


FIGURA 3.6

E

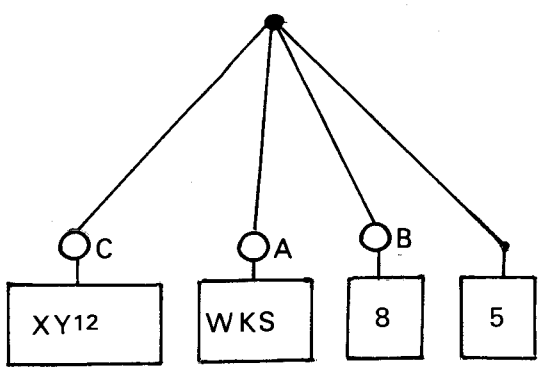


FIGURA 3.7

as definições dos pontos estão dadas abaixo:

$$P_1 = (-, C_1)$$

$$P_2 = (-, C_1)(A, C_2)$$

$$P_3 = (-, C_1)(B, C_4)$$

$$P_4 = (-, C_1)(C, C_3)$$

$$P_5 = (-, C_1)(-, C_5)$$

Para uma representação como mostrada na figura 3.7, os nós da árvore que são representados por um ponto cheio (.), indicam construções sem nomes, ou seja o que na sequencia definidora de um ponto aparece como "-", para os com

ponentes com nomes a representação é feita por uma bola (O) e o nome do componente.

3.4. Caminho em Uma Árvore de Pontos

Um caminho em uma árvore de pontos de um ponto P_1, para um ponto P_2, pode ser visto na figura 3.8, demarcado por linhas tracejadas.

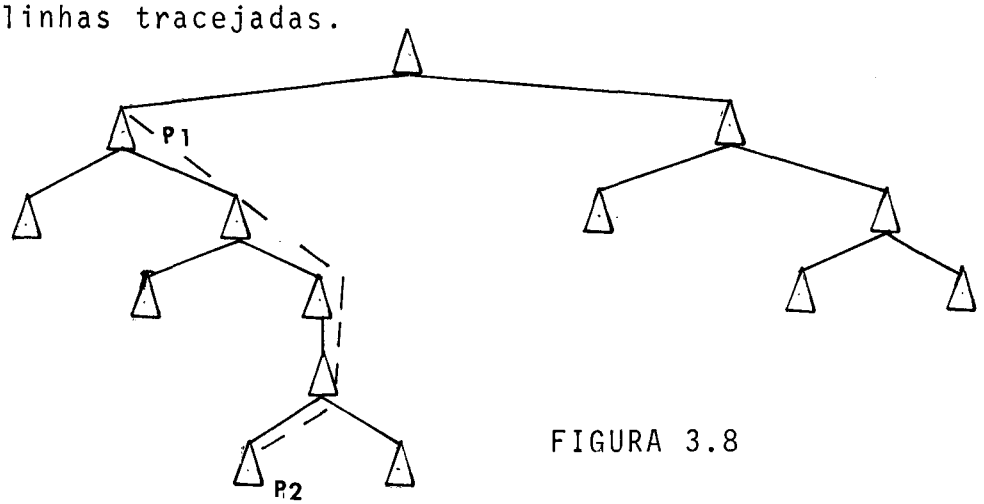


FIGURA 3.8

As linhas tracejadas também servem, para identificar o caminho de P_2 para P_1. A definição de um caminho de um ponto a outro, obedece a mesma definição de caminho em árvores tradicionais abordadas no tratamento de estrutura de dados. Para nossos objetivos, só estamos interessados em caminhos diretos, isto é, sem repetição de nós.

3.5. Cone de Pontos

Dado um ponto P em uma árvore de pontos, definimos como cone de pontos associado ao ponto P, o conjunto formado pelos pontos P', tais que, existe um caminho de P de cima para baixo até P'. O ponto P é denominado vértice do cone.

EXEMPLO 3.4.: Dada a árvore de pontos da figura 3.9, o cone de pontos associado ao ponto P está envolto, por linhas tracejadas.

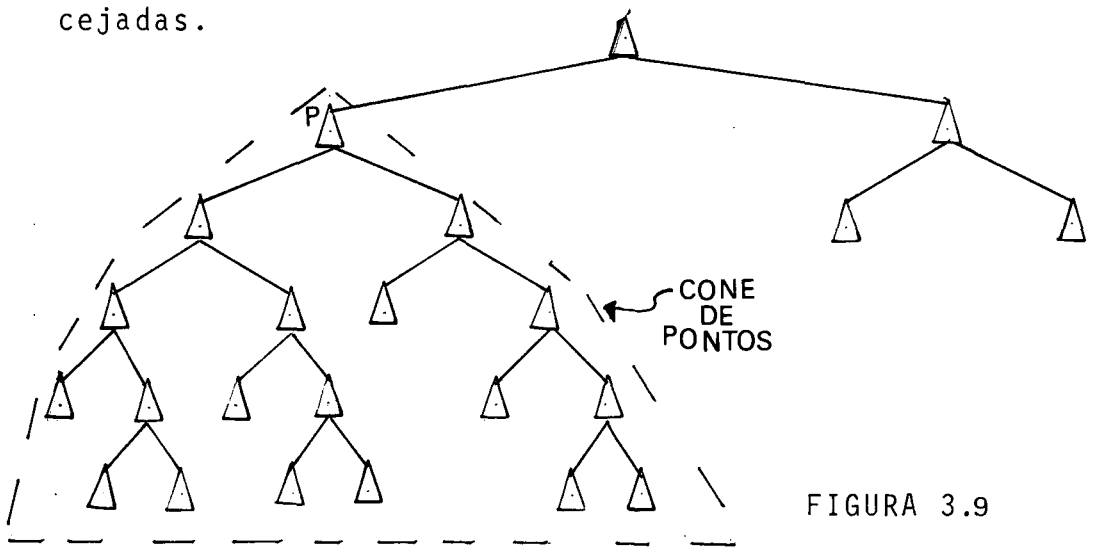


FIGURA 3.9

Em uma terminologia corrente para árvores, P é a raiz da subárvore e o cone é o conjunto de todos os nós da subárvore de raiz P.

3.6. Endereço de Ponto

O endereço de ponto é a forma pela qual o usuário tem para endereçar ou identificar os pontos de seu interesse. Um endereço de ponto pode identificar vários ou nenhum ponto como veremos em exemplos a seguir. A sintaxe para endereços de pontos se encontram no anexo 1.

EXEMPLO 3.5: Identificar os pontos das tuplas na tabela relacional da figura 3.10, contida em um arquivo relacional de nome EMP, mas somente as tuplas para as quais a construção sob nome SAL, seja maior do que 10.000.

ENDEREÇO: ACERVO.EMP.TR.(C SAL > 10000)

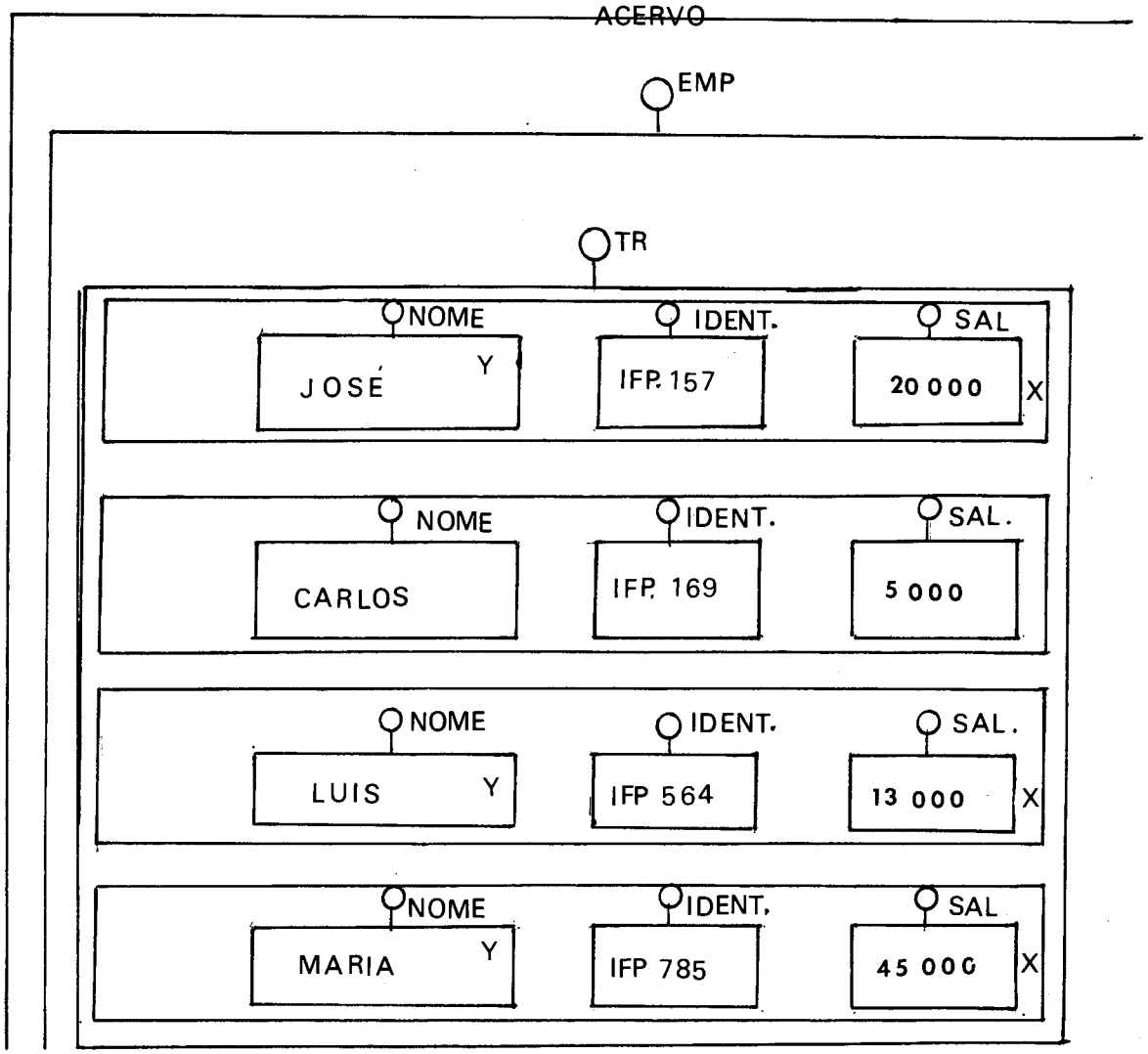


FIGURA 3.10

As tuplas nos pontos identificados estão mostradas na figura 3.10, com a marca X.

EXEMPLO 3.6.: Identificar os nomes para os empregados no arquivo EMP, da figura 3.10, que ganham mais do que 10.000.

ENDEREÇO: ACERVO.EMP.TR(C SAL > 10.000).NOME

os pontos identificados por esse endereço estão marcados na figura 3.10 por Y.

EXEMPLO 3.7.: Este exemplo apresenta um endereço de ponto, que não endereça nenhum ponto na figura 3.10

ENDEREÇO: ACERVO.EMP.TR.(C NOME = 'PEDRO')

CAPÍTULO IV

COERÊNCIA NA BASE DE DADOS

Como já introduzido anteriormente o conceito de coerência define os acervos admitidos na base de dados bem como as transições admitidas de um acervo para outro acervo resultando de uma operação de alteração.

As definições de coerência refletem a composição de um acervo: Todas as especificações que podem ser verificadas em cima de um componente "sozinho", isto é, em cima da construção fora do seu contexto, serão ligadas ao tipo dessa construção. Regras que abrangem também o contexto dessa construção tem que ser associadas ao tipo de construção que com a menor abrangência possível inclui todos os componentes afetados (por exemplo o conceito de "chave", pertence a tipos de tabelas e não a tipos de tuplas, embora sendo cada valor chave constituido por componentes de tuplas).

As regras de consistência podem ser estabelecidas em termos da lógica dos predicados, este fato utilizaremos posteriormente no capítulo IX.

Daremos a seguir alguns exemplos:

EXEMPLO: Tipo de tupla com designação ADMIN, cinco componentes imediatos VAG, ANASC, GRUPO, IDA, OCU sendo esses componentes átomos dos tipos com designação NATEX e SIGRUP, respectivamente.

Além da composição existe a conexão indicada pelas linhas da figura 4.1 entre os componentes

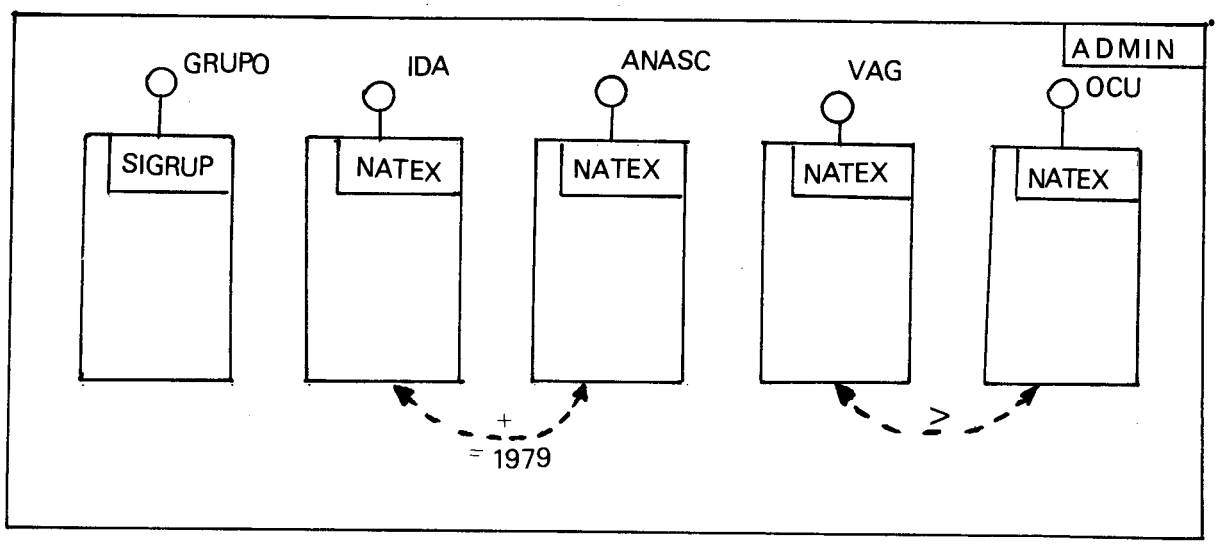


FIGURA 4.1

Este tipo de tupla com designação ADMIN, teria a seguinte definição:

ADMIN: TIPO DE TUPLA TAL QUE

COMPOS

GRUPO → SIGRUP,

IDA → NATEX,

ANASC → NATEX,

VAG → NATEX,

DCU → NATEX

CONEX

(C OCOR.ANASC + C OCOR.IDA) = 1979

∧

C OCOR.VAG > C OCOR.OCU

Podemos representar então através de uma árvore de pontos um exemplar deste tipo de tupla por:

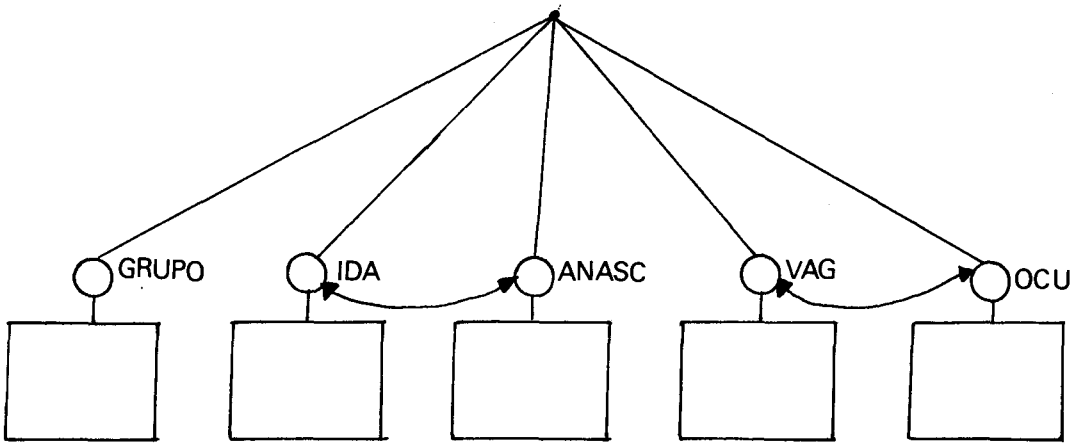


FIGURA 4.2

O identificador padrão OCOR, é o identificador da ocorrência em exame, quando da avaliação pelo sistema se uma dada construção pertence ou não a um dado tipo. Por exemplo,

EXEMPLO 4.1: Um usuário U1, realiza uma operação de inserção de uma tupla em uma tabela relacional que possui tuplas do tipo ADMIN, segundo a instrução abaixo:

```
INSERIR EM ACERVO.ARQ.TR
 OBTER TUPLA COMPOR(GRUPO:=32
                    IDA:=20
                  ANASC:=1959
                    VAG:=10
                    OCU:=20)
```

Esta tupla a ser inserida está apresentada na figura 4.3

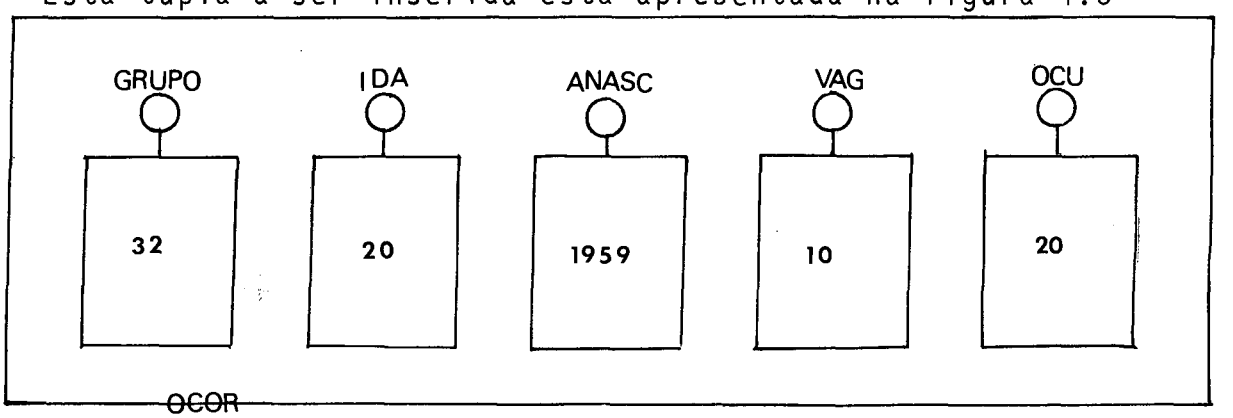


FIGURA 4.3

Para a avaliação se a tupla da figura 4.3 poderia ou não ser inserida na tabela relacional o sistema analisaria a ocorrência em questão verificando se esta ocorrência que seria identificada com identificador padrão OCOR é do tipo ADMIN. No caso da tupla da figura 4.3 esta tupla não pertence ao conjunto de tuplas definidas pelo tipo ADMIN, pois a conexão C OCOR.VAG $\geq$ C OCOR.OCU, não é verificada dado que C OCOR.VAG é a construção 10 e C OCOR.OCU é a construção 20.

Com o objetivo, posterior quando do tratamento de áreas conectadas algumas definições serão dadas a seguir.

4.1. Consistência Implícita

É a consistência inerente ao pretipo abrangente, dado a definição de um tipo de construção.

Como já definimos anteriormente, um pretipo é um tipo de construção que tem a sua definição embutida na interface. Por exemplo , temos os pretipos átomo, tupla, tabela relacional etc... . Quando da definição de um tipo de construção, uma referência é feita a um pretipo, a consistência devido a esta referência a este pretipo denomina-se então consistência implícita.

EXEMPLO 4.2: Seja definição de um tipo de tupla X

```
X : TIPO DE TUPLA TAL QUE
    A → TIPA,
    B → TIPB,
    C → TIPC
    CONEX
    C OCOR.A ≥ C OCOR.B
```

Temos então que pela referência na definição do tipo de construção X, ao pretipo tupla, temos como consistência implícita que:

- Uma ocorrência de uma construção que pertence ao tipo X, deve ser uma nominação.

Se definirmos um tipo de construção TAB dada na seguinte forma:

```
TAB: TIPO DE TABELA RELACIONAL TAL QUE
        COMPOS
         X
```

Temos pela referência na definição do tipo de construção TAB, a seguinte consistência implícita devido a referência do pretipo TABELA RELACIONAL.

- Uma ocorrência de uma construção que pertence ao tipo TAB, tem que ser um coletivo. De outra forma, tem que ser um conjunto de tuplas do mesmo tipo, e que não possua dois elementos, ou seja duas tuplas iguais.

Assim para cada pretipo referenciado na definição de um tipo teríamos as definições embutidas na interface.

4.2. Consistência Explícita

É toda consistência definida dado um tipo de construção, excluída a consistência implícita.

Temos a seguinte subdivisão para consistência explícita:

- consistência "longitudinal"

É a consistência que define a "composição",ou os componentes imediatos para um tipo de construção.

- consistência "lateral"

É a consistência que define as "conexões" entre os componentes para um tipo de construção.

EXEMPLO 4.3: Seja a definição de um tipo de tupla T1P1, dado por:

```
T1P1: TIPO DE TUPLA TAL QUE
      COMPOS
       ELEM1 → T1,
       ELEM2 → T2,
       ELEM3 → T3
       CONEX
       C OCOR.ELEM1+C OCOR.ELEM2=C OCOR.ELEM3
```

Temos como consistência longitudinal o seguinte:

- Uma tupla de tipo TIP1, possui 3 componentes imediatos de nomes ELEM1, ELEM2 e ELEM3 e cada um desses componentes imediatos pertencem aos tipos T1, T2 e T3 respectivamente.

e como consistência lateral:

- Para uma tupla de tipo TIP1, os componentes imediatos sob nomes ELEM1 e ELEM2 somados resultam na construção sob nome ELEM3. Dizemos que os componentes sob nomes ELEM1, ELEM2 e ELEM3 estão conectados , e utilizamos a seguinte representação mostrada na figura 4.4, para uma representação do tipo de tupla T1P1

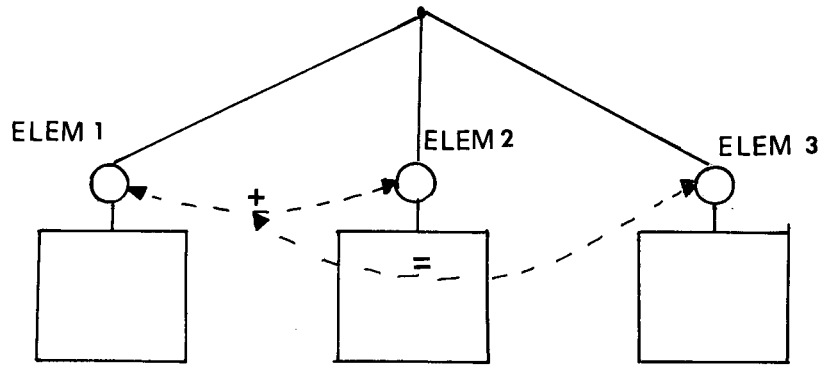


FIGURA 4.4

- Consistencias laterais ajustáveis ou conexões ajustáveis

Existem conexões que quando violadas, podem ser ajustadas pelo sistema a pedido do usuário.

EXEMPLO 4.4: Tomemos a definição de um tipo de tupla, TU.

```
TU: TIPO DE TUPLA TAL QUE
    COMPOS
    A → TIP1,
    B → TIP2
    CONEX
      C OCOR.A = C OCOR.B AJUSTAR (1)
```

Através deste exemplo, o sistema a pedido do usuário ajusta o valor da construção sob nome B, sempre que este valor for diferente ao valor encontrado em uma ocorrencia deste tipo de tupla para a construção sob nome A. Se desejassemos que também quando da alteração do valor da construção sob nome B, a construção sob nome A fosse ajustada escreveriamos no lugar de expressão (1) a seguinte expressão:

AJUSTAR C OCOR.A = C OCOR.B AJUSTAR

Nem todas as conexões são ajustáveis, isto é, existe uma possibilidade de ajuste por parte do sistema, logo o que

definimos como conexões ajustáveis, são as conexões que podem ser ajustadas e mais que haja um pedido explícito do usuário, para este ajuste.

EXEMPLO 4.5: Definimos um tipo de tupla NAJUST

```
NAJUST:TIPO DE TUPLA TAL QUE
        COMPOS
        A → T1,
        B → T2,
        C → T3,
        D → T4
        CONEX
    AJUSTAR C OCOR.A + C OCOR.B
                >
    C OCOR.C + C OCOR.D AJUSTAR
```

Vemos por este exemplo, que o usuário pediu um ajuste, porém trata-se de conexão não ajustável pois, dado um estado inicial consistente de uma ocorrência deste tipo de tupla NAJUST mostrado na figura 4.5

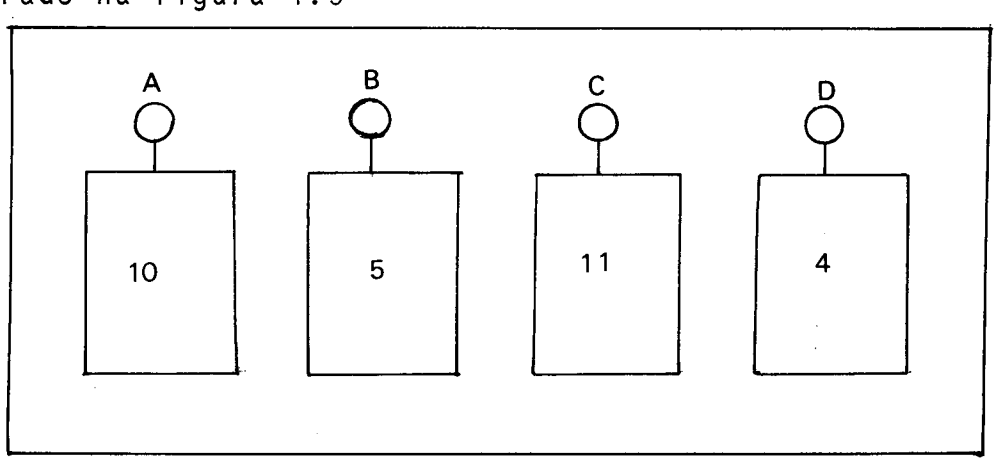


FIGURA 4.5

e uma posterior modificação desta ocorrencia para a ocorrência mostrada na figura 4.6.

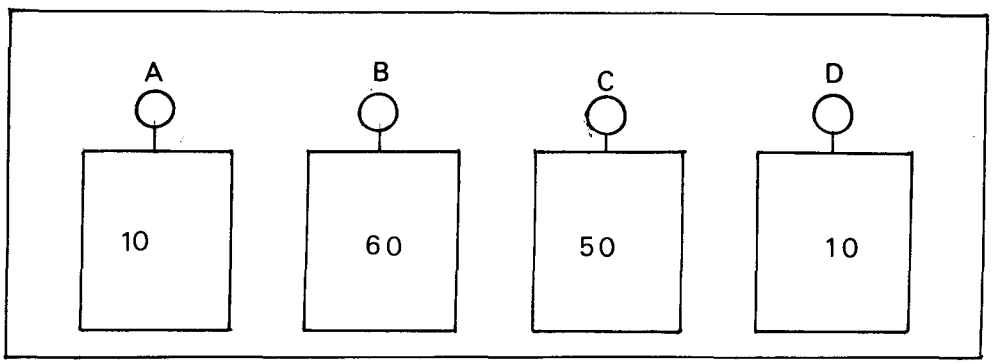


FIGURA 4.6

existe uma infinidade de valores que poderiam ser atribuidos as construções sob nomes A e B, de tal forma a ajustar esta conexão.

Na parte concernente, a área conectada trataremos novamente dos aspectos enunciados anteriormente, e que implicações estes aspectos terão relativamente a proteção.

- Consistência explícita e consistência implícita

  Quando definimos consistência explícita, dissemos que era toda a consistência definida dado um tipo de construção excluída a consistência implícita. Esta definição tem consequências não evidentes como mostraremos no exemplo 4.6.

EXEMPLO 4.6: Definimos um tipo de tabela relacional TAB, dado a seguir, a partir do tipo de tupla TIPTUP.

```
TIPTUP : TIPO DE TUPLA TAL QUE
         COMPOS
         A → TA,
         B → TB,
         C → TC,
         D → TD
```

```
TAB:TIPO DE TABELA RELACIONAL TAL QUE
    COMPOS
      TIPTUP
    CONEX
(1)   CARD C OCOR = CARD ESTREIT C OCOR
                          PARA A, B
(2)   CALC  SOMA EM OCOR.TUP.A
                =
      CALC SOMA EM OCOR.TUP.C
```

A expressão (1), na definição do tipo de tabela relacional TAB, define a chave para uma ocorrência deste tipo de tabela. Esta definição não sendo realizada, o que significa, retirar a expressão (1) da definição do tipo de tabela TAB, o sistema considerará como chave ao invés de combinações diferentes dos componentes, sob nome A e B, para cada tupla que aparece em uma ocorrência do tipo TAB, combinações diferentes dos componentes sob nomes A,B, C e D. Isto porque, não podem coexistir em uma tabela relacional duas tuplas identicas, dado que como vimos no exemplo 4.2 trata-se de uma consistência implícita. Então no caso de definição de chaves para tabelas relacionais ou ligacionais esta definição não será considerada como uma consistência explícita,pois trata-se de uma restrição a uma consistência implícita(ao invés de considerar A, B, C e D, considerar A e B como chave) para o pretipo abrangente (no caso tabela relacional).

O mesmo já não podemos dizer sobre a expressão (2), pois se retirada da definição do tipo de tabela re-

lacional TAB, nada relacionado com a consistência definida, por esta expressão seria mantida pelo sistema. Isto se deve, ao fato de que a consistência definida pela expressão(2), não é uma restrição a uma consistência implícita ao pretipo tabela relacional. Assim a consistência definida pela expressão (1) não define uma consistência explícita, e consistência definida pela expressão (2), define uma consistência explícita que no caso é uma consistência lateral.

Para o tratamento do controle de acesso concorrente em SBD, estamos interessados particularmente em consistências explícitas e as consequencias resultante pela existência destas consistências explícitas em termos dos pontos envolvidos devido a definição destas consistências para ocorrências de um tipo de construção.

- Persistência

O aspecto de definição de persistência para um tipo de construção não é relevante para o assunto deste trabalho,pois a definição de uma persistência para um tipo de construção não associa o ponto de ocorrência deste tipo com outros pontos no mesmo acervo, e sim com o ponto de ocorrência deste tipo de construção no acervo modificado.

CAPÍTULO V

TRANSAÇÃO E PROTEÇÃO

5.1. Bloco de Transação

É o conjunto de instruções limitadas sintaticamente pela instrução CONSIDERAR TRANSAÇÃO.

EXEMPLO 5.1: Exemplo de um bloco de transação:

```
(CONSIDERAR TRANSAÇÃO
      < INSTRUÇÃO-1 >
      < INSTRUÇÃO-2 >
            .
            .
            .
      < INSTRUÇÃO-n > )
```

5.2. TRANSAÇÃO

Definimos como transação o processo definido pela execução de um bloco de transação.

O bloco de transação define sintaticamente o processo, e a transação é o processo. Uma transação relativamente ao sistema satisfaz as seguintes propriedades:

1. Ao término de cada transação o sistema verifica se as ações realizadas por esta transação transformaram um acervo consistente em outro acervo consistente.
2. Uma transação pode violar a consistência de um acervo. Porém nunca terminar com um acervo inconsistente.

Dadas estas duas propriedades, podemos então explicar quais os aspectos devido a transações são relevantes no tratamento de controle de acesso concorrente em SBD. Suponhamos que um usuário U1, defina uma transação e que ao final da realização desta transação o sistema detete que o acervo resultante pela execução das ações determinadas por essa transação, já não é mais consistente. Neste caso o sistema deve providenciar o que chamamos de reconstrução regressiva, que significa desfazer todas as ações definidas nesta transação que alteraram o acervo, reconstituindo o acervo de forma a deixa-lo no estado inicialmente encontrado antes da execução da transação. Assim para toda transação que transforma um acervo consistente em um acervo inconsistente, o sistema realiza uma reconstrução regressiva.

5.3. Bloco de Proteção

Um bloco de proteção é definido sintaticamente pelas instruções de ESTABELECER PROTEÇÃO e ABANDONAR PROTEÇÃO(SINTAXE NO ANEXO 1).

5.4. Proteção

É o processo determinado pela execução de um bloco de proteção.

Novamente apresentamos a distinção realizada entre a definição de um processo (BLOCO DE PROTEÇÃO) e a execução do processo definido.

O bloco de proteção tem a finalidade de forne

cer ao sistema que pontos são do interesse do usuário antes que o usuário inicie as operações sobre estes pontos. Esta técnica também é utilizada em sistemas operacionais, onde o usuário fornece ao sistema que recursos (unidades de fita, unidades de disco, etc...) vai utilizar a priori. Ao usuário do SBD, realizar um pedido de ESTABELECER PROTEÇÃO, é de seu desejo que um conjunto de pontos seja protegido contra outros usuários, e que ele possa realizar as operações desejadas sobre este conjunto de pontos. Assim dizemos que ao usuário fornecer uma instrução de ESTABELECER PROTEÇÃO, ele deseja que o sistema realize uma proteção sobre um conjunto de pontos e que mantenha esta proteção (capítulo VIII), até que haja uma liberação deste conjunto de pontos que é realizada quando da execução da instrução de ABANDONAR PROTEÇÃO.

5.5. ENCAIXAMENTO ENTRE BLOCO DE PROTEÇÃO E BLOCO DE TRANSAÇÃO

Se o usuário pede uma proteção sobre um conjunto de pontos é claro que esta proteção deve se manter até que o usuário abandone ou libere esta proteção. Em alguns SBD, ao usuário definir uma transação, automaticamente ele está definindo um bloco de proteção, este esquema porém é pouco flexível, porque não garante que duas transações identicas definidas pelo mesmo usuário e sendo intercalada por outras transações tenham o mesmo efeito, pois os pontos foram liberados no término da primeira transação, sendo então passíveis de que outro usuário os proteja e os altere. A

solução flexivel é de considerar tanto blocos de proteção como blocos de transações como sendo definições autonomas.

Porém algumas consequencias imediatas podem ser observadas como, por exemplo qual o sentido de liberar pontos no interior de uma transação?

Para esta pergunta temos as seguintes alternativas:

1. se as construções nestes pontos foram modificadas, não faria nenhum sentido a liberação desses pontos já que o sistema terá que mante-los protegidos contra a ação de outros usuários, na expectativa do término da transação , para uma possível reconstrução regressiva.
2. se nenhuma construção em qualquer desses pontos sofrem modificações é claro que as construções nestes pontos estão em estado consistente, podendo então estes pontos serem liberados a outros usuários.

As alternativas (1) e (2), sugerem que além do usuário especificar os pontos a serem protegidos, que ele indique também o tipo de proteção, ou seja para leitura ou para alteração. Assim pontos protegidos para operações de leitura podem ser liberados no decorrer de uma transação e pontos mantidos para modificação, não, pois não teria sentido.

Como extensão destas propriedades anteriores, temos as seguintes regras com relação a encaixamento entre blocos de transações e blocos de proteção, mostradas na figu

ras 5.1 e 5.2.

Utilizaremos como convenção o seguinte:

- - - → define um bloco de proteção para leitura

. . . → define um bloco de proteção para alteração

——— → define um bloco de transação.

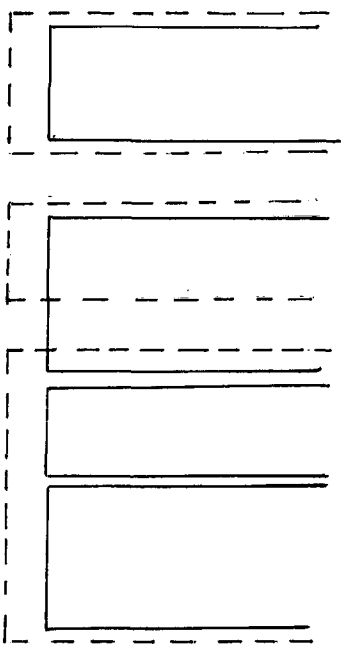


FIGURA 5.1 -ENCAIXAMENTOS POSSÍVEIS ENTRE BLOCOS DE PROTEÇÃO PARA LEITURA E BLOCOS DE TRANSAÇÃO

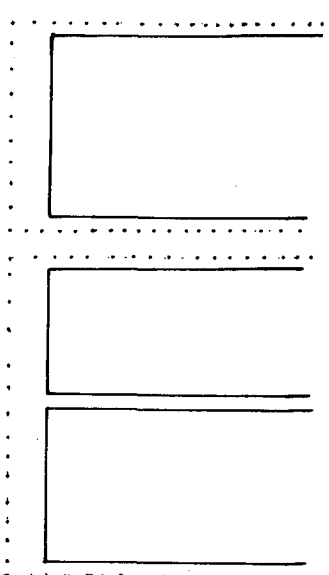


FIGURA 5.2 - ENCAIXAMENTOS POSSÍVEIS ENTRE BLOCOS DE PROTEÇÃO PARA ALTERAÇÃO DE BLOCOS DE TRANSAÇÃO

Uma regra exigida pelo sistema é de que uma transação não pode realizar ações seja de leitura ou alteração em pontos não protegidos. O que significa que não podemos ter a situação apresentada na figura 5.3.

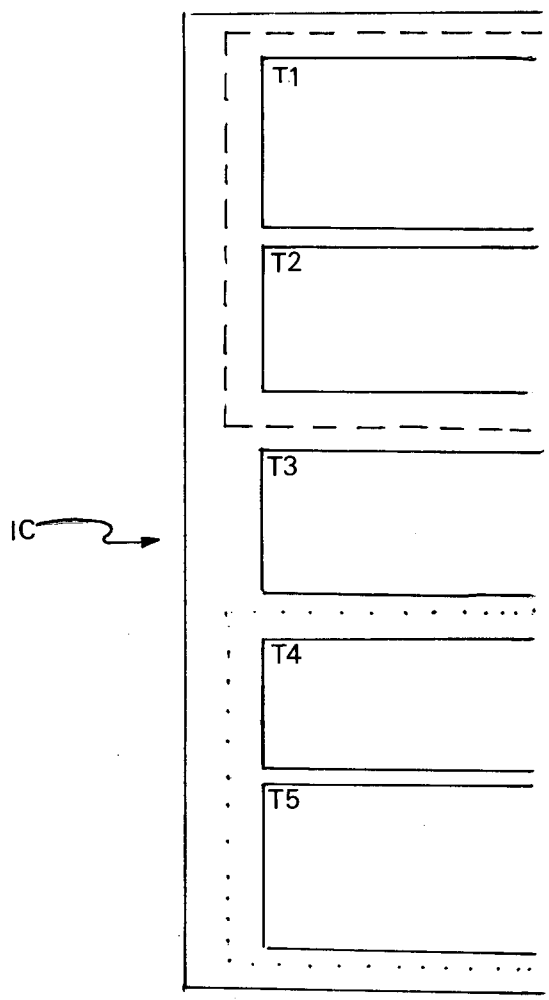


FIGURA 5.3. - EXEMPLO DE UM PROCEDIMENTO COMPLETO (IC), ONDE SE ENCONTRA UMA TRANSAÇÃO SEM ESTAR RELACIONADA COM UM BLOCO DE PROTEÇÃO.

O bloco de transação designado por T3, não possui nenhum bloco de proteção associado.

5.6. Possibilidades de Proteção e Consequências

Sabemos então que através de um bloco de proteção o usuário fornece ao sistema qual o conjunto de pontos

do seu interesse que ele deseja que o sistema realize uma proteção e até quando esta proteção deve ser mantida pelo sistema. Como vimos anteriormente o usuário tem a possibilidade de sobre os pontos identificados para uma proteção de fornecer ao sistema o tipo de proteção ou seja para leitura ou para alteração, que têm a seguinte forma:

- Bloco de proteção para leitura nos endereços

  $<end>_1,\ldots, <end>_n$

  ESTABELECER PROTEÇÃO <id-proteção>

  SOBRE$\{<end>_1,\ldots, <end>_n\}$ PARA LER

  $<instrução>_1$

  ⋮

  $<instrução>_m$

  ABANDONAR PROTEÇÃO <id-proteção>

- Bloco de proteção para alteração nos endereços

  $<end>_1,\ldots, <end>_n$

  ESTABELECER PROTEÇÃO <id-proteção>

  SOBRE$\{<end>_1,\ldots,<end>_n\}$ PARA ALTERAR

  $<instrução>_1$

  ⋮

  $<instrução>_m$

  ABANDONAR PROTEÇÃO <id-proteção>

Ao encontrar um pedido de estabelecimento de proteção o sistema tenta atendê-lo de forma a tornar os pontos determinados pelos endereços neste pedido em pontos protegidos. No entanto somente a proteção dos pontos especificado pelos endereços de ponto da instrução de ESTABELECER PRO

TEÇÃO não é suficiente para que o sistema possa garantir que se o usuário realizar operações somente nestes pontos a consistência será mantida em alguns casos. Para demonstrar este aspecto considere o exemplo 5.2

EXEMPLO 5.2: Suponhamos que tenhamos a definição dos seguintes tipos de tupla e tabela relacional.

```
TUPTIP: TIPO DE TUPLA TAL QUE
          COMPOS
           A  → TIPA,
           B  → TIPB,
           C  → TIPC
          CONEX
        AJUSTAR C OCOR·A = C OCOR.B
TABTIP: TIPO DE TABELA RELACIONAL TAL QUE
        COMPOS
        TUPTIP
```

e mais suponhamos que este tipo de tabela relacional faça parte de um tipo de arquivo relacional ARQTIP, que ocorra no acervo sob nome ARQ. Consideremos agora então dois usuários U1 e U2 e os seguintes blocos de proteção, definidos por eles.

```
U1: ESTABELECER PROTEÇÃO COL-B
    SOBRE {ACERVO.ARQ.TR.TUP.B}PARA ALTERAR
    SUBSTITUIR EM ACERVO.ARQ.TR.(C A > 2).B
                POR 4
        '
        '
        '
    < instrução>n
    ABANDONAR PROTEÇÃO COL-B
```

U2: ESTABELECER PROTEÇÃO COL-A
SOBRE{ACERVO.ARQ.TR.TUP.A} PARA ALTERAR
SUBSTITUIR EM ACERVO.ARQ.TR(C A > 2).A
POR 5
:
:
:
<instrução $>_n$
ABANDONAR PROTEÇÃO COL-A

É claro que os pedidos de proteção dos usuários U1 e U2, não estão conflitante como veremos mais tarde. Intuitivamente isto é fácil de ser verificado dado que U1, pede uma proteção sobre a coluna A da tabela e U2 pede uma proteção sobre a coluna B dessa mesma tabela. Então o sistema autorizaria o prosseguimento das operações de U1 e U2. Se a operação de substituição emitida por U2, fosse realizada primeiro que a operação de substituição emitida por U1 então U1, encontraria um estado inconsistente de acervo, pelo fato de que a consistencia lateral ou conexão definida para o tipo de tupla TIPTUP exige que os componentes sob nome A e B sejam iguais, e devido a operação realizada por U2 teríamos que, as construções em alguns pontos da coluna A, não seriam iguais a construções nos pontos da coluna B, para o mesmo ponto de tupla. E o mais importante é que quando da realização da operação de substituição emitida por U1, o efeito desta realização seria o de desfazer a operação de substituição emitida por U2, isto porque existe uma conexão ajustável, que define que a qualquer alteração realizada em construções nos pontos da coluna B, estas alterações ficam

ajustável em construções nos pontos da coluna A, para o mesmo ponto de tupla. Em permitindo que o apresentado por este exemplo pudesse ocorrer a não interferência entre dois usuário seria então também permitida. Assim para evitar este tipo de problema o sistema automaticamente inclui os pontos sob conexões ajustáveis com os pontos determinados pelos endereços de pontos de uma instrução de ESTABELECER PROTEÇÃO como pontos protegidos associados a esta instrução.

Para simplicidade de referência, chamaremos os pontos determinados pelos endereços de pontos de uma instrução de ESTABELECER PROTEÇÃO de pontos reclamados.

Trataremos a seguir dos motivos pelos quais o sistema não incluirá automaticamente os pontos que possuem conexões não ajustáveis com os pontos reclamados.

- O efeito apresentado no exemplo 5.2, que foi a interferencia entre os usuários U1 e U2, não ocorreria se não considerassemos os pontos sob conexão não ajustáveis com os pontos reclamados, porque o sistema não realizaria um ajuste automático.
- Se o usuário tem conhecimento dos pontos que estão conectados aos pontos reclamados referente ao seu pedido de proteção, então ele próprio tem condições de fornecer ao sistema que estabeleça proteção para estes pontos conectados para que nesses pontos não hajam outros usuários alterando-os.

- Se o usuário por outro lado não tem conhecimento dos pontos conectados com os seus pontos reclamados, então de nada resolveria o sistema realizar automaticamente uma proteção sobre estes pontos conectados, já que o próprio usuário desconhece a existência dessas conexões , não podendo assim, garantir a manutenção desta consistência.

Porém dado um pedido explícito do usuário o sistema automaticamente proteje os pontos conectados com pontos reclamados, seja esta conexão ajustável ou não, sobre este automatismo falaremos após o exemplo 5.3, que endossa os motivos apresentados anteriormente.

EXEMPLO 5.3: Por exemplo, suponhamos que tenhamos a seguinte conexão definida para um tipo de tupla J.

```
J : TIPO DE TUPLA TAL QUE
    COMPOS
      A → TIPA,
      B → TIPB,
      C → TIPC
    CONEX
  C OCOR.A = C OCOR.B
```

e um tipo de tabela relacional TA dado por:

```
TAB: TIPO DE TAREL TAL QUE
     COMPOS
       J
```

Agora suponhamos que este tipo de tabela relacional faça par

te de um tipo de arquivo TIPATALL que apareça no acervo sob nome ARQ. Então ao realizar um pedido de proteção dado por

ESTABELECER PROTEÇÃO COL-A SOBRE
{ACERVO.ARQ.TR.TUP.A} PARA ALTERAR

e se o usuário alterar um elemento da coluna A da tabela, é claro que para que o acervo permaneça consistente ele deve também alterar o valor correspondente na coluna B na tabela . No caso de terminar seu procedimento sem realizar a alteração correspondente na coluna B, para os pontos da coluna A,alterados seu procedimento será considerado incorreto, pois transformou um Acervo inicialmente consistente em um Acervo inconsistente. Se então o usuário tem o conhecimento de que ao alterar na coluna A, deve também alterar na coluna B então ele estabeleceria a seguinte proteção:

ESTABELECER PROTEÇÃO COL-AB SOBRE
{ACERVO.ARQ.TR.TUP.A,ACERVO.ARQ.TR.TUP.B}PARA
ALTERAR

O automatismo que o usuário pode se referir quando de um pedido de proteção que falamos anteriormente quando dizemos que as conexões serão protegidas dado um pedido explícito do usuário, é realizado da seguinte forma:

ESTABELECER PROTEÇÃO <id-proteção>
{<end ponto>,...} PARA { LER / ALTERAR }
COM CONEXÃO

Pela referência de COM CONEXÃO o sistema calcula automaticamente todos os pontos conectados aos pontos

reclamados para qualquer tipo de conexão ajustável ou não. Se o usuário não se referir ao termo COM CONEXÃO, como já dissemos anteriormente os pontos de serem protegidos serão somente os pontos que possuem conexões ajustáveis com os pontos reclamados.

Uma observação importante é que os pontos incluídos para serem protegidos devido as conexões tem associado a eles o mesmo tipo de proteção (ler, alterar), designados para os pontos reclamados aos quais estes pontos estão conectados.

CAPÍTULO VI

DETERMINAÇÃO DAS ÁREAS INDICADA E RECLAMADA

6.1. Área

Definimos como Área um conjunto de pontos.

6.2. Área de Partida

Uma área de partida é definida como o conjunto de pontos obtidos pela avaliação de um endereço de ponto num dado instante em um acervo.

EXEMPLO 6.1.: Dado o endereço de ponto: ACERVO.ARQ TR.TUP, avaliado no acervo mostrado na figura 6.1, temos como área de partida associada a este endereço os pontos assinaldos com x .

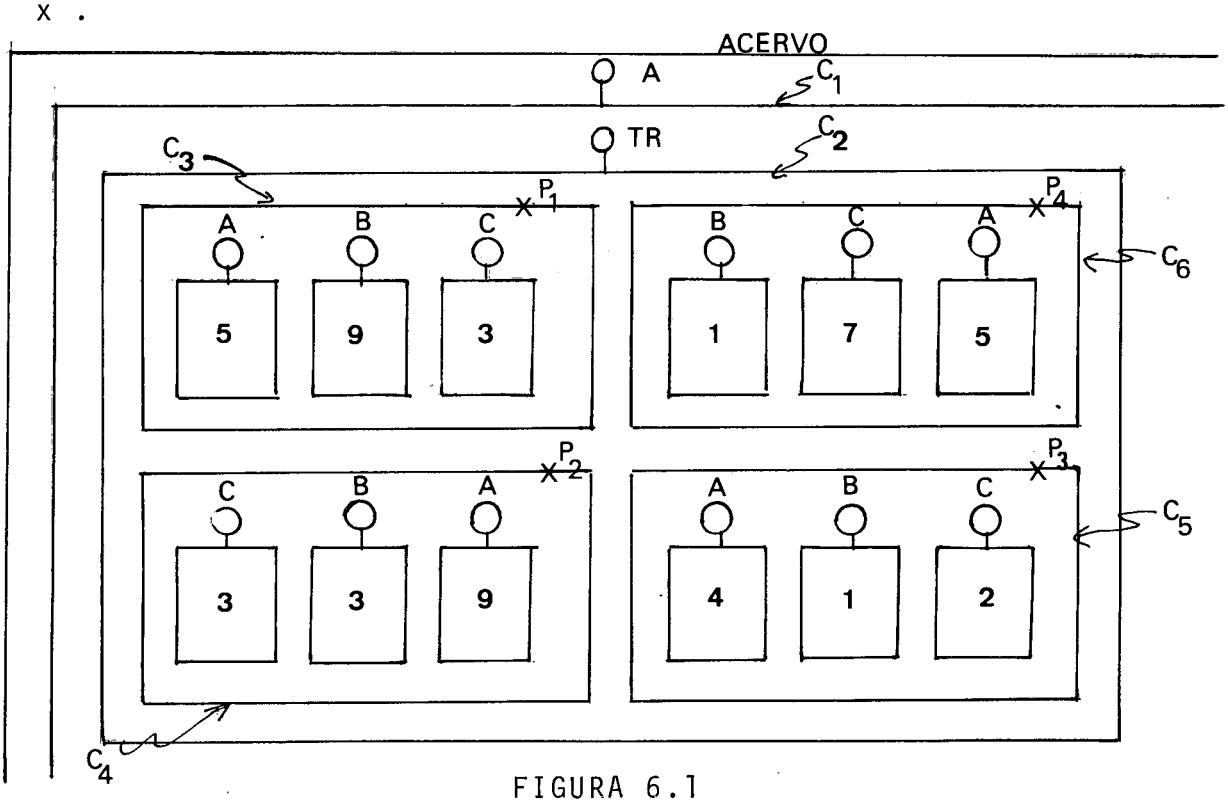


FIGURA 6.1

As expressões que definem estes pontos são:

$$P_1 = (-,ACERVO)(A,C_1)(TR,C_2)(-,C_3)$$

$$P_2 = (-,ACERVO)(A,C_1)(TR,C_2)(-,C_4)$$

$$P_3 = (-,ACERVO)(A,C_1)(TR,C_2)(-,C_5)$$

$$P_4 = (-,ACERVO)(A,C_1)(TR,C_2)(-,C_6)$$

Assim a área de partida , associada ao endereço de ponto ACERVO.ARQ.TR.TUP é

$$APART = \{P_1,P_2,P_3,P_4\} \text{ ou } APART_{end} = \{P_1,P_2,P_3,P_4\}$$

EXEMPLO 6.2.: Dado o endereço de ponto ACERVO.ARQ.TR.TUP.A, avaliado no acervo da figura 6.1, temos como área de partida os seguintes pontos:

$$P'_1 = (-,ACERVO)(A,C_1)(TR,C_2)(-,C_3)(A, 5)$$

$$P'_2 = (-,ACERVO)(A,C_1)(TR,C_2)(-,C_4)(A, 9)$$

$$P'_3 = (-,ACERVO)(A,C_1)(TR,C_2)(-,C_5)(A, 4)$$

$$P'_4 = (-,ACERVO)(A,C_1)(TR,C_2)(-,C_6)(A, 5)$$

$$APART_{end} = \{P'_1, P'_2, P'_3, P'_4\}$$

Podemos verificar que as áreas de partidas para os endereços dos exemplo 6.1 e 6.2, não possuem pontos em comum, isto é, chamando de end_1 o endereço fornecido no exemplo 6.1 e end_2 o endereço fornecido no exemplo 6.2 temos que:

$$APART_{end_1} \cap APART_{end_2} = \emptyset$$

embora os pontos da área de partida para o exemplo 6.1 sejam superiores aos pontos da área de partida para o exemplo 6.2.

6.3. Área Abrangida

Dado uma área de partida $APART_{end} = \{P_1, \ldots, P_n\}$, definimos como área abrangida associada a área de partida AP, o conjunto de pontos formados pela união aos cones de pontos cujos vértices são os pontos $P_i \in APART_{end}$. Podemos dizer também que a área abrangida está associada ao endereço end

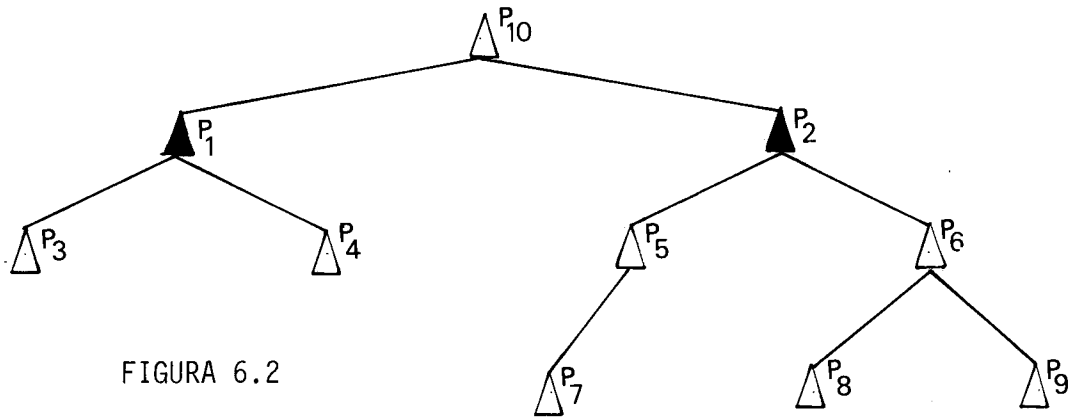


FIGURA 6.2

EXEMPLO 6.3.: Se tivermos como área de partida os pontos P_1 e P_2 na figura 6.2 então a área abrangida associada a esta área de partida seria dada por:

$$AAB_{end} = \{P_3, P_4, P_5, P_6, P_7, P_8, P_9\}$$

EXEMPLO 6.4.: Dado o endereço de ponto ACERVO.ARQ.TL.LIG, avaliado no acervo da figura 6.3.

Teríamos como área de partida os pontos de todas as ligações no instante da avaliação do endereço de ponto dado por ACERVO.ARQL.TL.LIG como área abrangida teríamos:

- todos os pontos de ligante
- todos os pontos de tabela ligada
- todos os pontos de tupla para todas as tabelas ligadas
- todos os pontos de átomos para todos os pontos de ligante

- todos os pontos de átomos, para todos os pontos de tupla das tabelas ligadas.

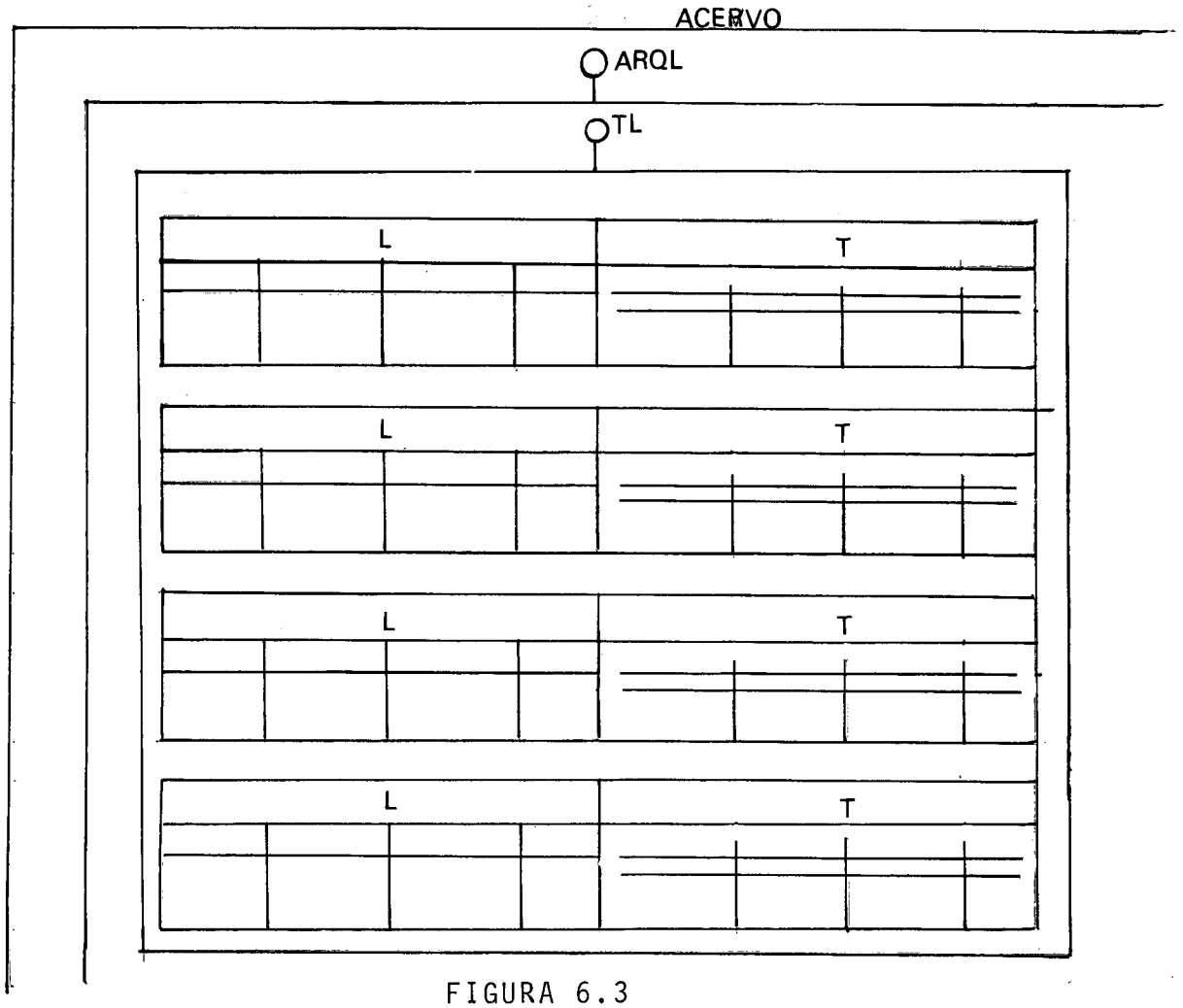


FIGURA 6.3

Podemos visualizar através da figura 6.4 que mostra a árvore de pontos respectiva a figura 6.3. Os pontos a serem considerados estão envoltos por linhas tracejadas, e os pontos da área de partida estão hachuriados.

6.4. Área indicada

Definimos como área indicada associada a um endereço de ponto, o conjunto de pontos definido pela união da área de partida associada a este endereço de ponto com a

área abrangida associada a esta área de partida:

$$AIND_{end} = APART_{end} \cup AAB_{end}$$

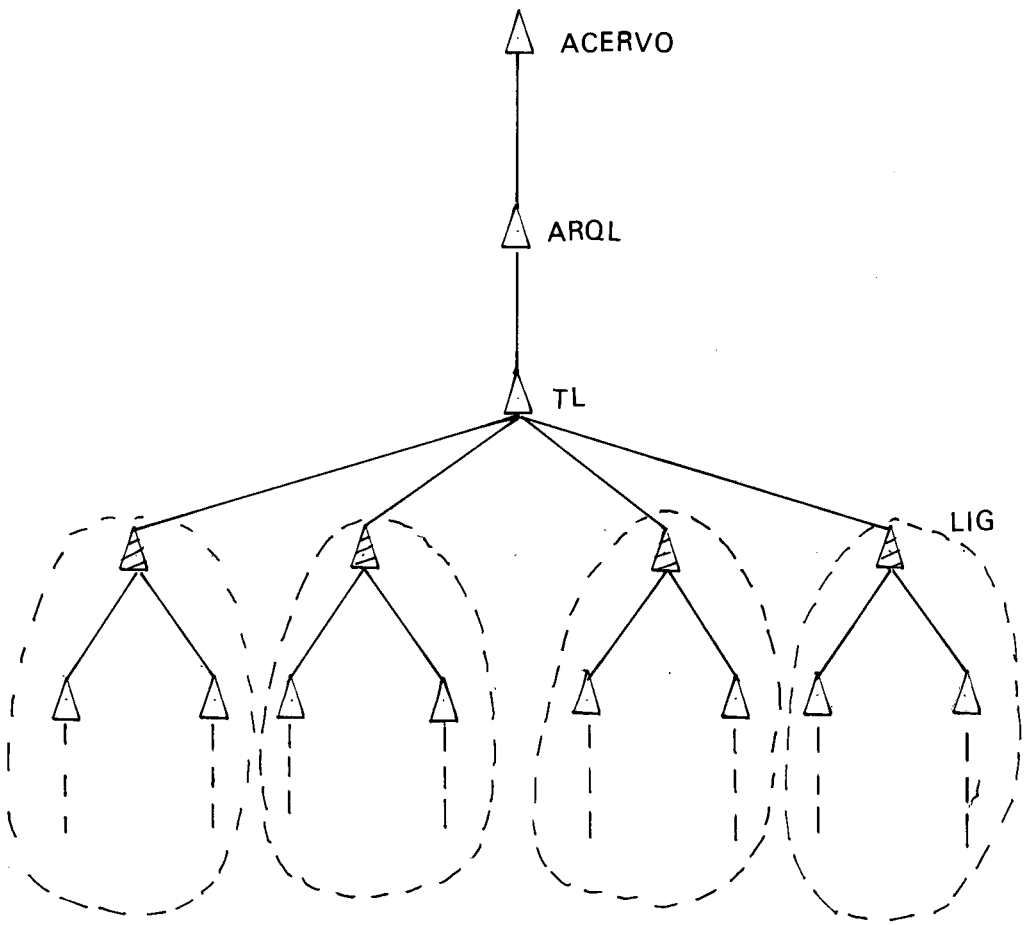


FIGURA 6.4

6.5. Área Conectada

A área conectada associada a um endereço end é formada pelo conjunto de pontos, que possuem conexão direta ou indireta com pontos da área indica relativa ao endereço end e mais os cones de pontos para cada um desses pontos, ou

seja os pontos que possuem conexões direta ou indereta com pontos da área indicada.

Podemos então através de um exemplo compreender mais claramente como é formada a área conectada.

EXEMPLO 6.5.: Considere a árvore de pontos indicada na figura 6.5.

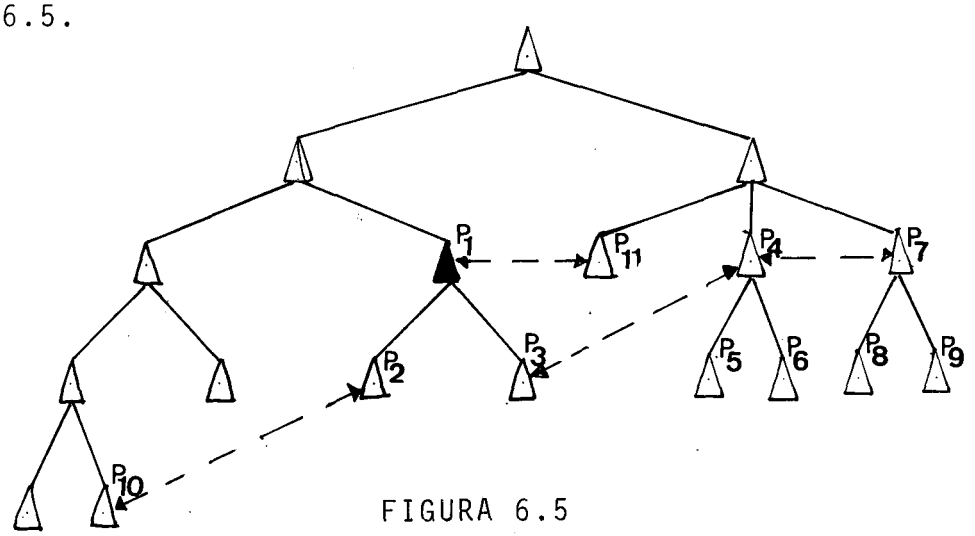


FIGURA 6.5

Suponhamos então que um dado endereço de ponto enderece somente o ponto P_1 indicado na figura 6.5, teríamos então como área de partida

$$APART_{end} = \{P_1\}$$

calculando a área abrangida temos então os pontos P_2 e P_3

$$AAB_{end} = \{P_2, P_3\}$$

cálculo da área conectada, observando que as linhas tracejadas na figura 6.5 indicam conexões entre os pontos.

Vemos que o ponto P_{11} está conectado ao ponto P_1 e os pontos P_{10} e P_4 estão conectados respectivamente aos pontos P_2 e P_3 da área abrangida. Vemos também que o ponto

P_4 está conectado ao ponto P_7, então dizemos que P_3 tem conexão indireta com P_7.

Assim como conjunto de pontos que possuem conexão direta ou indireta com pontos da área indicada temos:

$$C = \{P_4, P_7, P_{10}, P_{11}\}$$

Calculando-se então os cones de pontos para cada ponto do conjunto C, temos como área conectada associada ao endereço end

$$ACOND_{end} = \{P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}\}$$

6.6. Área Reclamada

Definimos como área reclamada associada a uma instrução de ESTABELECER PROTEÇÃO o seguinte conjunto de pontos

ESTABELECER PROTEÇÃO <id-proteção>
SOBRE {<end>$_1$,...,<end>$_n$} PARA {LER / ALTERAR}
[COM CONEXÃO]

$$AREL = AIND_{<end>_1} \cup AIND_{<end>_2} \text{---} \cup AIND_{<end>_n}$$
$$\cup ACON_{<end>_1} \cup ACON_{<end>_2} \text{- - -} \cup ACON_{<end>_n}$$

onde as áreas conectadas incluem todos os pontos sob conexão no caso de especificação de COM CONEXÃO e em caso contrário incluem somente pontos sob conexões ajustáveis.

CAPÍTULO VII

REGRA DE CORRESPONDÊNCIA

Como vimos no capítulo III, o conceito de ponto é um conceito estático, porque dada uma modificação na construção de referência a sequência definidora de um ponto antes da modificação não tem mais validade, isto é, não identifica mais pontos na construção de referência.Com a finalidade da aplicação do conceito de ponto é necessário então que hajam regras que permitam associar, pontos em acervos antes e depois de modificações.

EXEMPLO 7.1.: Dados duas construções como mostrado na figura 7.1

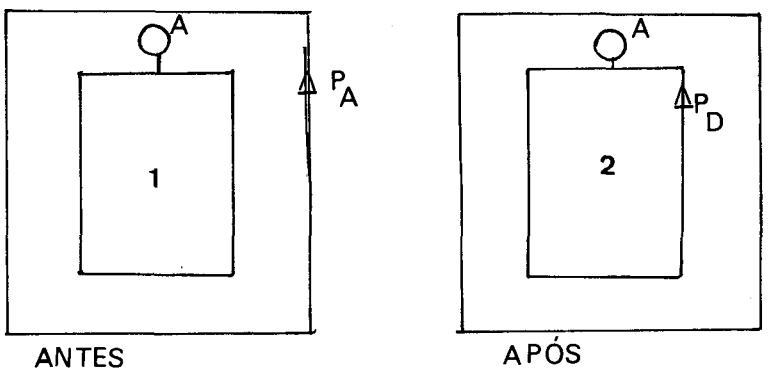


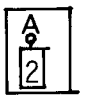
FIGURA 7.1

O ponto P_A que identifica a construção sob nome A, antes da modificação é dado por:

$$P_A = (-, \boxed{\overset{A}{\boxed{1}}}) \ (A, \boxed{1})$$

O ponto P_D que identifica a construção sob no

me A, após a modificação é dado por:

P_D = (-, )(A, )

Queremos formular um conjunto de regras que associe os pontos P_A e P_D, pontos antes e após modificações com a finalidade de tornar dinâmica uma aplicação do conceito de ponto.

Esta regra evidentemente deverá abranger a associação ou correspondência entre pontos para qualquer tipo de acervo permitido pela interface. Esta regra servirá posteriormente para o tratamento de manutenção de áreas protegidas, que será apresentado no capítulo VIII.

Antes da definição das regras de correspondência, falaremos sobre as instruções introduzidas no capítulo II, que serviram para a definição das regras pretendidas.

7.1. Operação de Substituição (Substituir Construção)

A operação de substituição tem a seguinte apresentação sintática:

SUBSTITUIR [<termo de tipificação>] EM <end ponto>
POR <obter construção>

EXEMPLO: Substituir todos os átomos sob nome A, de todas as tuplas de uma tabela relacional em um arquivo relacional sob nome ARQ, pela construção 3.

SUBSTITUIR ÁTOMO EM PONTOS ACERVO.ARQ.TR.TUP.A
POR 3

É claro que, na operação de substituição é

necessário que a construção obtida no <obter construção> seja admitida no ponto endereçado pelo <end ponto>.

7.2. Operação de Modificação

A operação de modificação tem a seguinte apresentação sintática.

MODIFICAR [<termo de tipificação>] EM <end ponto>
COM <obter nominação>

EXEMPLO 7.2.: Modificar os átomos sob nome A para 3, sob nome B para 2, e sob nome C para 4 das tuplas da tabela relacional, de um arquivo sob nome ARQ, somente para as tuplas cujos átomos sob nome D sejam maiores do que 10. As tuplas desta tabela tem como componentes átomos sob nomes A, B, C e D.

```
MODIFICAR TUPLA EM PONTOS ACERVO.ARQ.TR.(C D > 10)
                COM OBTER TUPLA COMPOR
                    (1( A:=3
                        B:=2
                        C:=4
                    )1)
```

Pela instrução de modificar, o sistema faz uma sobreposição da nominação definida em <obter nominação>, com uma "subnominação" das construções nos pontos endereçados.

7.3. Operação Substituir Nome

Tem a seguinte apresentação sintática:

SUBSTITUIR NOME EM <end ponto> POR <obter nome>

EXEMPLO 7.3.: Substituir o nome de um arquivo relacional de nome ARQ, para o nome ARQNOVO.

SUBSTITUIR NOME EM PONTO ACERVO. ARQ POR ARQNOVO

7.4. Operação de Inserção

Apresentação sintática:

INSERIR <obter construção> EM <end ponto>

EXEMPLO 7.4.: Inserir uma tupla em uma tabela relacional de um arquivo relacional sob nome ARQ.

```
INSERIR  OBTER TUPLA
         COMPOR ( A:= 5
                  B:= 10
                  C:= 20)
EM PONTO ACERVO.ARQ.TR
```

7.5. Operação de Retirar

Apresentação sintática:

RETIRAR [<termo de tipificação>] DE <end ponto>

Retira as construções nos pontos endereçados.

7.6. As cinco regras de correspondência

Podemos então definir um conjunto de regras para correspondência de pontos. No decorrer da elaboração deste trabalho, foi tentada a elaboração de uma regra de corres-

pondência estática entre pontos, não tendo alcançado resultados que satisfizessem aos requisitos básicos necessários devido as operações permitidas. Falamos em regra estática, que significa a comparação entre duas construções e a associação por um processo qualquer estático, como por exemplo, "as construções que possuem o mesmo nome estão em correspondência", e outras formas mais, para indicar a correspondência entre pontos, porém como resultados forneceram somente regras de "afinidades" entre construções.

Antes da apresentação das regras de correspondência introduziremos as seguintes convenções:

(1) um ponto será representado por

$$P_{ji} = (a_1,c_1)(a_2,c_2)\text{---}(a_i,c_i)$$

onde j serve para a distinção entre dois pontos do mesmo acervo e i indica até que par ("profundidade") vai a sequência do ponto em questão.

(2) A_A e A_N representarão os acervos antigo e novo respectivamente, isto é, o acervo antigo (antes da alteração) e novo (após a alteração)

(3) P_{ji} representam pontos de A_A e P'_{ji} representam pontos de A_N

REGRA 1 : REGRADA DA MODIFICAÇÃO

Dadas as ocorrências de um tipo de acervo A_A e A_N , dizemos que:

$P_{1n} = (-,C_1)...(a_n,c_n)$, ponto de A_A, $c_1 = A_A$

e

$P'_{1n} = (-,c'_1)...(a'_n,c'_n)$, ponto de A_N, $c'_1 = A_N$

estão em correspondência se:

P'_{1n} foi obtido por uma operação abaixo realizada em P_{1n}

- SUBSTITUIR nome
- MODIFICAR
- SUBSTITUIR construção

Dizemos que se P_{1n} está em correspondência com P'_{1n}, então P_{1n} e P'_{1n} pertencem ao mesmo lugar.

Esta regra sifnifica que, quando da realização por parte do usuário de uma das operações citadas na regra-1, o sistema associa o ponto antes da operação com o ponto após a operação.

EXEMPLO 7.5 : Efetuar uma modificação nas tuplas cujos componentes sob nome B são maiores do que 4.

A operação definida como:

```
MODIFICAR TUPLA EM PONTOS ACERVO.ARQ.TR.(C B > 4)
 POR OBTER TUPLA
 COMPOR(2( A:=1
           C:=4 )2)
```

Na figura 7.2 apresentamos as ocorrências A_A, ou seja acervo antigo, e o acervo novo A_N, obtido após a execução da operação de modificação. As linhas tracejadas indicam os pontos correspondentes aos acervos A_A e A_N provenien-

te da aplicação da regra 1.

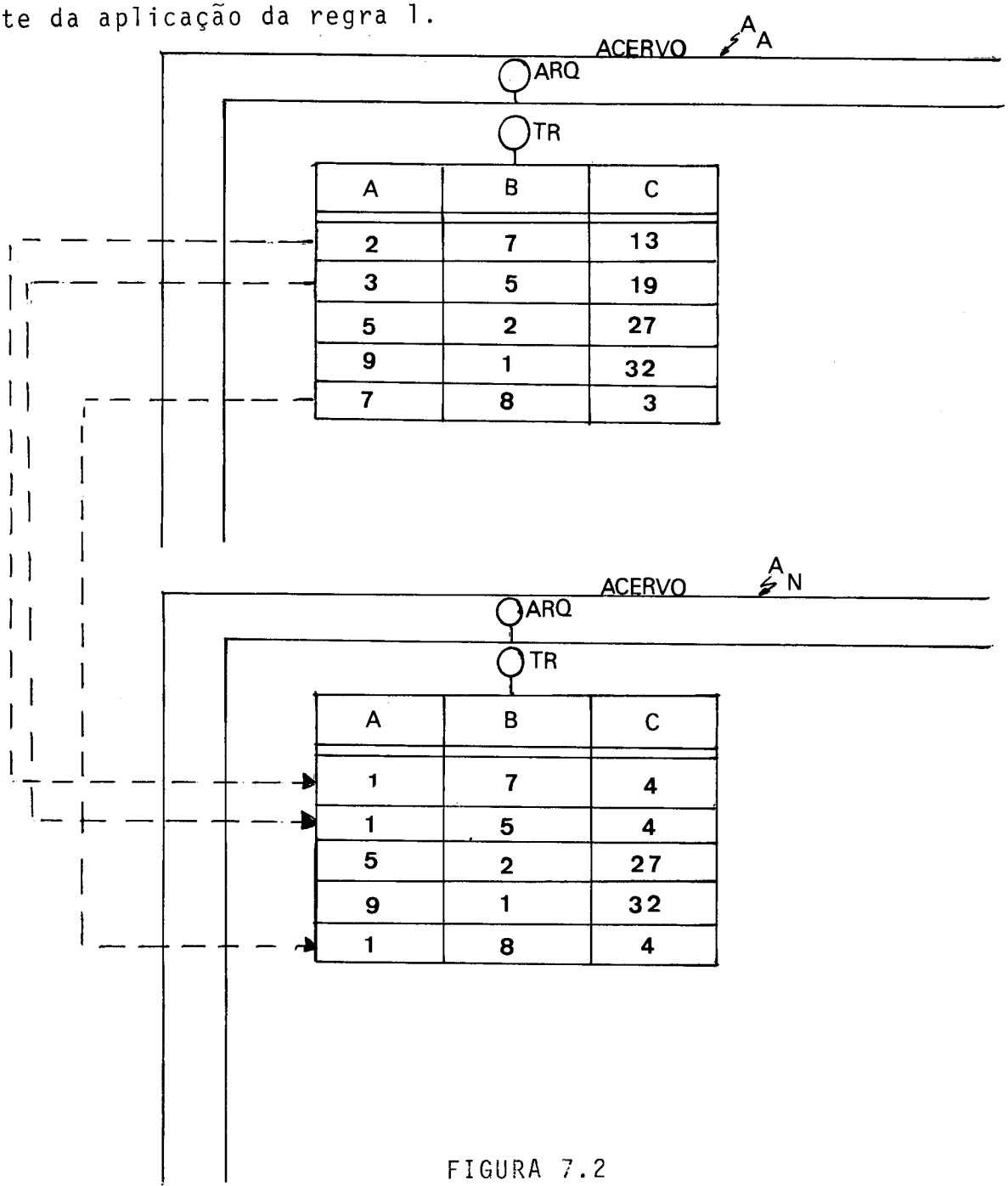


FIGURA 7.2

Em termos de árvore de pontos teríamos o mostrado na figura 7.3 com a correspondência indicada por linhas tracejadas.

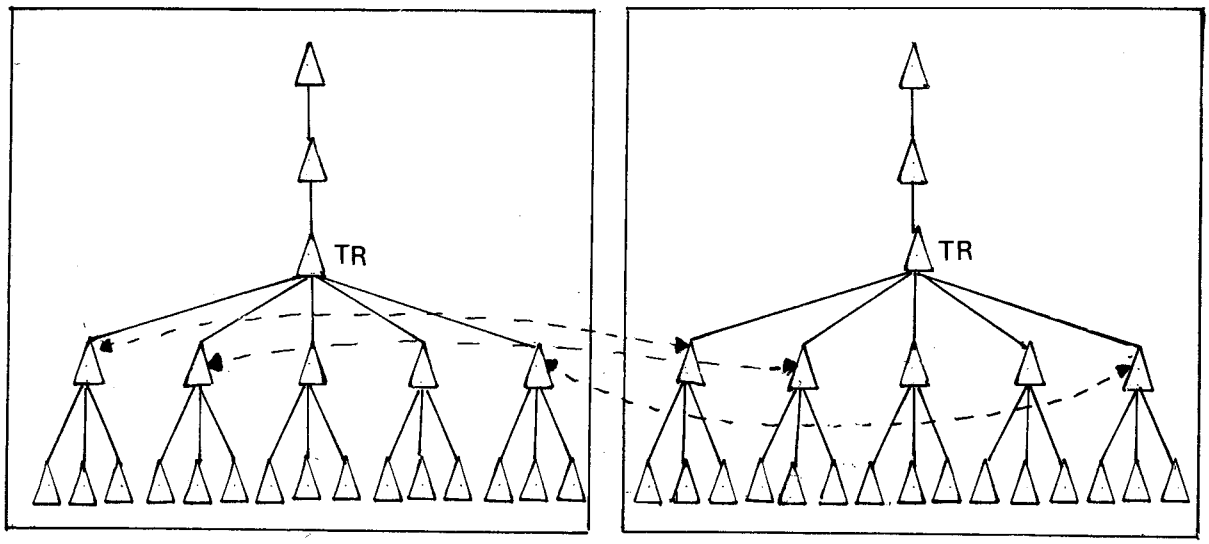


FIGURA 7.3

Podemos observar, que a regra-1, não diz nada com respeito aos pontos inferiores ou superiores do ponto modificado. Com respeito aos pontos superiores a pontos modificado introduziremos agora a regra de número dois, que definirá uma correspondência entre pontos superiores ao ponto modificado. Esta regra serve para evitar que aconteça o seguinte caso mostrado na figura 7.4.

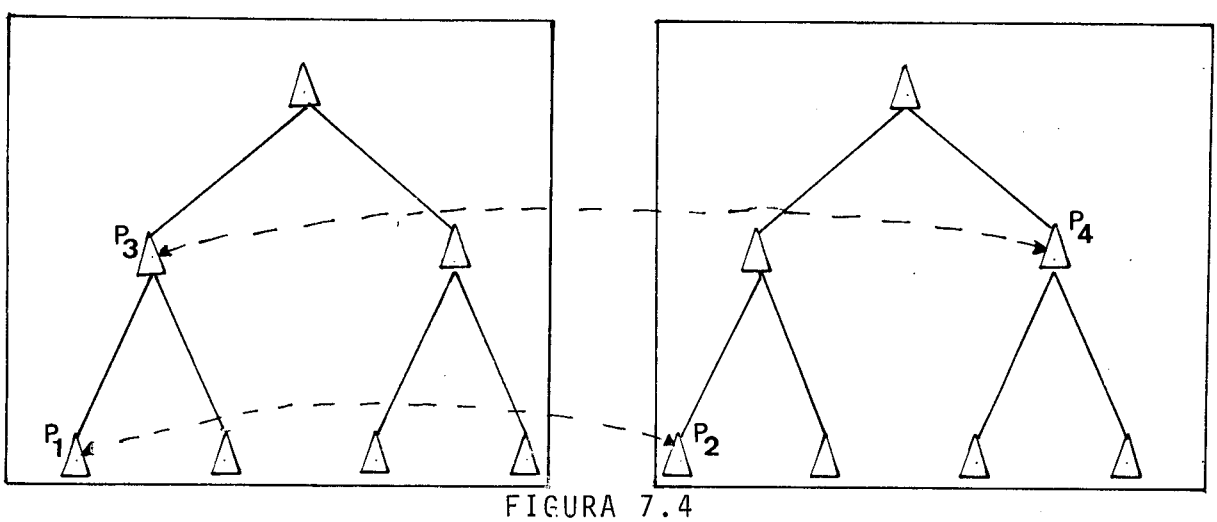


FIGURA 7.4

De haver uma correspondência entre um ponto inferior P_{1i}, com um outro ponto inferior P'_{1i}, e que os pontos superiores a P_{1i} e a P'_{1i} respectivamente não tenham cor-

respondência. Como exemplo, podemos dizer que seria equivalente a substituir um átomo em uma tupla (T1) e este átomo estar em correspondência com um átomo de uma tupla T2 no novo acervo e a tupla T1 estar em correspondência com uma tupla T3 no novo acervo. E mais ainda, da possibilidade de não correspondência entre um acervo antes da alteração e após a alteração.

REGRA-2 : Regra do caminho

Dadas duas ocorrências A_A e A_N de um tipo de acervo:

se $P_{1n} = (-,c_1)(a_2,c_2)\ldots(a_n,c_n)$, ponto de A_A

e

$P'_{1n} = (-,c'_1)(a'_2,c'_2)\ldots(a'_n,c'_n)$, ponto de A_N

onde $c_1 = A_A$ e $c'_1 = A_A$

são pontos correspondentes pela regra 1 então os pontos

$P_{1i} = (-,c_1)(a_2,c_2)\ldots(a_i,c_i)$

e

$P'_{1i} = (-,c'_1)(a'_2,c'_2)\ldots(a'_i,c'_i)$

são correspondentes, para i = 1,2,..., n - 1

Como consequência da aplicação da regra-2, se um acervo for modificado então os pontos de acervo antes e depois da alteração estão em correspondência, a saber, para i = 1 temos que; os pontos:

$P_{11} = (-,c_1) = (-,A_A)$

$P'_{11} = (-,c'_1) = (-,A_N)$

estão em correspondência. Uma forma simples e que deu o nome a regra é de visualizarmos a aplicação da regra número 2 em árvores de pontos.

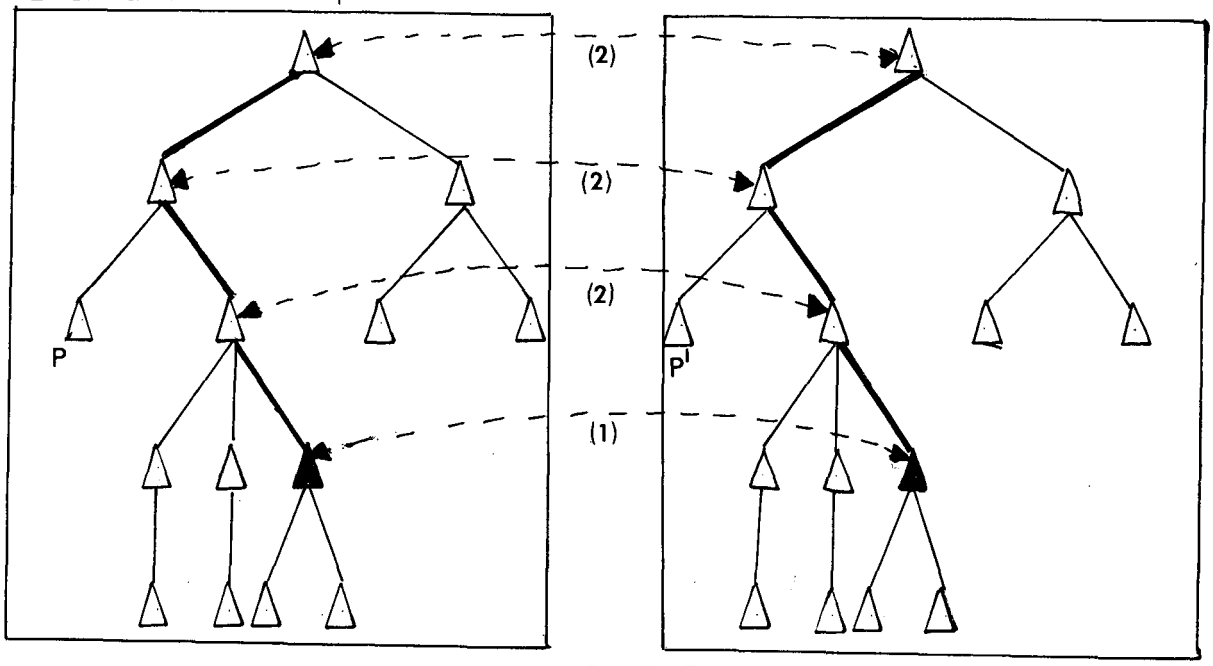


FIGURA 7.5

Os pontos em negritos da figura 7.5 têm correspondência determinada pela regra-1. Tomando-se o caminho destes pontos até o ponto de acervo determina-se a correspondência entre os pontos encontrados neste caminho para os acervos antes e depois da modificação.

As regras 1 e 2, definem correspondência relativamente a pontos modificados, porém nem todas as construções em pontos sofrem modificações como por exemplo, quando efetuada somente operações de leitura. Neste caso as regras 1 e 2 não determinam qualquer tipo de correspondência, como por exemplo para os pontos P e P' indicados na figura 7.5. Com o objetivo de suprir esta necessidade introduzimos a Regra 3.

REGRA 3 : Regra de semelhança

Dados duas ocorrências de um tipo de acervos A_A e A_N. Dizemos que:

$P_{1n} = (-,c_1)(a_2,c_2)\ldots(a_n,c_n)$, ponto de A_A

e

$P'_{1n} = (-,c'_1)(a'_2,c'_2)\ldots(a'_n,c'_n)$ ponto de A_N

com $A_A = c_1$ e $A_N = c'_1$, estão em correspondência ou no mesmo lugar se:

Para $I = \max \{i \mid P_{1i} = (-,c_1)(a_2,c_2)\ldots(a_i,c_i)$

e

$P'_{1i} = (-,c'_1)(a'_2c'_2)\ldots(a'_i,c'_i)$

são correspondentes pelas regras 1 ou 2}

e para todo j com $I + 1 \leq j \leq n$ temos que:

$$(a_j,c_j) = (a'_j,c'_j)$$

EXEMPLO 7.6.: Dada a alteração sobre um acervo, dado por:

SUBSTITUIR ÁTOMO EM PONTOS ACERVO.ARQ.TR.(C B $\geq$ 10).A

POR 4

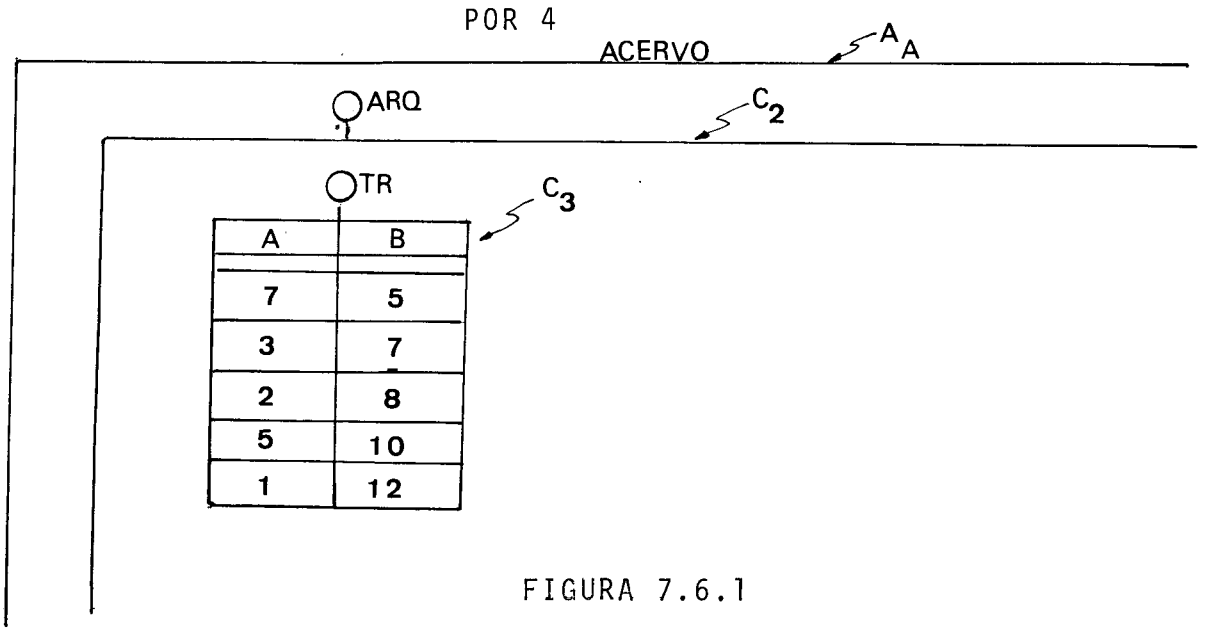


FIGURA 7.6.1

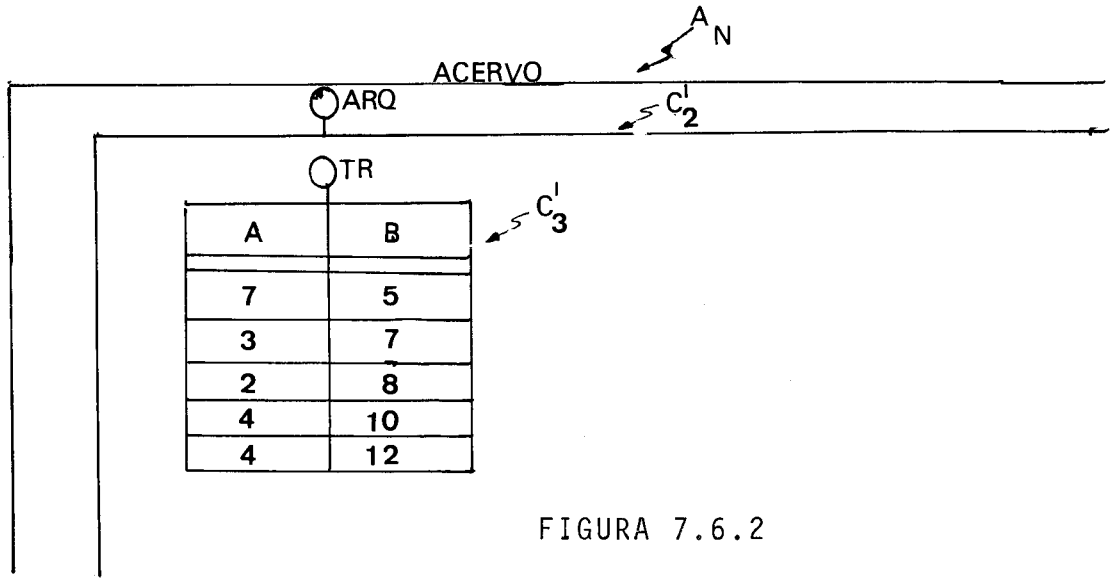


FIGURA 7.6.2

Temos pela aplicação da regra-1 a correspondência entre os pontos:

$P_1 = (-,A_A)(ARQ,C_2)(TR,C_3)(-,$ | A | B | $)(A,$ | 5 | $)$
|---|---|---|---|---|

com

$P'_1 = (-,A_N)(ARQ,C'_2)(TR,C'_3)(-,$ | A | B | $)(A,$ | 4 | $)$
|---|---|---|---|---|

mais

$P_2 = (-,A_A)(ARQ,C_2)(TR,C_3)(-,$ | A | B | $)(A,$ | 1 | $)$
|---|---|---|---|---|

com

$P'_2 = (-,A_N)(ARQ,C'_2)(TR,C'_3)(-,$ | A | B | $)(A,$ | 12 | $)$
|---|---|---|---|---|

Pela aplicação da regra-2, a correspondência entre os pontos

$P_3 = (-,A_A)(ARQ,C_2)(TR,C_3)(-,$	A	B	$)$

com

$P'_3 = (-,A_N)(ARQ,C'_2)(TR,C'_3)(-,$	A	B	$)$

e

$P_4 = (-,A_A)(ARQ,C_2(TR,C_3)(-,$	A	B	$)$

com

$$P'_4 = (-,A_N)(ARQ,C'_2)(TR,C'_3)(-, \begin{array}{|c|c|}\hline A & B \\ \hline 4 & 12 \\ \hline \end{array})$$

e

$$P_5 = (-,A_A)(ARQ,C_2)(TR,C_3)$$

com

$$P'_5 = (-,A_N)(ARQ,C'_2)(TR,C'_3)$$

e

$$P_6 = (-,A_A)(ARQ,C_2)$$

com

$$P'_6 = (-,A_N)(ARQ,C'_2)$$

e

$$P_7 = (-,A_A)$$

com

$$P'_7 = (-,A_N)$$

O restante dos pontos, são correspondidos pela aplicação da regra-3, de semelhança. Mostraremos somente como funciona para um ponto. Queremos por exemplo achar o ponto correspondente ao ponto de átomo P.

$$P = (-,A_A)(ARQ,C_2)(TR,C_3)(-, \begin{array}{|c|c|}\hline A & B \\ \hline 7 & 5 \\ \hline \end{array})(A, \boxed{7})$$

Pois as regras 1 e 2, não definem correspondência para tal ponto. Aplicaremos então a regra de número 3. É claro que desejamos que este ponto P seja correspondente a:

$$P' = (-,A_N)(ARQ,C'_2)(TR,C'_3)(-, \begin{array}{|c|c|}\hline A & B \\ \hline 7 & 5 \\ \hline \end{array})(A, \boxed{7})$$

aplicando então a regra-3, para P e P' temos que:

$I = \max\{i \mid P_{1i} = (-,c_1)(a_2,c_2)\ldots(a_i,c_i)$

corresponde a

$P'_{1i} = (-,c'_1)(a'_2,c'_2)\ldots(a'_i,c'_i)$

pelas regras 1 ou 2}

temos então que I = 3, o que corresponde a

$P_{1I} = (-,A_A)(ARQ,c_2)(TR,c_3)$

$P'_{1I} = (-,A_N)(ARQ,c'_2)(TR,c'_3)$

que são respectivamente os pontos P_5 e P'_5, como mostrado na aplicação da regra de número-2.

Assim para cada $4 \leq j \leq 5$ tem-se que:

$(a_4,c_4) = (a'_4,c'_4)$ e $(a_5,c_5) = (a'_5,c'_5)$

pois $a_4 = a'_4 = -$, $c_4 = c'_4 =$ | A | B | / | 7 | 5 |, $a_5 = a'_5 = A$ e $c_5 = c'_5 =$ 7.

Mostrando então que P e P' são correspondentes pela aplicação da regra-3. De forma análoga cobriríamos todos os pontos mostrados em 7.6.1 e 7.6.2 determinando as correspondências. Uma forma de visualizar a aplicação da regra número 3 em termos de árvores de pontos é a seguinte:

Dadas as árvores de pontos correspondentes a 7.6.1 e 7.6.2

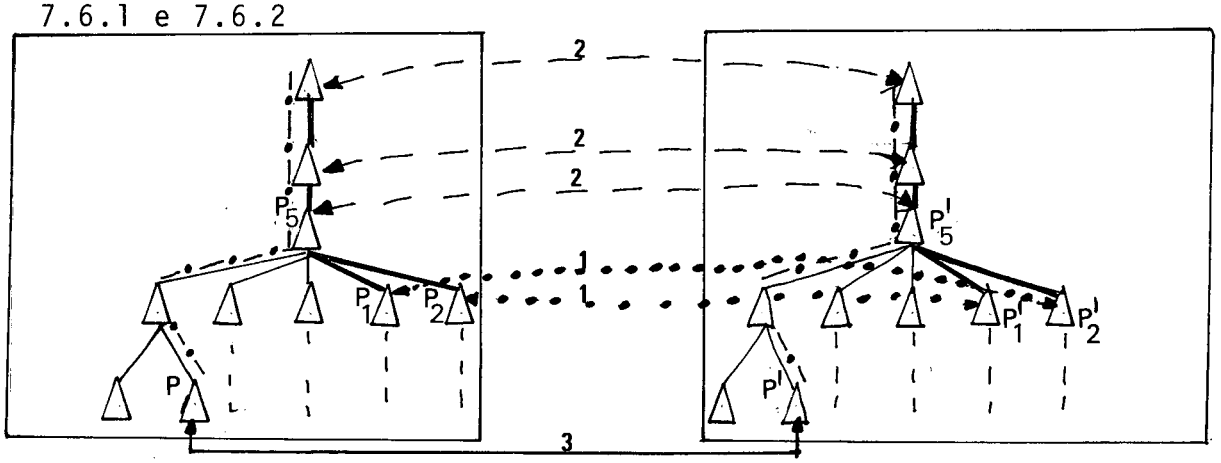


FIGURA 7.7

Por exemplo, queremos verificar se P e P' indicados na figura 7.7 são correspondentes. Tomamos então o caminho de P até o ponto de acervo, como mostrado na figura 7.7, por linhas com pontos e traços. Realizamos o mesmo procedimento com relação a P', um caminho de P' até o ponto de acervo. Os primeiros pontos nestes caminhos que estiverem em correspondência, pela regra 1(representada por linhas pontilhadas) ou pela regra 2(representada por linhas tracejadas), no caso P_5 e P'_5 correspondem aos pontos:

$$P_5 = (-,A_A)(a_2,c_2)---(a_I,c_I)$$

e

$$P'_5 = (-,A_N)(a'_2,c'_2)---(a'_I,c'_I)$$

onde I é definido pela regra-3. Encontrado os pontos P_5 e P'_5, verifica-se a identidade do resto das expressões que definem os pontos P e P',se forem iguais então P corresponde a P', caso contrário não são correspondentes.

REGRA 4: DESAPARECIMENTO DE LUGAR

(1) se P é um ponto de A_A, que foi referenciado por uma operação de RETIRAR, então não existe P', ponto de A_N correspondente a P.

(2) O ponto imediatamente superior ao ponto P no acervo A_A, será considerado como ponto que sofreu uma operação de alteração, podendo asssim para este ponto aplicar-se as regras de 1 a 3.

A motivação intuitiva para a existência desta regra reside no aspecto de que,se por exemplo um usuário retira uma tupla de uma tabela relacional evidentemente o sistema não poderia manter esta tupla protegida dado que esta tupla já não faz mais parte do acervo.

Exemplo 7.7.: Considere a figura 7.8 e a seguinte operação

RETIRAR TUPLA DE PONTOS ACERVO.ARQ.TR.(C A = 5)

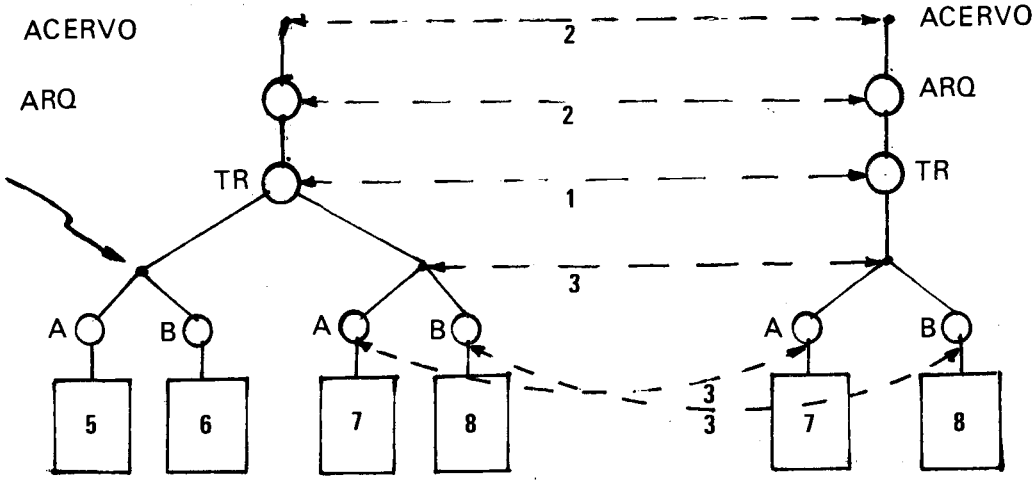


FIGURA 7.8

Devido a aplicação da regra de número 4 pelo item (1), observamos que o ponto de tupla indicado na figura 7.8 por uma seta não tem correspondente no acervo novo. Pela aplicação do item (2), desta mesma regra com o ponto imediatamente superior ao ponto de tupla retirado no acervo antigo (A_A) sendo o ponto de tabela relacional TR, pela aplicação da regra-1, existe uma correspondência entre os pontos de tabela relacional respectivamente nos acervos antigo e novo. Analogamente ao exemplo anterior as correspondências restantes são determinadas, aplicando-se as regras 2 e 3.

REGRA 5: APARIÇÃO DE LUGAR

(1) se P' é um ponto de A_N, que foi gerado por uma operação de INSERIR, então P' não é correspondente a nenhum ponto de A_A.

(2) o ponto imediatamente superior ao ponto P' no acervo A_N, será considerado como ponto que derivou de uma operação de alteração, podendo assim para este ponto aplicar-se as regras de 1 a 3.

De forma similar ao explicado para a motivação intuitiva da regra de número 4, temos que no caso de uma nova construção inserida em um acervo em um determinado ponto, no caso desta operação de inserção ser realizada com sucesso, o ponto desta construção não é correspondente a nenhum ponto no acervo antigo já que esta construção não ocrria no acervo antigo.

EXEMPLO 7.8: Considere a figura 7.9 e a seguinte operação

INSERIR OBTER TUPLA

COMPOR(1(A:=5

B:=6)1)

EM ACERVO.ARQ.TR

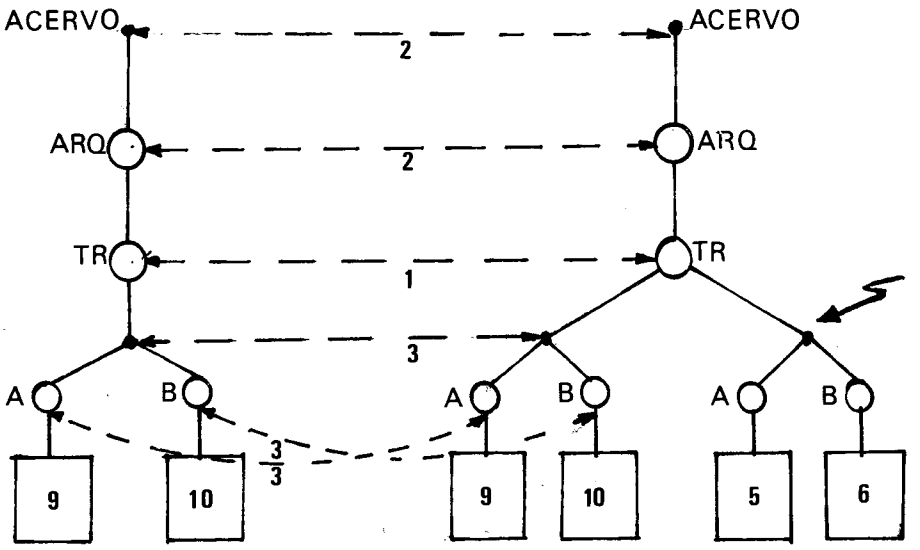


Figura 7.9

Devido a aplicação da regra de número 5 através do item (1), observamos que o ponto de tupla indicado na figura 7.9 por uma seta não tem correspondente no acervo antigo. Aplicando-se o item (2), com o ponto imediatamente superior ao ponto de tupla inserido no acervo novo, sendo o ponto de tabela relacional TR, aplicamos então a regra de número 1, realizando então uma correspondência entre os pontos de tabe-

la relacional dos acervos antigo e novo. E pela aplicação das regras de semelhança e do caminho determinamos as correspondências restantes como mostrado na figura 7.9.

7.7. Aglutinação

Pela aplicação das regras de 1 a 5, podemos garantir que, um dado ponto em A_A tem um e somente um ponto correspondente em A_N. Porém não podemos garantir que um dado ponto em A_N, tenha um e somente um ponto correspondente em A_A, apesar da criação da regra número 5 como vimos anteriormente. Podemos dizer que este conjunto de regras formam uma função dos pontos de A_A, para os pontos de A_N, como visto na figura 7.10, considerando somente os pontos de A_A que não foram retirados.

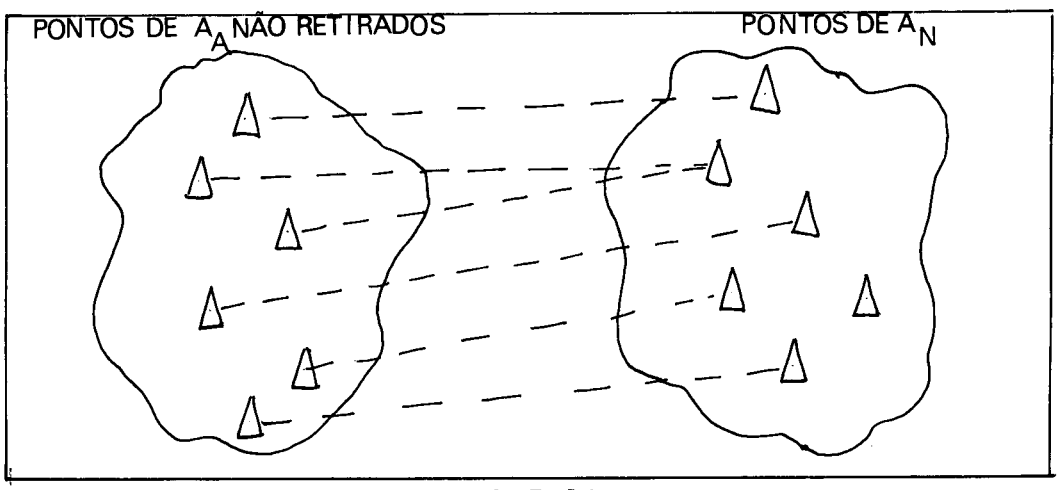


FIGURA 7.10

Descreveremos a seguir, através de um exemplo o que denominamos de aglutinação.

EXEMPLO: Considere a operação realizada por U1 e a proteção mantida (Cap.VIII) pelo usuário U2.

U1: SUBSTITUIR ÁTOMO EM PONTOS ACERVO.ARQ.TR.(C B=7).B POR 9

U2: MANTEM SOB PROTEÇÃO A TUPLA A:=5 , B:=9.

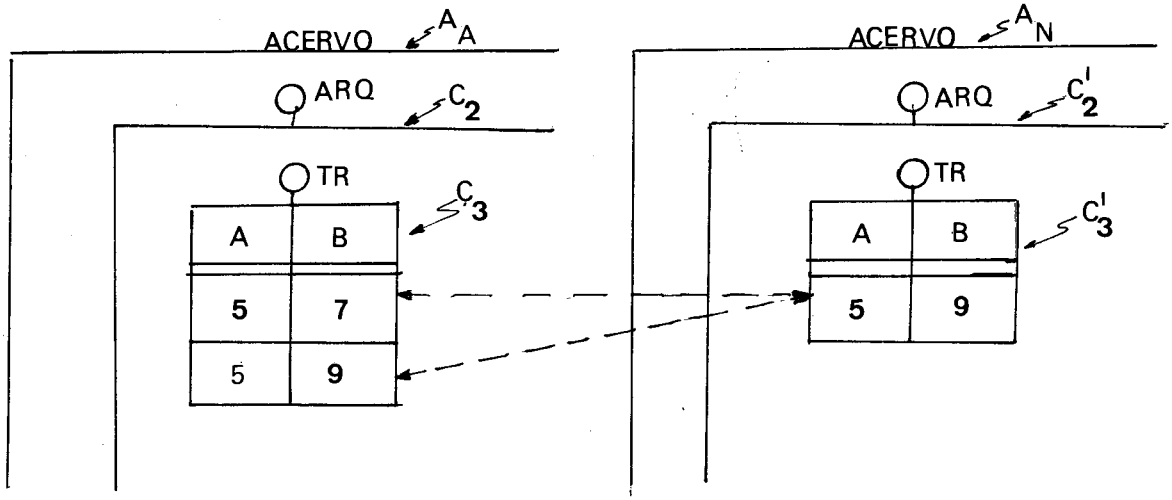


FIGURA 7.11

Vemos que os pontos:

$P_1 = (-,A_A)(ARQ,C_2)(TR,C_3)(-,$ | A | B | 5 | 7 | $)$, que foi manipulado por U1.

e

$P_2 = (-,A_A)(ARQ,C_2)(TR,C_3)(-,$ | A | B | 5 | 9 | $)$, que esta protegido por U2. Têm um único ponto correspondente em A_N, a saber:

$P'_3 = (-,A_N)(ARQ,C'_2)(TR,C'_3)(-,$ | A | B | 5 | 9 | $)$

Este exemplo caracteriza uma situação de aglutinação de dois pontos em um . Porém para estes casos, não poderíamos definir uma regra de tal forma que elimina-se este conflito proviniente da manipulação de pontos em áreas protegidas, sem prejudicar, pelo menos um usuário, como por exemplo a regra de artibrariamente dizer que o pon-

to P'_3, não mais está na área protegida por U2.

É claro que não é uma solução razoável. Claro também é que, nem tudo é permitido ao usuário. Este exemplo caracteriza uma situação de real conflito, que o sistema dificilmente tem condições de impedir a priori, mesmo com a aplicação de métodos sofisticados.

Não há meios de evitar a priori a ocorrência deste fenômenos em geral. E o usuário não está consciente também das alterações que outros usuários estão realizando no decorrer de seus procedimentos.

Como uma solução rígida, o sistema pode tomar as seguintes atitudes, que detetamos. Antes como terminologia dizemos que o mostrado no exemplo anterior foi uma invasão de U1, na área protegida por U2.

- se houver uma invasão por um ou mais usuários em uma área protegida para ALTERAÇÃO/LEITURA então os procedimentos, ou transações dos usuários invasores devem ser canceladas.

O mostrado no exemplo anterior, caracterizou a invasão por parte do usuário U1, na área protegida por U2, assim o sistema cancelaria o procedimento de U1.

Para que o usuário, não corra o risco de inva

são de áreas protegidas algumas providências por eles podem ser tomadas como:

- sempre que possível não efetuar alterações em componentes de chaves de tabelas relacionais, ou componentes de chaves para ligações.
- caso seja necessário a manipulação com componentes de chaves, então o usuário deve requerer uma proteção para toda a tabela relacional ou ligada, para que não corra o risco de ter seu procedimento cancelado. Com esta solução o usuário deve estar consciente de que seu procedimento levará mais tempo para ser executado dado que requer um grande número de pontos, porém evita totalmente o problema da invasão.

CAPÍTULO VIII

MANUTENÇÃO DE ÁREAS PROTEGIDAS

8.1. Área Protegida

É o conjunto de pontos presentes no acervo que estão sob a proteção do sistema em razão da execução de um processo de proteção definido por um usuário.

A obtenção de uma área protegida, como veremos nos capítulos IX e X, depende do método aplicado. Neste capítulo estamos interessados somente em como o sistema mantém uma área protegida, até que haja uma liberação dos pontos protegidos nesta área.

8.2. Aplicação da regra de correspondência para manutenção

Pela aplicação do conceito de ponto vimos que, dado uma alteração qualquer em um acervo, os pontos encontrados para o acervo antes da alteração, não são pontos válidos para o acervo após a alteração. Isto devido ao aspecto de que o conceito de ponto não revela as alterações, isto é, é um conceito estático.

Para dinamizarmos a aplicação do conceito de ponto introduzimos no capítulo VII, o que chamamos de regra de correspondência, que relaciona pontos do acervo antes de alterações com pontos do acervo após alterações. Este regra de correspondência é então embutida no sistema e utilizada para manutenção de áreas protegidas.

A forma pela qual o sistema aplica a regra de correspondência obedece a seguinte regra:

(8.1) se $APROT_A$ designa uma área protegida antiga e $APROT_N$ designa a área correspondente a $APROT_A$, então $APROT_N$ de ser protegida.

O que significa que, os pontos protegidos em um acervo antes da alteração são mantidos protegidos aplicando-se a regra de correspondência. O que pode ser visto na figura 8.1.

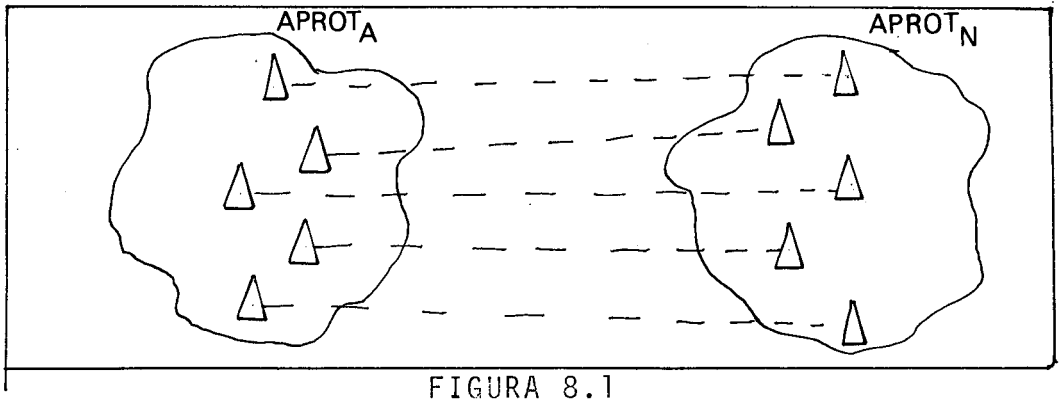


FIGURA 8.1

A figura 8.1, apresenta pontos do acervo antigo, isto é, antes da alteração e seus correspondentes no acervo novo, isto é, após a alteração. Os pontos protegidos em A_A, tem os seus correspondentes protegidos em A_N.

8.3. Cálculo da Área Reclamada para Manutenção

No entanto para a manutenção de áreas protegidas a aplicação somente das regras de correspondência, não é o suficiente, como poderemos verificar através do exemplo 8.1.

EXEMPLO 8.1: Tomemos o bloco de proteção realizado por um usuário U1, relativamente ao acervo apresentado na figura 8.2(a).

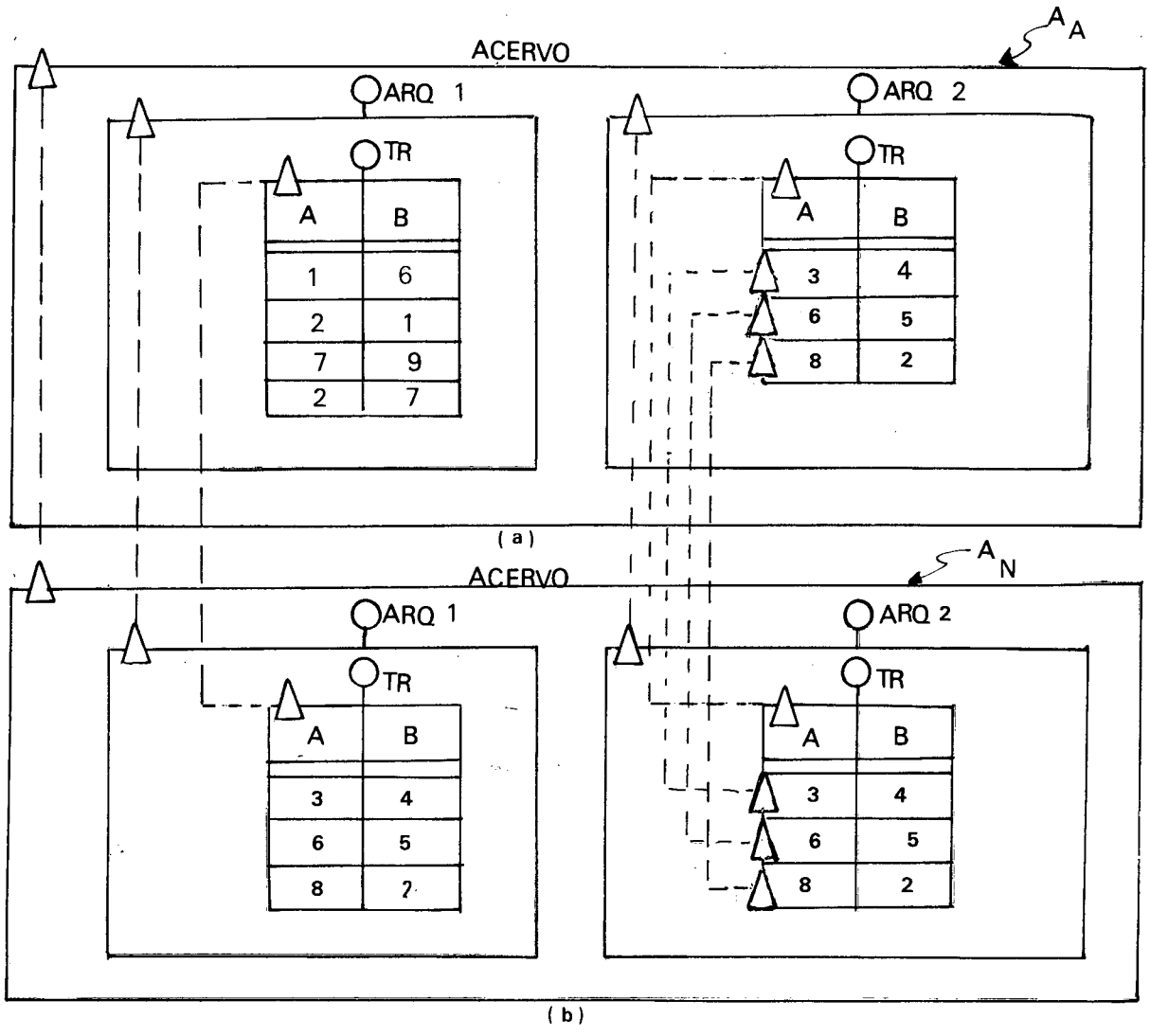


FIGURA 8.2

bloco de proteção definido por U1.

```
U1: ESTABELECER PROTEÇÃO TAB12
    SOBRE {ACERVO.ARQ1.TR} PARA ALTERAR
    E PARA {ACERVO.ARQ2.TR} PARA LER
    SUBSTITUIR TAREL EM ACERVO.ARQ1.TR
                        POR C ACERVO.ARQ2.TR.
                :
                :
                :
    ABANDONAR PROTEÇÃO TAB12
```

Quando da realização do bloco de proteção emitido por U1, teríamos após a execução da operação de substituição a correspondência mostrada na figura 8.2, onde A_A é o acervo antes da realização desta operação de A_N e o acervo após a realização desta operação. Podemos então observar que, para os pontos de tupla da tabela relacional TR, do arquivo ARQ1, no acervo A_A, não existe correspondentes em A_N pela aplicação das regras de correspondência. Neste caso teríamos que o ponto de tabela relacional do arquivo ARQ1, no acervo A_N seria mantida a proteção e no entanto os pontos de tupla relativos a esta tabela não estariam mais sob proteção pois não tem correspondentes em, A_A. Para que esta situação não ocorra, o sistema realizará a manutenção das áreas protegidas, aplicando além da regra de correspondência o seguinte procedimento.

(8.2) Calcular a área reclamada ($AREL_N$) devido a área protegida $APROT_N$, considerando $APROT_N$ como área de partida para o cálculo de $AREL_N$. Isto é, considerar($APROT_N$=APART) e executar os procedimentos apresentados no capítulo VI.

Então pela aplicação dos procedimentos (8.1) e (8.2) o sistema realiza manutenção de áreas protegidas.

CAPÍTULO IX

DETERMINAÇÃO DE CONFLITO POR AVALIAÇÃO DOS ENDEREÇOS DE PONTOS NA FASE DE PEDIDO DE PROTEÇÃO(MÉTODO 1)

Este primeiro método de determinação de conflito está baseado, na real avaliação dos endereços de pontos pelo sistema, para a determinação da existência de conflito na fase de um pedido de proteção. Falamos na real avaliação com o sentido de que o sistema identifica todos os pontos para todos os pedidos de proteção presentes no acervo com o objetivo da determinação de conflitos entre vários pedidos de proteção.

Daremos alguns exemplos a seguir do funcionamento deste método.

EXEMPLO 9.1: Considere dois usuários e os seguintes pedidos de proteção, realizados tendo em vista a figura 9.1.

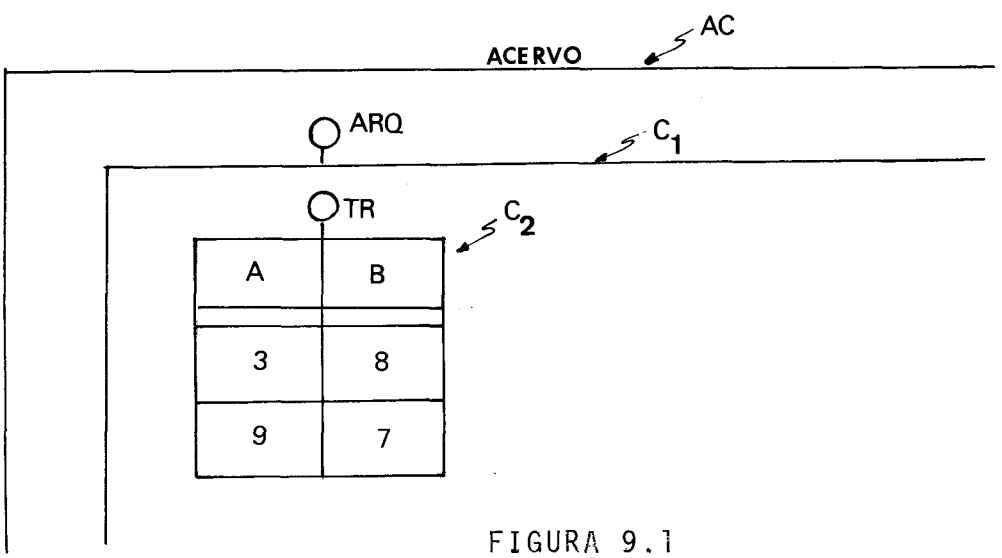


FIGURA 9.1

```
U1: ESTABELECER PROTEÇÃO COLUNA-A SOBRE
    {ACERVO.ARQ.TR.TUP.A} PARA ALTERAR
     < INSTRUÇÃO-1 >
           :
           :
           :
     < INSTRUÇÃO-n >
     ABANDONAR PROTEÇÃO  COLUNA-A

U2: ESTABELECER PROTEÇÃO COLUNA-B SOBRE
    {ACERVO.ARQ.TR.TUP.B} PARA ALTERAR
     < INSTRUÇÃO-1 >
           :
           :
           :
     < INSTRUÇÃO-n >
     ABANDONAR PROTEÇÃO COLUNA-B
```

Ao receber os pedidos de proteção por parte dos usuários U1 e U2, o sistema calcularia as áreas reclamadas associadas a ambos os pedidos como faremos a seguir:

- cálculo da área reclamada
  - área de partida

$P_1 = (-,AC)(ARQ,C_1)(TR,C_2)(-, \begin{array}{|c|c|}\hline A & B \\ \hline 3 & 8 \\ \hline \end{array})(A,\boxed{3})$

$P_2 = (-,AC)(ARQ,C_1)(TR,C_2)(-, \begin{array}{|c|c|}\hline A & B \\ \hline 9 & 7 \\ \hline \end{array})(A,\boxed{9})$

  - área abrangida

    como a área de partida, só contém pontos terminais, isto é, as construções nesses pontos são átomos, temos que a área agrandiga é vazia

  - área conectada

    como não foi realizado um pedido explícito para a prote

ção de pontos sob conexão, e não temos neste exemplo conexões ajustáveis, então a área conectada é vazia.

Assim temos os pontos P_1 e P_2 na área reclamada para o usuário U1.

- cálculo da área reclamada

  - área de partida

$P_3 = (-,AC)(ARQ,C_1)(TR,C_2)(TR,C_2)(-,$ | A | B | | 3 | 8 | $)(B,$ | 8 | $)$

$P_4 = (-,AC)(ARQ,C_1)(TR,C_2)(-,$ | A | B | | 9 | 7 | $)(B,$ | 7 | $)$

  - área abrangida e área conectada

Analogamente ao explicado para o usuário U1, são vazias.

Assim, temos como área reclamada para U2, os pontos P_3 e P_4.

Como $P_1 \neq P_2 \neq P_3 \neq P_4$, os usuários U1 e U2, não estão em conflito, podendo assim o sistema tornar ambas as áreas reclamadas em áreas protegidas.

Com o objetivo de caracterizar quando dois ou mais usuários estão em conflito quando do pedido de proteção, considere o exemplo abaixo.

EXEMPLO 9.2.: Considere dois usuários U1 e U2,com os seguintes pedidos de proteção, tendo em vista o acervo mostrado na figura 9.2.

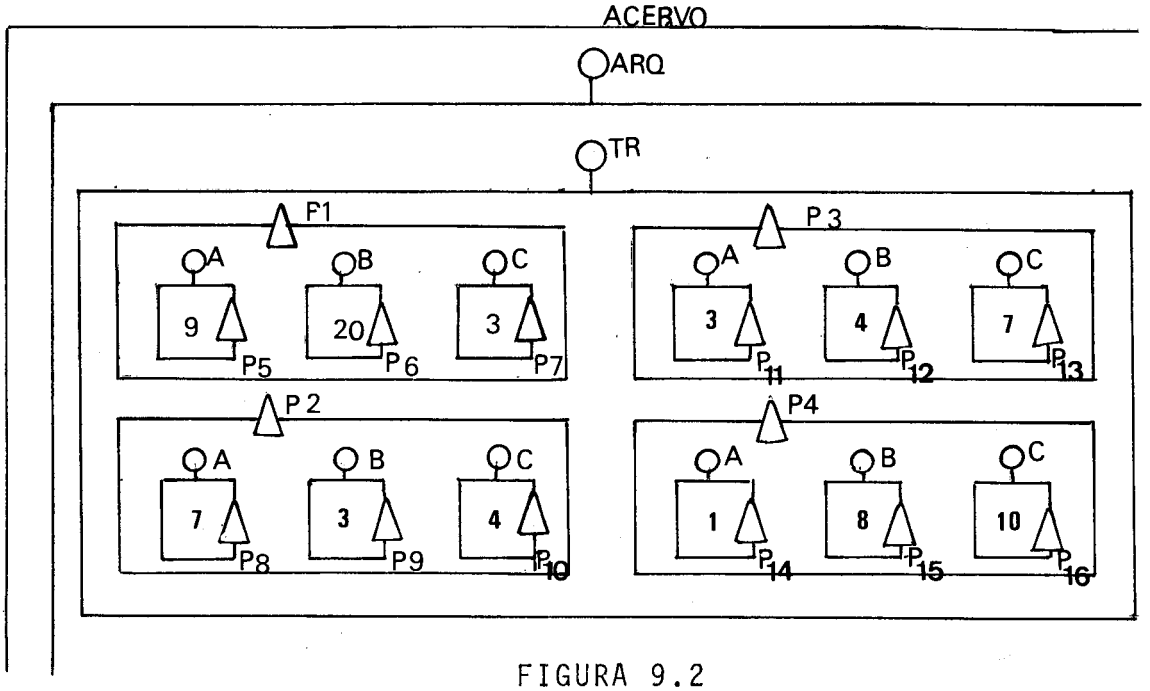


FIGURA 9.2

```
U1: ESTABELECER PROTEÇÃO MOD-A SOBRE
    {ACERVO.ARQ.TR.TUP.A} PARA ALTERAR
    ESTABELECER PROTEÇÃO  LER-U1 SOBRE
    {ACERVO.ARQ.TR.(C C < 8)} PARA LER
     < INSTRUÇÃO-1 >
           :
           :
     < INSTRUÇÃO-n >
    ABANDONAR TODAS AS PROTEÇÕES

U2: ESTABELECER PROTEÇÃO MOD-B SOBRE
    {ACERVO.ARQ.TR.TUP.B} PARA ALTERAR
    ESTABELECER PROTEÇÃO LER-U2 SOBRE
    {ACERVO.ARQ.TR( C C > 6)} PARA LER
     < INSTRUÇÃO-1 >
           :
           :
     < INSTRUÇÃO-m >
     ABANDONAR TODAS AS PROTEÇÕES
```

Tendo em vista os pedidos realizados por U1 e U2, calculemos as áreas reclamadas respectivas.

- áreas de partida

$APART_1 = \{P_5, P_8, P_{11}, P_{14}\}$ devido ao pedido MOD-A

$APART_2 = \{P_1, P_2, P_3\}$, devido ao pedido LER-U1

$APART_1 \cup APART_2 = \{P_1, P_2, P_3, P_5, P_8, P_{11}, P_{14}\} = APART(U_1)$

- áreas abrangidas

$AAB_1 = \emptyset$, associada a $APART_1$

$AAB_2 = \{P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}, P_{13}\}$, associada a $APART_2$

$AAB_1 \cup AAB_2 = \{P_5, \ldots, P_{13}\} = AAB(U1)$

- áreas conectadas

$ACON(U1) = \emptyset$

- temos como área reclamada para U1:

$APART(U1) \cup AAB(U1) \cup ACON(U1) = \{P_1, P_2, P_3, P_5, \ldots, P_{14}\} = AREL(U1)$

- áreas de partida

$APART_1 = \{P_6, P_9, P_{12}, P_{15}\}$ devido ao pedido MOD-B

$APART_2 = \{P_3, P_4\}$, devido ao pedido LER-U2

- áreas abrangidas

$AAB_1 = \emptyset$, associada a $APART_1$

$AAB_2 = \{P_{11}, P_{12}, P_{13}, P_{14}, P_{15}, P_{16}\}$ associada a $APART_2$

$AAB(U2) = \{P_{11}, \ldots, P_{16}\}$

- áreas conectadas

$ACON(U2) = \emptyset$

- temos como área reclamada para U2:

$AREL(U2)=APART(U2) \cup AAB(U2) \cup ACON(U2)=\{P_3,P_4,P_6,P_9,P_{11},\ldots,P_{16}\}$

Se aplicassemos a regra para detetar conflito como:

$AREL(U1) \cap AREL(U2) = \emptyset$

Os usuários U1 e U2 estariam em conflito,pois:

$AREL(U1) \cap AREL(U2) = \{P_3,P_9,P_{11},P_{12},P_{13},P_{14}\}$

porém se observarmos que os pontos de interceção serão manipulados tanto por U1, quanto por U2 somente para leitura. Então na realidade poderíamos permitir que ambos usuários efetuassem suas operações pois não haveria interferência entre eles. Assim a simples condição de que a interação entre as áreas reclamadas seja vazia para que estas áreas se tornem protegidas, diminuiria a concorrência óbviamente no sistema. Para elaborarmos uma condição entre áreas reclamadas, com a finalidade de proteção, considere o seguinte:

9.1. pontos protegidos para leitura, podem ser requeridos e obtidos por um ou mais usuários que desejem efetuar operações de leitura sobre estes pontos.

9.2. pontos protegidos para alteração são de uso exclusivo de um único usuário até a sua liberação.

Tendo em vista as considerações (9.1) e (9.2), podemos então determinar a condição de conflito, ob-

servando que as considerações (9.1) e (9.2) mantêm o princípio da não interferência e assegura a integridade do acervo. Como já vimos as áreas abrangidas e concectadas levam o mesmo tipo de proteção (ler ou alterar), do fornecido para as áreas de partida. Aqui então faremos as distinções entre as várias partes de uma determinada área de partida associada a um usuário Ui.

Denominaremos por:

$APART_L(Ui)$ → os pontos das áreas de partida que estão sendo requeridos somente para leitura, pelo usuário Ui.

$APART_A(Ui)$ → os pontos das áreas de partida que estão sendo requeridos para alteração, pelo usuário Ui.

As áreas abrangidas devido a $APART_L(Ui)$ e a $APART_A(Ui)$ carregam o mesmo tipo de proteção que as áreas de partida $APART_L(Ui)$ e $APART_A(Ui)$, denominaremos pois $AAB_L(Ui)$ e $AAB_A(Ui)$ respectivamente. Da mesma forma as áreas concetadas associada a $APART_L(Ui)$ e $AAB_L(Ui)$ denominaremos $ACON_L(Ui)$ e a área conectada associada a $APART_A(Ui)$ e $AAB_A(Ui)$ denominaremos por $ACON_A(Ui)$. Temos então que a área reclamada associada a Ui será expressa por:

$$AREL(Ui) = AREL_L(Ui) \cup AREL_A(Ui)$$

onde

$$AREL_L(Ui) = APART_L(Ui) \cup AAB_L(Ui) \cup ACON_L(Ui)$$

e

$$AREL_A(Ui) = APART_A(U_i) \cup AAB_A(Ui) \cup ACON_A(Ui)$$

Podemos agora caracterizar situação de con-

flitos entre pedidos de proteção pelo usuários Ui e Uj, pelas condições dada por:

(9.3) $AREL_L(Ui) \cap AREL_A(Uj) = \emptyset$

(9.4) $AREL_L(Uj) \cap AREL_A(Ui) = \emptyset$

(9.5) $AREL_A(Ui) \cap AREL_A(Ui) = \emptyset$

Se as condições (9.3), (9.4) e (9.5) forem satisfeitas então Ui e Uj não estão em conflixo. De uma forma geral dizemos que não existem pedidos conflitantes no sistema se todos os usuários naquele instante do sistema, Ul,..., Un temos, que as condições (9.3), (9,4) e (9.5) são satisfeitas para $i = 1,\ldots, n$ e $j = 1,\ldots,n$ com $i \neq j$.

No caso das condições (9.3), (9.4) e (9.5) serem satisfeitas ainda não indica o sinal verde para que uma área reclamada se torne uma área protegida, pois esta área reclamada pode estar em conflito com uma área protegida por um outro usuário. Assim, para que uma área reclamada possa ser transformada em uma área protegida temos as seguintes condições adicionais:

(9.6) $AREL_L(Ui) \cap APROT_A(Uj) = \emptyset$

(9.7) $AREL_A(Ui) \cap APROT_L(Uj) = \emptyset$

(9.8) $AREL_A(Ui) \cap APROT_A(Ui) = \emptyset$

onde APROT, significa área protegida.

Estas condições traduzem, o fato de que áreas reclamadas para leitura não podem conflitar com áreas protegidas para alteração. Áreas reclamadas para alteração, não

podem conflitar com áreas protegidas para leitura e áreas reclamadas para alterações não podem conflitar com áreas protegidas para alteração. Assim com as condições de (9.3) a (9.8) podemos determinar a existência de conflito na fase de pedido de proteção.

Para o sistema determinar se uma áreas reclamada está conflitante com uma área protegida admitimos que o sistema tem a possibilidade de efetuar operações de leitura sobre áreas protegidas pelos usuários. De outra forma o sistema não poderia detetar este conflito.

Fornecemos a seguir um fluxograma, que indica os caminhos necessários a uma área reclamada para que se torne uma área protegida sob o ponto de vista do método-1.

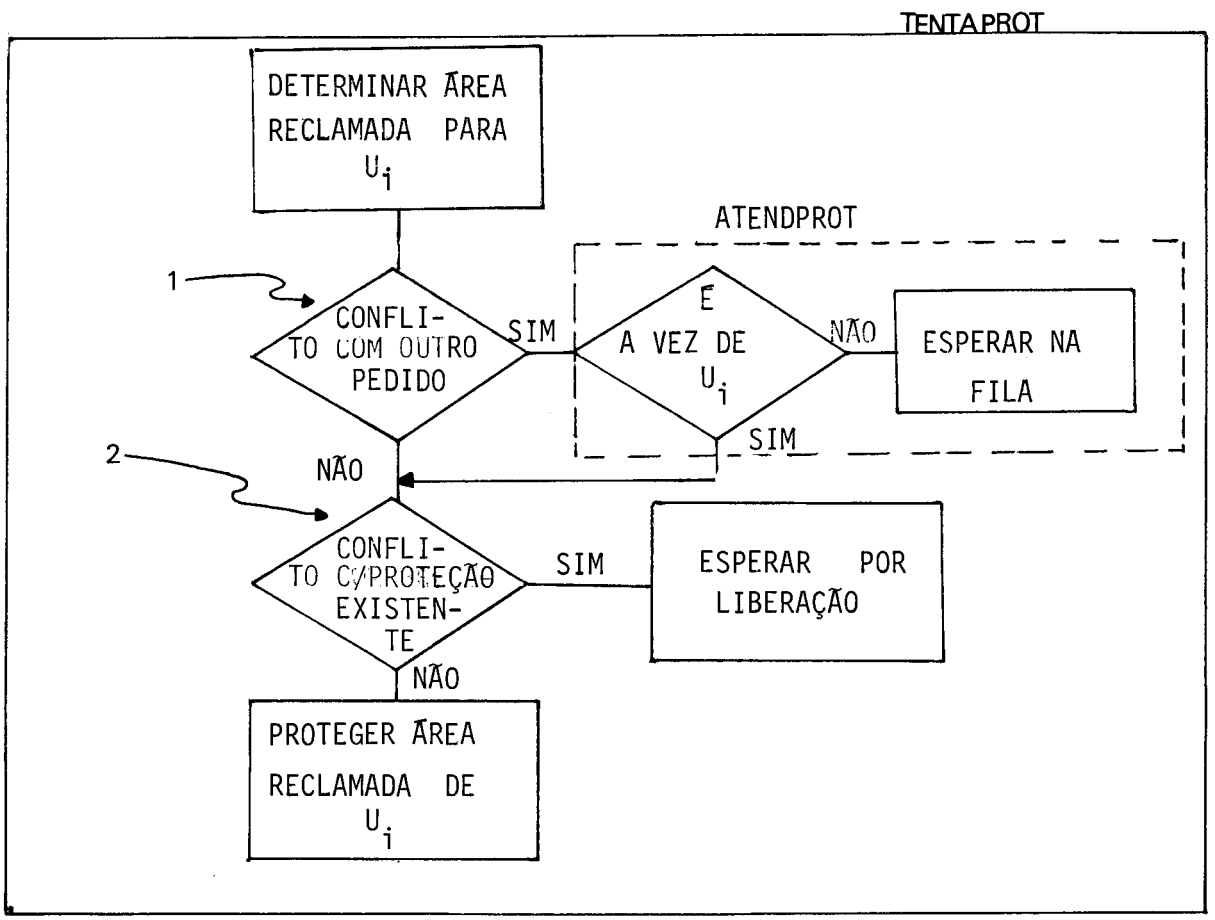


FIGURA 9.3

O Fluxograma TENTAPROT define quais os caminhos para que uma área reclamada se transforme em uma área protegida. As condições aplicadas para detetar conflitos para os blocos 1 e 2, como mostradas na figura 9.3 são respectivamente as condições (9.3),(9.4) e (9.5) para o bloco 1 e (9.6),(9.7) e (9.8) para o bloco 2, condições que foram as vistas anteriormente.

No caso do fluxograma da figura 9.3 quando ocorre um conflito entre áreas reclamadas é função do sistema determinar entre as várias áreas reclamadas conflitantes, qual das áreas reclamadas poderá ser comparada com áreas já protegidas. Isto é, dentre vários pedidos de proteção por parte de vários usuários qual o pedido que deve seguir no caminho para o atendimento. Chamamos esta fase, de fase de conflito entre áreas reclamadas. O bloco do fluxograma que decide nesta fase de conflito quem deve ser atendido é denominado ATENDPROT e está representado por linhas pontilhadas na figura 9.3. O processo definido por este bloco deve satisfazer o seguinte critério:

- Evitar que um usuário espere indefinidamente para ser atendido

Qualquer técnica empregada como por exemplo:

- atendimento por ordem de chegada,isto é,quem chegou primeiro é atendido primeiro
- mecanismos de prioridade,por exemplo quem pede menos é atendido primeiro.E para que não haja uma espera indefinida, cada vez que um usuário não é atendido, sua prioridade de atendimento é aumentada.

O importante, independentemente da técnica aplicada é que satisfaça o critério de evitar que um usuário espere indefinidamente.

Um bloco semelhante a ATENDPROT é a que denominamos ATENDLIB. Este bloco tem como função indicar quando de uma área liberada por um usuário, a quem fornecer esta área liberada. Da mesmo forma que o bloco ATENDPROT, o processo que define o bloco ATENDLIB, deve satisfazer ao critério de evitar que um usuário espere indefinidamente para ser atendido. Qualquer técnica que satisfaça a este critério pode ser aplicada.

Quando um usuário não é atendido imediatamente, então quando da primeira liberação de área realizada,sua área reclamada é novamente avaliada e é verificado se ele pode ou não segundo o bloco TENTAPROT ter sua área reclamada protegida. Este procedimento se efetua até que sua área reclamada se torne área protegida o que é garantido pelo fato de que os processos definidos pelos blocos ATENDPROT e ATENDLIB, não admitem que um usuário espere indefinidamente. Mais tarde daremos um fluxograma total do funcionamento do método-1. O importante a ressaltar como já foi dito, que quando um pedido não é prontamente atendido, a área reclamada é novamente avaliada.

Apresentaremos agora uma indicação de como o sistema calcula a área conectada pelo método de avaliação dos endereços de pontos.

Para o cálculo da área conectada associada a

uma área de partida ou a uma área abrangida o sistema percorre todos os pontos que possuem conexão com os pontos das áreas de partida e abrangidas consultando os verbetes de definição para cada tipo de construção que ocorrem em pontos destas áreas, encontrando assim os pontos conectados, presentes no acervo.

EXEMPLO 9.3.: Suponhamos a seguinte definição de um tipo de arquivo relacional TIPARQ, que aparece no acervo sob nome ARQ

- Tipo de tupla

```
TUPTIP: TIPO DE TUPLA TAL QUE
          COMPOS
          A → TIPA,
          B → TIPB,
          C → TIPC
        CONEX
        C OCOR.A > C OCOR.B
```

- tipo de tabela relacional

```
TABTIP: TIPO DE TAREL TAL QUE
          COMPOS
           TUPTIP
           CONEX

CARD ESTREIT C OCOR
      PARA A, B = CARD C OCOR
                    ∧
      PARA CADA PONTO OCOR.(C A > 3)
      (C PC.A + C PC.B = C PC.C)
```

- O tipo de arquivo TIPARQ, é composto pela tabela relacional do tipo TABTIP.

Suponhamos que um usuário U1, estabeleça uma proteção dada por

U1: ESTABELECER PROTEÇÃO COL-B SOBRE
{ACERVO.ARQ.TR.(CA $\leq$ 2).B} PARA ALTERAR
COM CONEXÃO

Podemos esboçar uma ocorrência deste arquivo, como mostrado na figura 9.6.

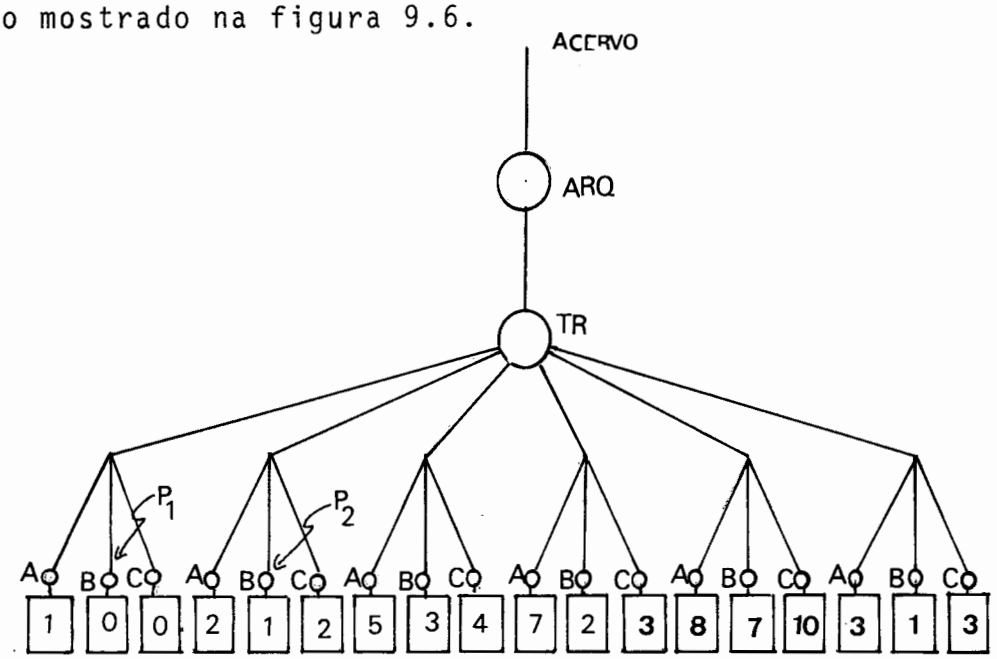


FIGURA 9.6

As áreas de partida associada ao pedido de proteção emitido por U1, contêm os átomos indicados por setas na figura 5. As áreas abrangidas não contêm nenhum ponto, dado que a granulosidade da proteção é atomica. As áreas conectadas seriam calculadas levando-se em conta as conexões dos pontos das áreas de partida com outros pontos. Assim, analisando a definição do tipo de tupla TUPTIP, o sistema verificaria uma conexão existente entre os componentes sob nomes A e B.

Analisando o tipo de tabela relacional o sistema verificaria uma conexão entre os componentes das tuplas deste tipo de tabela sob nomes A, B e C que para o pedido fornecido pelo usuário não seria considerado pois esta conexão está definida somente para as tuplas cujos sob nome A são maiores do que 3, e o usuário deseja os menores que dois. Desta forma teríamos, então como áreas conectadas os pontos P_1 e P_2 como indicado na figura 9.6.

EXEMPLO .4.: Considere o tipo de tupla TUPTIP, definida no exemplo anterior e o seguinte tipo de tabela relacional que compõe um arquivo relacional sob nome ARQ, no acervo

```
TB: TIPO DE TAREL TAL QUE
    COMPOS
     TUPTIP
     CONEX
      CARD ESTREIT C OCOR
           PARA A,B
            = CARD C OCOR
              ∧
         EXIST OCOR.(C A =   ∧ C B = 2)  →

         EXIST OCOR.(C A =   ∧ C B = 2)

∧        AJUSTAR C OCOR.(C A = 5 ∧ C B = 2).C
                        =
         C OCOR.(C A = 7 ∧ C B = 2).C
```

Considere também, a seguinte ocorrência do arquivo sob nome ARQ dado na figura 9.7.

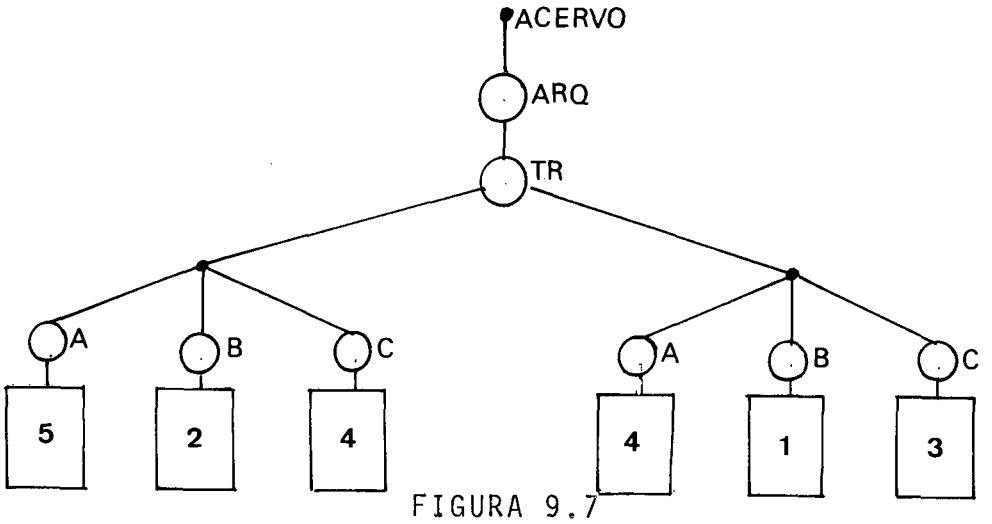


FIGURA 9.7

e o pedido de proteção realizado por um usuário U1 dada por:

U1: ESTABELECER PROTEÇÃO PROT-1 SOBRE
{ACERVO ARQ.TR(C A > 4)} PARA LER
COM CONEXÃO

Teríamos como áreas de partida, o ponto de tupla mostrada na figura 9.7 por uma seta, como áreas abrangidas os pontos para os componentes desta tupla. Como áreas concetadas não teríamos nenhum ponto, pois o pontos de tupla das áreas de partida possuem conexão somente com outros pontos de tupla que não estão presentes na tabela no instante da avaliação das áreas conectadas. Suponhamos então que um outro usuário insira a seguinte tupla na tabela dada por:

A:= 7, B:=2, C:=12

Podemos então observar que no instante da inserção desta tupla entra em vigor a conexão definida para o tipo de tabela TB, e trata-se de uma conexão ajustável, existindo então a possibilidade de quando não protegido os pontos sob conexão ajustável de uma possível interferência entre usuários. A atitude então do sistema para conexões que envol

vem operadores lógicos do tipo implicação é a de uma proteção sobre o primeiro ponto superior no caminho do ponto encontrado nas áreas indicadas no acervo. Por exemplo no caso do exemplo 5.4, o sistema efetuaria uma proteção sobre o ponto da tabela relacional. A política explicada anteriormente também deve ser aplicada quando na definição do verbete de coerência existirem conexões definidas por operadores lógicos de tal forma que sejam equivalentes a uma implicação lógica, como por exemplo:

$\neg$(C A = 0) $\vee$ C Q = 3, que é equivalente a C A = 0 $\rightarrow$ C Q = 3.

CAPÍTULO X

DETERMINAÇÃO DE CONFLITO NA FASE DE UM PEDIDO DE PROTEÇÃO PELA UTILIZAÇÃO DOS PREDICADOS OBTIDOS NOS ENDEREÇOS DE PONTO (MÉTODO 2)

Este método está baseado na detenção de conflito entre dois pedidos de proteção, não só pela existência de pontos em comuns presentes no acervo no instante da avaliação dos pedidos, bem como numa possível existência de pontos em comum que são passíveis de ocorrer no acervo e que satisfazem aos endereços de pontos fornecidos para proteção.

10.1. - Pontos possíveis dado um endereço de ponto

Introduziremos através de um exemplo os pontos possíveis devido a um endereço de ponto.

EXEMPLO 10.1: Suponhamos que o pretipo INTEIRO, seja embutido no sistema e que defina o conjunto dos número inteiros. Defineremos um tipo de acervo que contem somente um arquivo relacional sob nome ARQ.

```
/* TIPOS DE ATOMOS */
 TIPA:TIPO DE ATOMO TAL QUE
      C OCOR ⊂ INTEIRO
                 Λ
      C OCOR > 2  Λ  C OCOR < 5
```

```
TIPB:TIPO DE ATOMO TAL QUE
     C OCOR ⊂ INTEIRO
              Λ
     C OCOR > 7  Λ  C OCOR < 10
/* TIPO DE TUPLA */
TIPTUP: TIPO DE TUPLA TAL QUE
        COMPOS
            A → TIPA,
            B → TIPB
/* TIPO DE TABELA RELACIONAL */
 TAB: TIPO DE TAREL TAL QUE
      COMPOS
      TIPTUP
/* TIPO DE ARQUIVO */
 ARQTIP: TIPO DE AREL TAL QUE
         COMPOS
            TR → TAB
/* TIPO DE ACERVO */
 ACTIPO: TIPO DE ACERVO TAL QUE
          COMPOS
          ARQ → ARQTIP
```

OBS: Deixaremos de definir construções secundárias ou seja de controle das informações contidas no acervo, pois não são relevantes para as pretenções deste exemplo.

Dado então o endereço de ponto :

ACERVO.ARQ.TR.(C A = 4) e a figura 10.1, que apresenta todos os pontos possíveis para o tipo de acervo ACTIPO.

Temos como pontos possíveis relativos a este endereço dado o tipo de acervo ACTIPO:

P_1 = (-,ACERVO)(ARQ,C_2)(TR,C_3)(-,

A	B
4	8

)

P_2 = (-,ACERVO)(ARQ,C_2)(TR,C_3)(-,

A	B
4	9

)

Ou seja devido as definições dos tipos de átomos TIPA e TIPB, e do tipo de tupla TIPTUP, P_1 e P_2 são todos os pontos possíveis cujos componentes sob nome A, sejam iguais a 4.

Dado agora o endereço:

ACERVO.ARQ.TR.(C A = 20), temos que para este endereço nenhum ponto de tupla é passível de ocorrência em cujo componente sob nome A, tenha valor igual a 20, devido a definição do tipo de átomo TIPA, que se restringe aos valores 3 e 4.

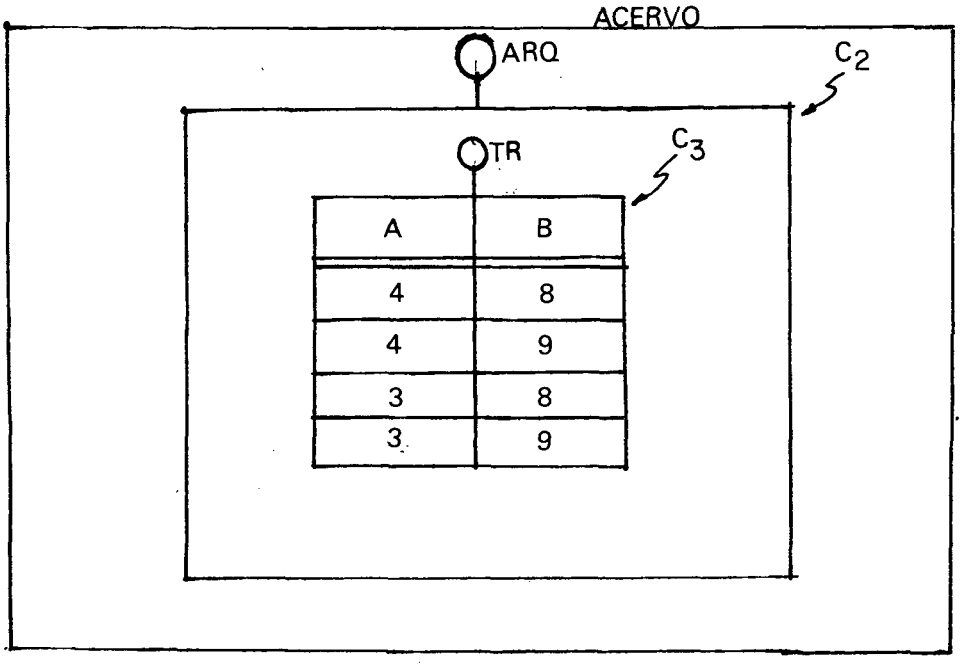


Figura 10.1

Exemplo 10.2: Dada uma ocorrência do tipo de acervo ACTIPO, mostrada na figura 10.2, e os seguintes endereços de ponto:

END1 : ACERVO.ARQ.TR.(C A = 4)

END2 : ACERVO.ARQ.TR.(C B = 9)

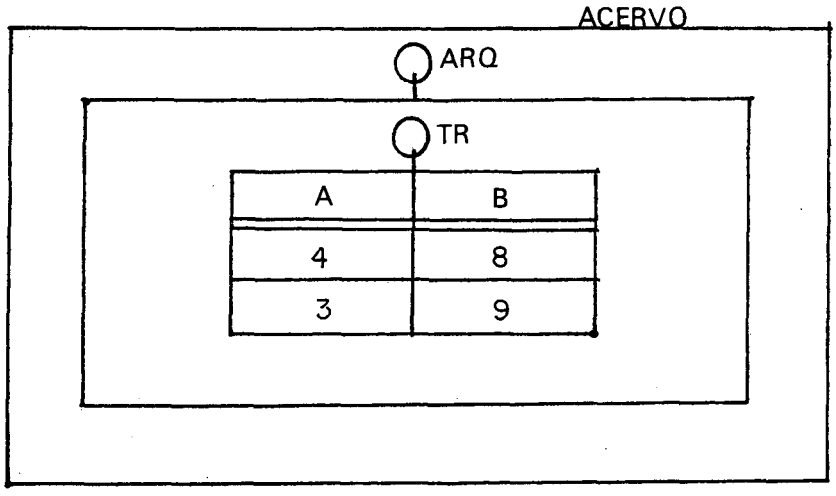


Figura 10.2

É fácil observar que para esta ocorrência do acervo ACTIPO, mostrada na figura 10.2. os endereços END1 e END2, não possuem pontos comuns. No entanto existe um ponto possível para os endereços END1 e END2, a saber o ponto de tupla cujos componentes sob nome A e B, tenha respectivamente os valores 4 e 9, que pode ser visto na ocorrência do tipo de acervo ACTIPO, mostrada anteriormente na figura 10.1. Para efeitos então deste método 2 diríamos que o endereço END1 e END2 são conflitantes. Embora a não existência de conflito para os pontos presentes no acervo no ato da avaliação destes endereços. Mostramos por este exemplo em síntese como funciona o método 2. De forma mais formal temos o seguinte, apresentado na seção 10.2.

10.2. Determinação de endereços conflitantes

Antes da definição formal de endereços confli-

tantes pelo método 2, introduziremos algumas notações necessárias.

Dado um endereço de ponto, para nossa finalidade este endereço será considerado como expressões booleanas intercaladas por pontos:

<end ponto>:=<obter valor booleano>.<obter valor booleano>...

a estas expressões booleanas chamamos de predicados. Assim podemos visualizar um endereço de ponto end como:

end $\equiv (P^{i}_{end})^{n}_{i=1}$, onde P^{i}_{end} representa cada expressão booleana no nível i, e n é denominado o comprimento do endereço end.

Esta visualização para os endereços de ponto aparece mais clara quando estes endereços são escritos na sua forma extensa com notação explícita.

EXEMPLO 10.3: O endereço ACERVO.ARQ.TR.(C A > 5).B escrito na sua forma extensa com notação explícita tem a seguinte forma:

ACERVO.N PX = ARQ.N PX = TR.(C PX.A > 5).N PX = B

assim para este endereço de ponto teríamos:

n = 5 , e:

P^{1}_{end} ≡ ACERVO, que para nossa finalidade define sempre o valor booleano VERDADE, dado que todo endereço de ponto tem como referência inicial sempre o ponto de acervo.

P^{2}_{end} ≡ N PX = ARQ

P^{3}_{end} ≡ N PX = TR

$P^4_{end} \equiv$ C PX.A > 5

$P^5_{end} \equiv$ N PX = B

Dados então dois endereços de pontos:

$end_1 \equiv (P^i_{end_1})^n_{i=1}$ e $end_2 \equiv (P^i_{end_2})^m_{i=1}$

Dizemos que end_1 e end_2 <u>são conflitantes</u> se:

Definindo: $i_{comum} = \min\{n,m\}$ então

$\neg\exists i$ com $1 \leq i \leq i_{comum}$, tal que:

$\neg\exists PV$ tal que $(P^i_{end_1}\ PV \wedge P^i_{end_2}\ PV) =$ VERDADE

ou

$\not\forall i$ com $1 \leq i \leq i_{comum}$, temos que:

$\exists PV$ tal que $(P^i_{end_1}\ PV \wedge P^i_{end_2}\ PV) =$ VERDADE

De forma verbal temos:

Não existe um nível em comum onde a conjunção dos predicados resulta em FALSO para todos os pontos possíveis candidatos ou seja: em todos os níveis comuns existe pelo menos um <u>p</u>onto passível ou <u>v</u>iável (PV), que torna a conjunção dos predicados verdadeira.

A notação P^i_{end} PV, representa e aplicação do predicado P^i_{end} no ponto PV.

Daremos a seguir uma série de exemplos de forma a tornar mais clara a aplicação destas asserções feitas.

EXEMPLO 10.3: Verificação se os endereços end_1 e end_2 dados abaixo são conflitantes.

end_1: ACERVO.ARQ.TR.(C A > 2).B

end_2: ACERVO.ARQ.TR, tendo em vista o acervo de tipo ACTIPO.

Teremos os seguintes predicados para cada endereço:

$P^1_{end_1} \equiv$ ACERVO $\equiv$ VERDADE $\quad P^1_{end_2} \equiv$ ACERVO $\equiv$ VERDADE

$P^2_{end_1} \equiv$ N PV = ARQ $\quad P^2_{end_2} \equiv$ N PV = ARQ

$P^3_{end_1} \equiv$ N PV = TR $\quad P^3_{end_2} \equiv$ N PV = TR

$P^4_{end_1} \equiv$ C PV.A > 2

$P^5_{end_1} \equiv$ N PV = B

$i_{comum} = \min\{5,3\} = 3$, aplicando as asserções temos:

Para i = 1

$\exists$PV (VERDADE $\wedge$ VERDADE), fornece logicamente um resultado VERDADE. O que significa que existe um ponto viável, a saber, o ponto de acervo comum aos dois endereços.

Para i = 2

Descemos então para o nível 2, que significa a avaliação da existencia de um ponto viável de arquivo. Como trata-se de uma nominação óbviamente se tal ponto é viável e é comum a ambos os endereços então deve aparecer sob o mesmo nome. Avaliando então temos:

$\exists$PV(N PV = ARQ $\wedge$ N PV = ARQ) = VERDADE

Por exemplo, neste caso se tivessemos ao invés de $P^2_{end_2} \equiv$ N PV = ARQ, tivessemos $P^2_{end_2} \equiv$ N PV = ARQ1, a ava-

liação da expressão:

$\exists$PV(N PV = ARQ $\wedge$ N PV = ARQ1), forneceria um resultado FALSO, dado que uma mesma construção em um mesmo ponto não pode aparecer com dois nomes.

Para i = 3

Já temos que os dois endereços se referenciam ao mesmo arquivo, agora para i = 3 entramos para a análise dos componentes imediatos para o arquivo ARQ em questão, aí temos que:

$\exists$PV(N PV = TR $\wedge$ N PV = TR) = VERDADE

Dado que PV que satisfaz esta expressão, a saber, é o ponto de tabela relacional contida no arquivo ARQ.

Assim concluimos que end_1 e end_2 são conflitantes.

EXEMPLO 10.4: Considere os endereços; relativamente ao acervo de tipo ACTIPO.

end_1: ACERVO.ARQ1.TR.TUP.A

end_2: ACERVO.ARQ1.TR.TUP.B

temos como forma extensa a notação explícita:

Para end_1:

ACERVO.N PV = ARQ1.N PV = TR.C PV ε TUP.N PV = A

e para end_2:

ACERVO.N PV = ARQ1.N PV = TR.C PV ε TUP.N PV = B

e $i_{comum} = min\{5,5\} = 5$

Obs: O predicado C PV ε TUP, indica que a construção no ponto viável deve pertencer ao pretipo tupla.

Para i = 1,2,3 o processo é análogo ao exemplo anterior.

Para i = 4, temos:

∃PV(C PV ε TUP ∧ C PV ε TUP), o que fornece um resultado verdadeiro, dado que existe um ponto possível de tupla.

Para i = 5, temos como pontos possíveis, todos os pontos, dos componentes imediatos possíveis para o tipo de tupla pertencente a esta tabela relacional. Avaliamos então:

∃PV(N PV = A ∧ N PV = B), fornece o resultado FALSO, pois para o mesmo ponto possível a mesma contrução neste ponto não pode ocorrer com nomes diferentes.

EXEMPLO 10.5: Considere o acervo apresentado na figura 10.3 e os endereços: end_1 e end_2

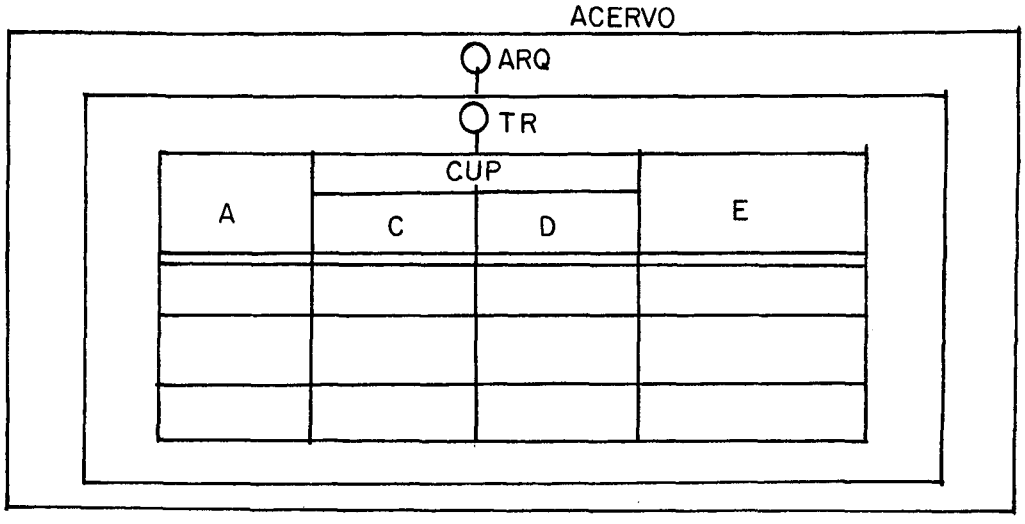


Figura 10.3

end_1: ACERVO.ARQ.TR.(C A > 5).(C D > 10).D

end_2: ACERVO.ARQ.TR.(C A > 5).E

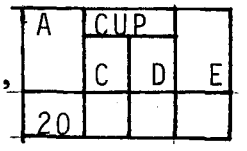
temos $i_{comum} = min\{6,5\} = 5$

Para i = 1,2,3 o procedimento é análogo aos exemplos anteriores.

Para i = 4, temos:

∃PV(C PV.A > 5 ∧ C PV.A > 5) = VERDADE.

Em razão do ponto:

PV = (-,ACERVO)(ARQ,C2)(TR,C3)(- , 

)

Para i = 5, temos:

∃PV(C PV.D > 10 ∧ N PV = E), temos como construções candidatas o mostrado na figura 10.4 por um Δ.

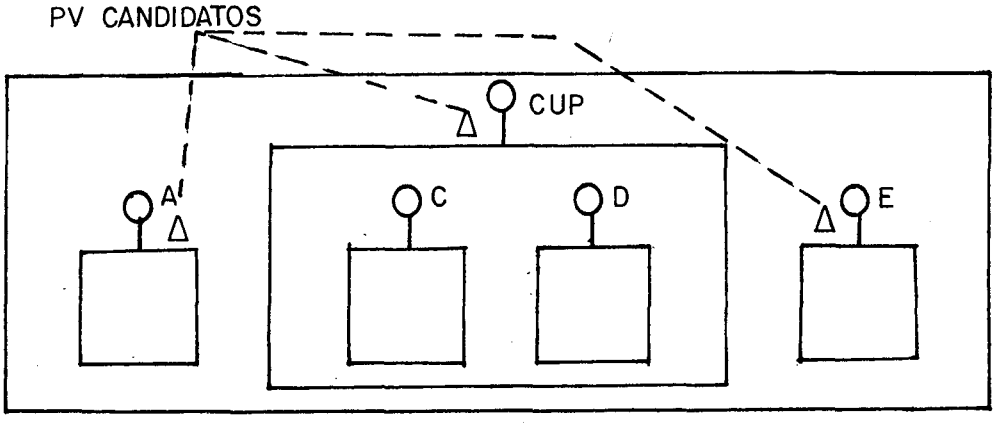


Figura 10.4

Podemos observar que pela construção apresentada na figura 10.4, não existe uma construção a qual apareça sob nome E e tenha como componente imediato uma construção sob nome D. Fornecendo assim a expressão um resultado FALSO.

10.3. O acervo máximo

Dado um tipo de acervo A, definimos como ACER-

VO MÁXIMO, a ocorrencia que contém todos os pontos possíveis para o tipo de acervo A.

Uma forma de visualizar o processo de avaliação de conflito entre dois endereços end_1 e end_2 pela utilização do método 2 é:

- se as áreas indicadas $AIND_{end_1}$ e $AIND_{end_2}$, são avaliadas no acervo máximo não possuirem pontos em comum então end_1 e end_2 não possuem conflito pelo método 2.

EXEMPLO 10.6: Considere os pedidos de proteção realizados por dois usuários U1 e U2 dados a seguir:

```
U1: ESTABELECER PROTEÇÃO LER-A SOBRE
    {ACERVO.ARQ.TR.( C A > 1)}
    POR PREDICADOS PARA LER
       <instrução-1>
             ⋮
       <instrução-n>
    ABANDONAR PROTEÇÃO LER-A
U2: ESTABELECER PROTEÇÃO LER-B SOBRE
    {ACERVO.ARQ.TR.(C B > 5)}
    POR PREDICADOS PARA LER
       <instrução-1>
             ⋮
       <instrução-m>
    ABANDONAR PROTEÇÃO LER-B
```

Levando-se em consideração somente os endereços a serem protegidos teríamos então uma situação de conflito, quando na realidade ambos os usuários poderiam ter pontos em comum, pois nenhum deles realizará operações de alterações. Pa

ra a verificação então de conflito entre endereços relevando as operações a serem realizadas apresentamos o fluxograma VERCONF, mostrado na figura 10.5.

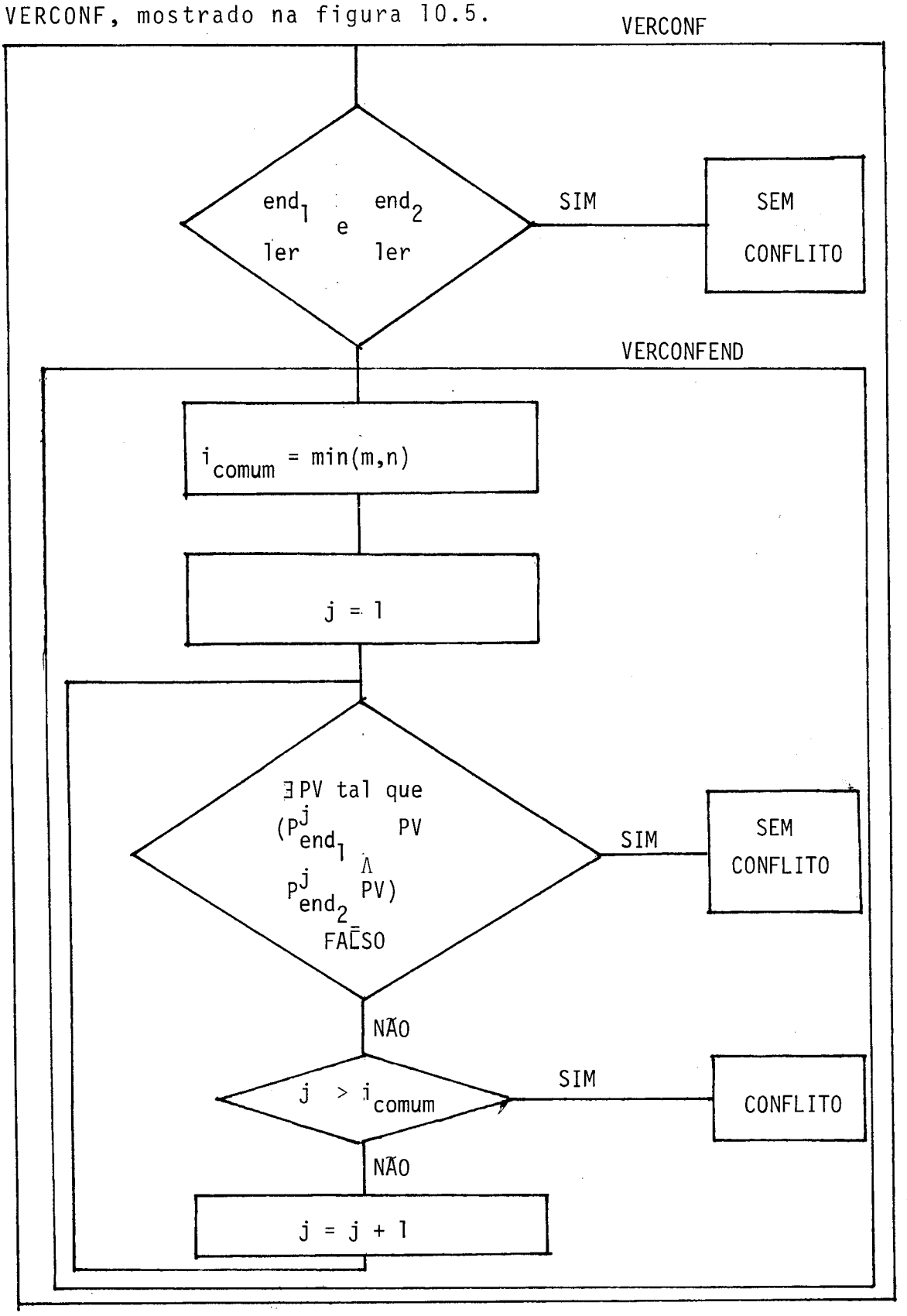


Figura 10.5

Apresentamos na figura 10.5, o fluxograma VERCONF que indica o procedimento para verificação se dois endereços são conflitantes examinando a compatibilidade das operações a serem realizadas nestes endereços, bem como a compatibilidade entre os pontos requeridos, que para a verificação desta compabitilidade assinalamos a parte do fluxograma VERCONF, na figura 10.5, apresentada com o nome VERCONFEND.

10.4. Completamento de endereços

Com o objetivo do tratamento das conexões na seção 10.5, introduziremos alguns conceitos necessários a este objetivo.

10.4.1. Critério global

Um critério global para um determinado nível de referencia é o critério que engloba todos os pontos possíveis neste nível. Daremos na figura 10.6 os critérios globais para cada nível:

NÍVEL	CRITÉRIO GLOBAL
ACERVO	ACERVO
ARQUIVO	C PX ε AREL V C PX ε ALIG
TABELA	C PX ε TAREL V C PX ε TALIG
LIGAÇÃO	C PX ε LIG
LIGANTE	N PX = L
TUPLA	C PX ε TUP
TABELA LIGADA	N PX = T
ATOMO	C PX ε AT

Figura 10.6

Podemos agora fornecer a maneira pela qual o sistema completa endereços.

Dados dois endereços de pontos o primeiro end_1 tendo como ponto de referência inicial o ponto de acervo, e o segundo endereço end_2 um endereço que participa em um verbete de definição que tem como ponto de referência inicial o ponto de ocorrencia (OCOR), o completamento do endereço end_2, em função do endereço end_1 se faz da seguinte forma:

se $end_1 \equiv (P^i_{end_1})^n_{i=1}$ e $end_2 \equiv (P^i_{end_2})^m_{i=1}$

onde $P^1_{end_1} \equiv$ ACERVO e $P^1_{end_2} \equiv$ OCOR

Completar com critérios globais o endereço end_2, para esquerda até o nível de acervo, ficando:

$end_2 \equiv (P^i_{end_2})^{\ell}_{i=1}$ então:

Para $j = 2,\ldots,\min\{\ell-m+1,n\}$

Substituir $P^j_{end_2}$ por $P^j_{end_1}$

Assim denotamos o novo endereço assim obtido por $exp_{end_1}(end_2)$, que significa a expansão do endereço end_2 em função do endereço end_1.

EXEMPLO 10.7: Considere os endereços tendo como referencia o ponto de acervo.

end_1: ACERVO.ARQ.TR.(C A > 5).B $\equiv (P^i_{end_1})^5_{i=1}$

end_2: ACERVO.ARQ.TR$\equiv (P^i_{end_2})^3_{i=1}$

e os endereços que podem aparecer em um verbete como:

end_{v1}: OCOR.(C B > 2).A $\equiv (P^{i}_{end_{v1}})^{3}_{i=1}$

end_{v2}: OCOR.B $\equiv (P^{i}_{end_{v2}})^{2}_{i=1}$

Verifequemos então qual a expressão de $exp_{end_1}(end_{v1})$..

O primeiro passo é completar end_{v1} para esquerda até atingir o nível de acervo. Para isto vamos supor que end_{v1} é um endereço que aparece em um verbete de definição de uma tabela relacional. Ficando então:

$end_{v1} \equiv$ ACERVO.C PX ε AREL V C PX ε ALIG.OCOR.(C B > 2).A

$end_{v1} \equiv (P^{i}_{end_{v1}})^{5}_{i=1}$

$\min\{\ell-m+1,n\} = \min\{5-3+1,5\} = 3$

então j = 2,3 substituimos

$P^{2}_{end_{v1}} \equiv$ C PX ε AREL V C PX ε ALIG por N PX = ARQ $\equiv P^{2}_{end_1}$

e

$P^{3}_{end_{v1}} \equiv$ OCOR por N PX = TR $\equiv P^{3}_{end_1}$

assim temos que:

$exp_{end_1}(end_{v1}) \equiv$ ACERVO.ARQ.TR.(C B > 2).A

aplicando-se então o mesmo procedimento teríamos:

$exp_{end_1}(end_{v2}) \equiv$ ACERVO.ARQ.TR.(C A > 5).B

$exp_{end_2}(end_{v1}) \equiv$ ACERVO.ARQ.TR.(C B > 2).A

$exp_{end_2}(end_{v2}) \equiv$ ACERVO.ARQ.TR.C PX ε TUP.B

10.5. Conexões

Como mostramos para o método 1, apresentaremos

agora também para o método 2, como o sistema pode realizar o tratamento das conexões. Para isto introduziremos mais alguns conceitos.

10.5.1 - Endereços soldados

Dizemos que dois endereços de ponto end_1 e end_2 estão soldados, quando pela aplicação do fluxograma VERCONFEND (Figura 10.5), para end_1 e $exp_{end_1}(end_2)$, fornece um resultado de conflito.

10.5.2 - Endereços ligados

Dizemos que dois endereços de ponto end_1 e end_2 que aparecem em um verbete de definição de tipo, estão ligados, quando end_1 e end_2 aparecem participando de uma mesma expressão booleana elementar em lados opostos relativamente ao operador de comparação.

EXEMPLO 10.8: Considere a definição de um tipo de tupla X.

```
X: TIPO DE TUPLA TAL QUE
   COMPOS
     A → TA,
     B → TB,
     C → TC,
     D → TD,
     E → TE
   CONEX
                 end1                end2                end3
     C [OCOR.A]  > C [OCOR.B]  +  C [OCOR.C]
                 end4                end5
     Λ C [OCOR.E]  < C [OCOR.D]
```

Neste exemplo temos que end_1 está ligado a end_2 como também end_1 a end_3, e end_4 ligado a end_5. Porém end_2 não está ligado a end_3 apesar de participar da mesma expressão booleana elementar, pois end_2 e end_3 estão do mesmo lado na expressão. Temos também que end_2 ou end_1 ou end_3 não estão ligados a end_4 ou end_5 dado que não participam da mesma expressão booleana elementar.

10.5.3 - Endereços relacionados

Dizemos que end_2 está relacionado a end_1, onde end_1 é um endereço participante de um pedido de proteção e end_2 é um endereço que aparece em um verbete se:

- end_1 está soldado a $exp_{end_1}(end_a)$

e

- end_a está ligado a end_2

Agora forneceremos o procedimento realizado pelo sistema para o tratamento das conexões.

Suponhamos que o usuário forneça o pedido abaixo:

U1: ESTABELECER PROTEÇÃO <id-prot> SOBRE

{end} POR PREDICADOS PARA { ALTERAR / LER }

COM CONEXÃO

Seria então realizado pelo sistema:

1. Calcular os endereços soldados e relacionados ao endereço end varrendo todos os verbetes de definição de tipo referenciados pelo endereço end. Incluir estes endereços em sua forma expandida no pedido de proteção do usuário.

2. Para cada endereço incluido efetuar o procedimento especificado em 1.

Obs: Logicamente endereços que já constem no pedido ou seja por serem fornecidos ou seja por serem incluidos, não podem aparecer duas vezes.

A seguir daremos um exemplo de como é realizado pelo sistema o tratamento das conexões para o método 2.

EXEMPLO 10.9: Considere a definição de um tipo de acervo

- tipo de tupla

```
TIPTUP: TIPO DE TUPLA TAL QUE
        COMPOS
          A → ATIP,
          B → BTIP,
          C → CTIP
        CONEX
          C OCOR.A > C OCOR.B
```

- tipo de tabela relacional

```
TIPTAB: TIPO DE TAREL TAL QUE
        COMPOS
          TIPTUP
        CONEX
    (*)  CALC SOMA OCOR.TUP.A >
         CALC SOMA OCOR.TUP.B
                Λ
         PARA CADA PONTO OCOR.( C B > 5)
           ( C PC.A > C PC.C)
```

(*) Obs: Supomos que no sistema já existe uma função de nome SOMA, que realiza a soma das construções nos pontos endereçados.

- tipo de ligação

```
LIGTIP: TIPO DE LIGAÇÃO TAL QUE
        COMPOS
         L → TIPTUP
         T → TIPTAB
```

- tipo de tabela ligacional

```
TIPTABL: TIPOS DE TALIG TAL QUE
         COMPOS
           LIGTIP
         CONEX
           PARA CADA PONTO OCOR.LIG
           (   C PC.L ELEM C PC.T)
         /*UM LIGANTE NÃO PODE APARECER
           COMO TUPLA LIGADA NA MESMA
           LIGAÇÃO*/
```

- tipos de arquivos

```
ARQTIP1: TIPO DE AREL TAL QUE
         COMPOS
           TR → TIPTAB
ARQTIP2: TIPO DE ALIG TAL QUE
         COMPOS
           TL → TIPTABL
```

- E finalmente um acervo definido por:

```
ACTIP: TIPO DE ACERVO TAL QUE
          COMPOS
            ARQ1 → ARQTIP1,
            ARQ2 → ARQTIP2
          CONEX
            COLEC OCOR.ARQ1.TR.TUP
                          =
            COLEC OCOR.ARQ.TL.LIG.L
/* ESTA CONEXÃO DEFINE QUE, PARA CADA TUPLA DA TABELA
   RELACIONAL NO ARQUIVO ARQ1, DEVE EXISTIR UMA E SO-
   MENTE UMA TUPLA LIGANTE NA TABELA LIGACIONAL DO AR
   QUIVO ARQ2, E VICE-VERSA */
```

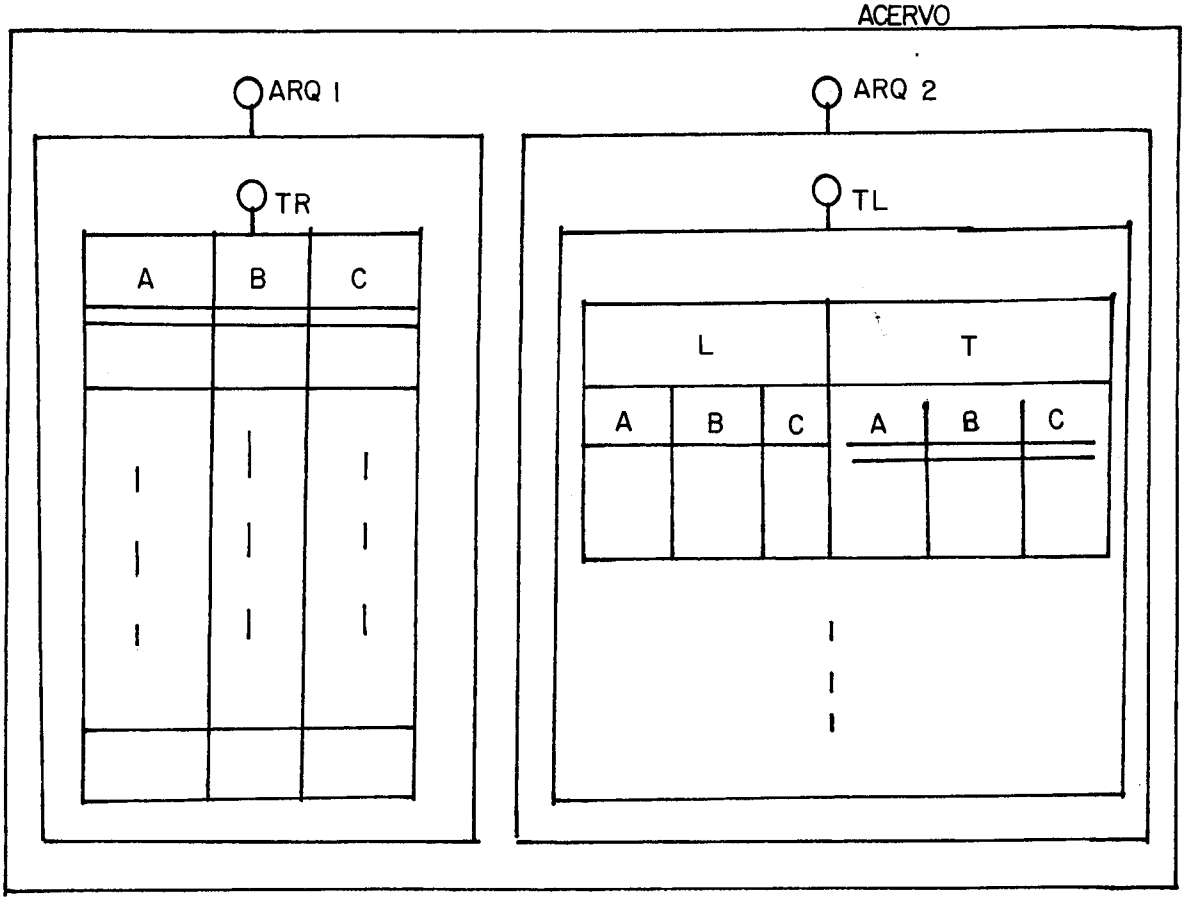


Figura 10.7

Agora considere o pedido de proteção emitido por um usuário U1:

U1: ESTABELECER PROTEÇÃO X SOBRE
{ACERVO.ARQ1.TR.(C B < 2).A}
POR PREDICADOS PARA ALTERAR COM CONEXÃO

O endereço de ponto fornecido para a proteção tem a seguinte forma explícita:

end ≡ ACERVO.N PX = ARQ1.N PX = TR.(C PX.B < 2).N PX = A

Então o sistema começaria pelo tratamento das conexões a nível de acervo, pois óbviamente é o primeiro tipo referenciado pelo endereço end. A nível de acervo temos na conexão lá definida os seguintes endereços de pontos envolvidos, que escreveremos de forma explícita:

end_1 ≡ OCOR.N PX = ARQ1.N PX = TR.C PX ε TUP

end_2 ≡ OCOR.N PX = ARQ2. N PX = TL.C PX ε LIG.N PX = L

verificamos através do fluxograma VERCONFEND da figura 10.5, a existência de conflito entre os endereços $exp_{end}(end_1)$ com o endereço fornecido, por U1 para proteção (end). Isto é, está indicado que o endereço fornecido por U1, pode endereçar um ponto possível, que também é um ponto possível para o endereço $exp_{end}(end_1)$.

Assim temos:

end soldado a end_1

e

end_1 está ligado a end_2

incluiríamos assim $exp_{end}(end_1)$ e $exp_{end}(end_2)$ ficando o pedido após a realização deste passo com os endereços:

{ ACERVO.ARQ1.TR.(C B < 2).A,
  ACERVO.ARQ1.TR.TUP,
  ACERVO.ARQ2.TL.LIG.L }

isto porque end soldado a end_1
e
end relacionado a end_2

Esgotadas as conexões a níveis de acervo, passaria então as conexões a nível de arquivos. Temos que o arquivo referenciado pelo endereço end é o arquivo ARQ1, e definido pelo tipo de arquivo ARQTIP1.

Podemos observar que para o tipo de arquivo ARQTIP1 não é definida qualquer conexão. Assim o sistema passaria ao tratamento das conexões a nível de tabela relacional. Temos como tipo de tabela relacional referenciada pelo endereço end o tipo TIPTAB onde encontramos os seguintes endereços:

end_3 ≡ OCOR.TUP.A
end_4 ≡ OCOR.TUP.B
end_5 ≡ PC.A, que é equivalente a OCOR.(C B > 5).A
devido ao endereço OCOR.(C B > 5)
end_6 ≡ PC.C, que é equivalente a OCOR.(C B > 5).C
por razão análoga a end_5

As expansões dos endereços de end_3 a end_6 tendo como referencia o endereço end seriam:

$exp_{end}(end_3) \equiv$ ACERVO.ARQ1.TR.TUP.A

$exp_{end}(end_4) \equiv$ ACERVO.ARQ1.TR.TUP.B

$exp_{end}(end_5) \equiv$ ACERVO.ARQ1.TR.(C B > 5).A

$exp_{end}(end_6) \equiv$ ACERVO.ARQ1.TR.(C B > 5).C

Verificamos aplicando o fluxograma VERCONFEND

end soldado a end_3

end relacionado a end_4

e end não possui relação alguma com end_5 e end_6.

Neste ponto teríamos como endereços incluidos os endereços end_3 e end_4. O pedido estaria assim como:

{ ACERVO.ARQ1.TR.(C B < 2).A,
ACERVO.ARQ1.TR.TUP,
ACERVO.ARQ2.TL.LIG.L,
ACERVO.ARQ1.TR.TUP.A,
ACERVO.ARQ1.TR.TUP.B }

Após analise do nível de tabela relacional o sistema passaria para a análise do tipo de tupla referenciado que é TIPTUP, onde encontramos os endereços:

$end_7 \equiv$ OCOR.A

$end_8 \equiv$ OCOR.B

que expandidos relativamente a end teríamos:

$exp_{end}(end_7) \equiv$ ACERVO.ARQ1.TR.C PX ε TUP.A

$exp_{end}(end_8) \equiv$ ACERVO.ARQ1.TR.C PX ε TUP.B

onde teríamos que:

end soldado a end_7

end relacionado a end_8

Porém nem end_7, nem end_8 seriam incluidos pois já existem ambos os endereços incluidos no pedido do usuário.

O nível atomico não requer análise porque para este nível não existem definições de conexões.

Efetuando-se então o mesmo procedimento realizado para o endereço end para cada endereço incluido no pedido de proteção obteríamos a partir do pedido inicial:

```
U1: ESTABELECER PROTEÇÃO <id-prot> SOBRE
    {ACERVO.ARQ1.TR.( C B < 2).A} POR PREDICADOS
    PARA ALTERAR COM CONEXÃO
```

o pedido equivalente:

```
U1: ESTABELECER PROTEÇÃO <id-prot> SOBRE
    {ACERVO.ARQ1.TR.( C B < 2).A,
     ACERVO.ARQ1.TR.TUP,
     ACERVO.ARQ.TL.LIG.L,
     ACERVO.ARQ1.TR.TUP.A,
     ACERVO.ARQ1.TR.TUP.B,
     ACERVO.ARQ2.TL.LIG.T.TUP}
     POR PREDICADOS PARA ALTERAR
```

Para minimizar a possibilidade de conflito sempre que o sistema encontra-se em critério global como por exemplo TUP ou LIG, que são critérios que envolvem todas as tuplas ou todas as ligações, poderia trocar este critério por um critério mais fraco, que é o critério de tuplas ou de ligações que aparece nos endereços fornecidos inicialmente no pedido de proteção. Porém esta troca só seria feita para os endereços a mesma tabela relacional ou ligacional. Por exemplo

no caso dos endereços:

ACERVO.ARQ1.TR.TUP, ACERVO.ARQ1.TR.TUP.A,
ACERVO.ARQ1.TR.TUP.B

que poderiam ser transformados para:

ACERVO.ARQ1.TR.(C B > 2),
ACERVO.ARQ1.TR.(C B > 2).A,
ACERVO.ARQ1.TR.(C B > 2).B

Estes aspectos estão ligados a um conceito mais geral que é o de intercessão entre endereços de pontos.

CAPÍTULO XI

CONSIDERAÇÕES SOBRE OS MÉTODOS 1 E 2 PARA DETETAR CONFLITO ENTRE PEDIDOS DE PROTEÇÃO

Para realizarmos considerações sobre ambos os métodos já apresentados, considere o fluxograma apresentado na figura 11.1.

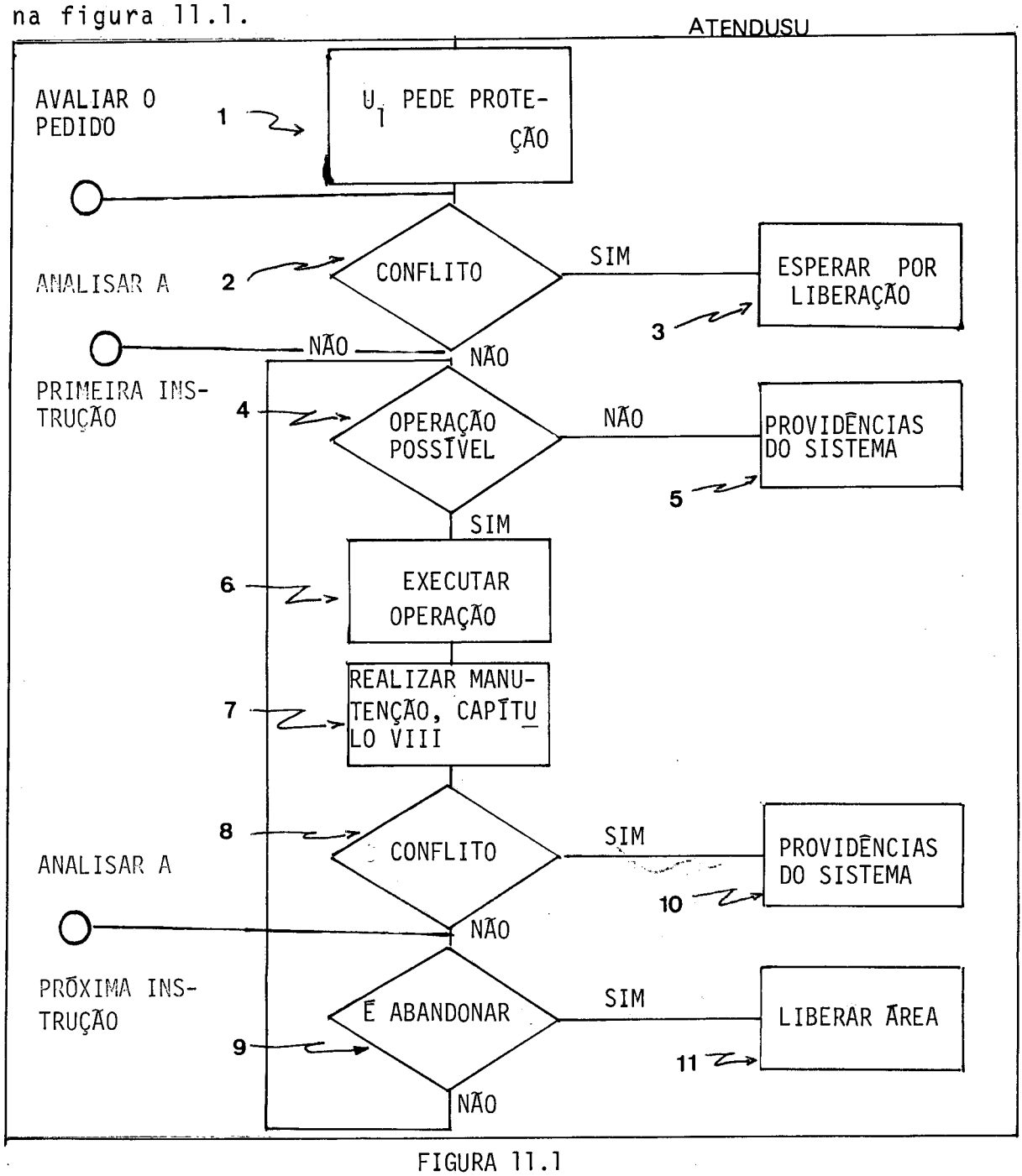


FIGURA 11.1

Faremos a seguir a análise dos blocos que aparecem numerados de 1 a 11 na figura 11.1, para cada método:

- BLOCO 1: o usuário realiza um pedido de proteção

- MÉTODO-1

Dado um pedido de proteção, o sistema calcula a área reclamada associada ao pedido fornecido, considerando somente as conexões ajustáveis, e no caso do pedido de proteção for realizado com a inclusão do termo COM CONEXÃO, então o sistema realiza o tratamento de todas as conexões como mostrado no capítulo IX.

- MÉTODO-2

Em caso de um pedido explícito através do termo COM CONEXÃO, o sistema realiza o tratamento de todas as conexões ajustáveis ou não segundo o apresentado no capítulo X, e no caso da não inclusão do termo COM CONEXÃO, somente as conexões ajustáveis seriam consideradas. Assim após o tratamento das conexões teríamos um pedido equivalente ao pedido original emitido pelo usuário. O sistema então analisaria os endereços encontrados neste pedido equivalente, transformando cada um desses endereços para a forma:

$$end \boxminus (P^{i}_{end})^{n}_{i=1}$$

- BLOCO 2: Conflito?

- MÉTODO-1

O conflito entre o pedido de proteção realizado pelo usuário U1, em relação a outros usuários é determinado co

mo já vimos no fluxograma TENTAPROT, apresentado no capítulo IX, pela comparação da área reclamada devido ao pedido realizado por U1 com áreas reclamadas ou áreas protegidas por outros usuários.

- MÉTODO 2

A verificação de conflito é realizada por meio de uma tabela que contém todos os pedidos de proteção no momento, no sistema. Esta tabela está representada na figura 11.2.

SITUAÇÃO	IDENTIFICAÇÃO DO USUÁRIO E DO PEDIDO	PEDIDO
ATIVO	<id-prot-1>/U1	$(P^i_{end_1})^n_{i=1},\ldots,(P^i_{end_n})^m_{i=1}$ PARA LER
NÃO ATIVO	<id-prot-5>/U2	$(P^i_{end_{27}})^i_{i=1},\ldots,(P^i_{end})^k_{i=1}$ PARA ALTERAR
	' ' '	
ATIVO	<id-prot-6>/U1	- - -

FIGURA 11.2

O pedido de proteção do usuário U1, é avaliado contra cada pedido encontrado na tabela apresentada na figura

11.2, segundo o fluxograma VERCONF, apresentado no capítulo X. Se o resultado dessa avaliação for uma situação de conflito então o pedido é inserido na tabela com situação NÃO ATIVO. Caso não ocorra conflito o pedido é inserido na tabela com situação ATIVO.

- BLOCO 3 : ESPERAR POR LIBERAÇÃO

- MÉTODO 1 e MÉTODO 2

Em ambos os métodos verificado uma situação de conflito os pedidos dos usuários devem esperar, até que haja uma liberação.

- BLOCO 4 : OPERAÇÃO POSSÍVEL

- Método 1

Pelo método-1, temos os seguintes casos para determinar se uma operação desejada é possível.

(1) As operações de SUBSTITUIR e MODIFICAR, são sempre possíveis de serem realizadas. Veremos quando falarmos sobre o bloco 6, porque isto é possível.

(2) Uma instrução de INSERIR, só é válida no caso do endereço de ponto desta instrução só identificar pontos não protegidos ou pontos protegidos pelo usuário que emite esta instrução. Por exemplo, um usuário não pode inserir uma tupla em uma tabela, caso este ponto de tabela esteja sob proteção de outro usuário. No entanto este usuário poderá inserir tuplas em uma tabela, mesmo que todas as tuplas

desta tabela estejam sob proteção desde que o ponto de tabela não esteja sob proteção.

(3) Uma instrução de RETIRAR,só é válida no caso do endereço de ponto desta instrução identificar somente pontos protegidos para o usuário que emitiu esta instrução.

- Método - 2

Sendo: end_{prot} → consta de um pedido de proteção

end_{op} → consta de uma instrução a ser executada (operação)

Dois endereços end_{prot} e end_{op} são <u>compatíveis</u> se:

$$\neg\exists i(1 \leq i \leq i_{max})(\exists PV(P^i_{end_{op}}\ PV \wedge P^i_{end_{prot}}\ PV) = \text{VERDADE}$$

ou seja

$$\forall i(1 \leq i \leq i_{max})(\forall PX(P^i_{end_{op}}\ PV \rightarrow P^i_{end_{prot}}\ PV) = \text{VERDADE}$$

Em nenhum nível existe um ponto viável que atende o critério end_{op} e não o critério end_{prot}. O menor endereço entre end_{op} e end_{prot} será completado com critérios globais até $i_{max} = \max\{i_{op}, i_{prot}\}$

Assim temos:

- Um endereço end_{op} sendo utilizado para a realização de uma alteração/leitura, isto é, os pontos identificados por este endereço sofrerão alteração/leitura, é um endereço <u>compatível totalmente</u> , se o endereço end_{op} é <u>compatível</u> com pelo menos um endereço referenciado nas instruções de ESTABELECER PROTEÇÃO para ALTERAR/LER,da qual este endereço está imerso dentro dos blocos de proteção definidos por estas instruções de ESTABELECER PROTEÇÃO.

- BLOCO 5 : PROVIDÊNCIA DO SISTEMA

- MÉTODO 1 e MÉTODO 2

Quando a operação desejada não é possível de ser realizada o sistema deve interromper a execução desta proteção e imediatamente liberar a área protegida associada ao usuário que submeteu a operação, porém esta operação só seria realizada após o sistema realizar a reconstrução regressiva.

- BLOCO 6: EXECUTAR OPERAÇÃO

- MÉTODO 1

Ao executar as operações de SUBSTITUIR ou MODIFICAR, todos os endereços de pontos referenciados nestas instruções, serão avaliados somente com relação aos pontos protegidos. Este procedimento é justificado pelo seguinte fato apresentado no exemplo 11.1, onde na primeira parte apresentaremos como é realizado este procedimento pelo sistema e na segunda parte porque a aplicação deste procedimento.

EXEMPLO 11.1: O usuário emite um pedido de proteção dado por:

```
        ESTABELECER PROTEÇÃO EX SOBRE
        {ACERVO.ARQ1.TR.(C A > 5)} PARA ALTERAÇÃO
        <INSTRUÇÃO-1>
              :
              :
              :
  ---->   SUBSTITUIR ITEM EM ACERVO.ARQ-2.TR.(C X > 5).Y POR 4
              :
              :
        <INSTRUÇÃO-n>
        ABANDONAR PROTEÇÃO EX
```

A instrução indicada pela seta, fornece um endereço de ponto, que não foi protegido. Quando o sistema avaliasse este endereço de ponto na área protegida, forneceria um conjunto de pontos vazio. O sistema também poderia considerar como um endereçamento fora da área protegida e assim resultando em erro. Porém ao nosso ver, se considerassemos como endereçamento fora da área protegida, em muitos casos, o sistema teria que considerar como erro os endereços que indicassem por exemplo pontos de tuplas em uma tabela relacional quando estes pontos de tuplas estivessem em áreas protegidas por outros usuários que os criaram por meio de operações de inserção. Por exemplo: considere dois usuários U1 e U2, e os seguintes procedimentos:

```
EXEMPLO 11.1(b): U1:ESTABELECER PROTEÇÃO PROTU1 SOBRE
                    {ACERVO.ARQ.TR.(C A > 5)} PARA ALTERAÇÃO
                    <INSTRUÇÃO-1>
                          :
                          :
                          :
                 (1) SUBSTITUIR ITEM EM ACERVO.ARQ.TR.(C A>5).B
                                        POR 3
                          :
                          :
                          :
                    <INSTRUÇÃO-n>
                   ABANDONAR PROTEÇÃO PROTU1
```

```
U2: ESTABELECER PROTEÇÃO PROTU2 SOBRE
    {ACERVO.ARQ.TR.(C B > 10)} PARA ALTERAÇÃO
    <INSTRUÇÃO-1>
          :
          :
(2) INSERIR OBTER TUPLA COMPOR( A:=20,
                                B:=50)
    EM ACERVO.ARQ.TR.
ABANDONAR PROTEÇÃO PROTU2
```

e o seguinte acervo, no instante da avaliação dos pedidos de proteção para U1 e U2.

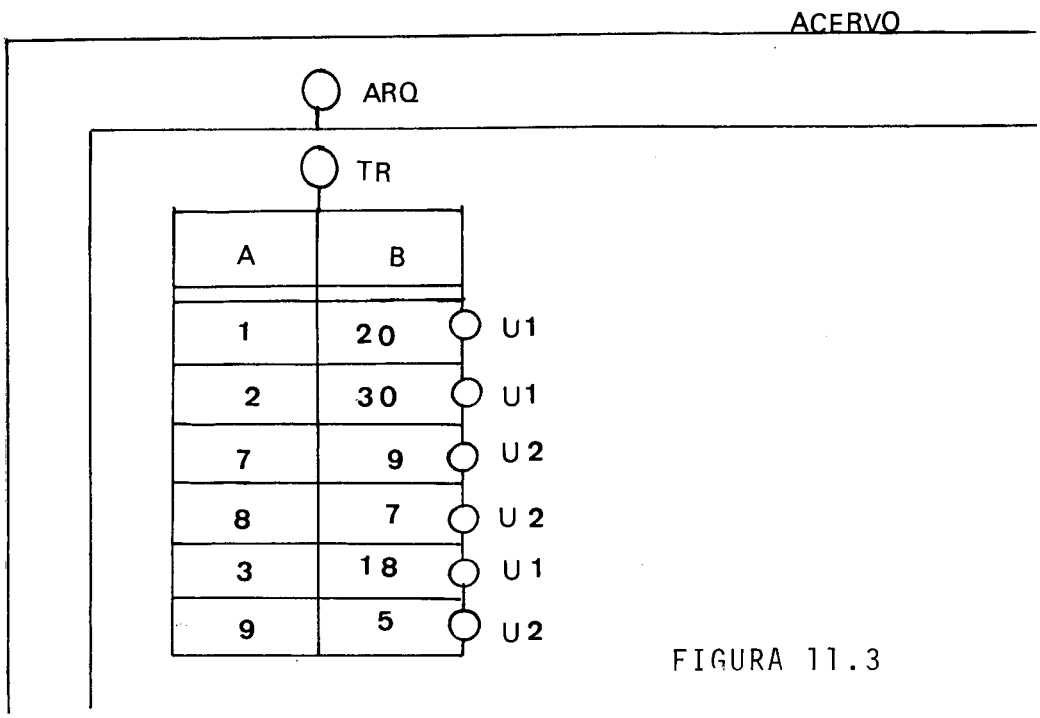


FIGURA 11.3

Ao avaliar as áreas reclamadas por U1 e U2 o sistema verificaria que não existe conflito pois, para U1 teríamos os pontos de tuplas indicados na figura 11.3 por ○ U1, e para U2 os pontos de tuplas indicados por ○ U2. Assim as áreas reclamadas se tornariam áreas protegidas. Ao começar a executar os seus procedimentos U1 e U2, considere que a operação de inserir realizada por U2 seja executada antes da operação de

substituir emitida por U1, teriâmos que quando da avaliação do endereço de ponto da operação de substituir, este endereço endereçaria a tupla inserida por U2, assim seria considerado um erro pelo sistema. Por este aspecto e também por considerar que pontos inseridos por operação de INSERIR, ficam automaticamente incluídos na área protegida pelo usuário que o inseriu, achamos que todos os endereços de pontos devem ser avaliados relativamente a área protegida, como consequência nenhum endereço será considerado fora da área protegida, para as instruções de SUBSTITUIR e MODIFICAR.

Após ser executada a operação desejada, o sistema marca os pontos com a identificação da proteção que permitiu a realização desta operação. Isto porque quando o usuário abandonar a proteção sobre uma parte da área protegida fazendo referência a identificação da proteção a ser abandonada, o sistema tenha conhecimento dos pontos que devem ser liberados. A identificação da proteção é dada quando do estabelecimento de uma proteção por um nome escolhido pelo usuário como:

ESTABELECER PROTEÇÃO XY...

é claro também que uma dada operação pode estar sendo realizada com a permissão de um ou mais estabelecimento de proteção. Neste caso o sistema marca com todos identificadores de proteção, em cada ponto, que permitiram a realização desta operação.

- MÉTODO 2

Após verificar a compatibilidade de um determinado endereço então o sistema executa a operação desejada marcando os pontos referenciados nesta operação com as marcas

<id-prot> para os pedidos que permitiram a realização desta operação. Isto é para pontos que foram lidos devido aos pedidos de leituras $<id_1>,...,<id_n>$, o sistema coloca nesses pontos as marcas $<id_1>,...,<id_n>$, e para os pontos de alteração $<id_1>,...,<id_k>$, o sistema coloca nesses pontos as marcas $<id_1>,...,<id_k>$. Esta marcação é necessária para verificação de conflito, que será visto quando analisarmos os blocos 8, 10 e 11.

- BLOCO 8: CONFLITO?

- MÉTODO 1

A possibilidade de conflito é ocasionada pela ocorrência de uma aglutinação, quando da manutenção das áreas protegidas.

- MÉTODO 2

A possibilidade de conflito é ocasionada quando a alteração realizada não obedece a qualificação apresentada pelos predicados dos pedidos. Por exemplo, dado o pedido de proteção:

```
ESTABELECER PROTEÇÃO <id-prot>
SOBRE {ACERVO.ARQ.TR.(C A > 5)}
 POR PREDICADOS PARA ALTERAÇÃO
  SUBSTITUIR ITEM EM ACERVO.ARQ.TR.(C A > 5).A
                    POR 3
        :
        :
        :
ABANDONAR PROTEÇÃO <id-prot>
```

A alteração realizada desqualificou a tupla alterada pois o componente sob nome A, não é mais maior do que 5.

- BLOCO 9: É ABANDONAR

- MÉTODO 1 e 2

Verifica que pedido de proteção pela utilização de <id-prot>, que o usuário deseja abandonar.

- BLOCO 10: PROVIDÊNCIAS DO SISTEMA

- MÉTODO 1

Ao detetar a aglutinação, o sistema toma as providências já apresentadas no capítulo VII, quando abordamos a regra de correspondência.

- MÉTODO 2

No caso de uma desqualificação a providência que poderia ser realizada pelo sistema é a de interromper a execução do usuário que determinou esta desqualificação.

- BLOCO 11: LIBERAR ÁREA

- MÉTODO 1

Quando da realização de uma liberação os pontos marcados com <id-prot>, são desmarcados e para cada pedido de proteção que estão esperando atendimento têm as áreas reclamadas reavaliadas pelo sistema e verificado se estas áreas reclamadas podem se tornar protegidas.

- MÉTODO 2

Quando da realização de uma liberação os pontos marcados com <id-prot>, são desmarcados e o pedido associado ao identificados <id-prot>, é retirado da tabela que contem todos os pedidos, (figura 11.2). Todos os pedidos não ativos são reavaliados contra todos os pedidos a ativo, para a verificação se alguns dos pedidos NÃO ATIVOS podem ser atendidos em decorrência desta liberação.

Análise Comparativa

O grande problema referente ao método 2 é a avaliação do resultado entre a comparação de dois predicados. Quando estes predicados são muito complicados é muito difícil saber se são os endereços compatíveis ou não, como por exemplo endereços de pontos que usam marcação. Para solucionar este problema é necessário que na definição dos endereços de pontos faça-se a restrição de só permitir expressões booleanas simples do tipo.

1) N Px = < literal de nome >

2) C < literal de nome > $\begin{matrix} > \\ = \\ < \end{matrix}$ < literal numérico>>

e mais os operadores booleanos, $\neg$, $\wedge$, $\vee$

Com estas restrições para as expressões booleanas o sistema pode então sempre decidir se dois endereços estão ou não em conflito. Como uma apresentação geral, vamos supor que o sistema é capaz de avaliar compatibilidade entre dois predicados quaisquer, ficando a cargo do implementador de

finir o conjunto de predicados permitidos.

EXEMPLO: Considere dois pedidos de proteção dados:

U1: ESTABELECER PROTEÇÃO CONFU1 SOBRE
{ACERVO.ARQ.TR.(C A > 10)} POR PREDICADOS
PARA LEITURA

e

U2: ESTABELECER PROTEÇÃO CONFU2 SOBRE
{ACERVO.ARQ.TR.(C B < 5)} POR PREDICADOS
PARA ALTERAÇÃO

e os pedidos realizados quando da representação do acervo mostrado na figura 11.4

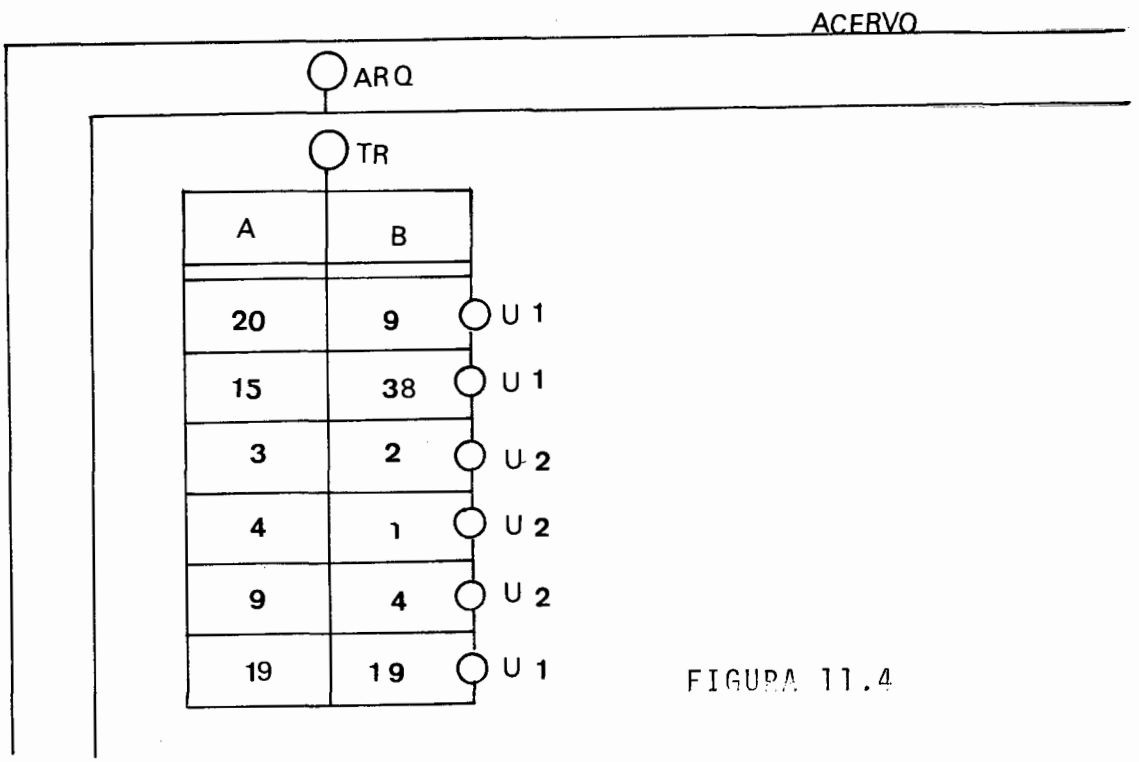


FIGURA 11.4

Os pedidos avaliados por predicados forneceriam ou resultariam em conflito, no entanto podemos verificar que ambos os pedidos poderiam ser atendidos simultaneamente pelo sistema, dado que os pontos presentes no acervo

requeridos para proteção por U1, não tem pontos comuns, com os pontos presentes no acervo requeridos por U2.

Aplicando o método 1, devido aos enumeros acessos que se realizam na base de dados, para o cálculo das áreas reclamadas e reavaliação sempre após uma liberação das áreas reclamadas, o esforço computacional é várias vezes superior que pela aplicação do método 2.

Como observação final, neste trabalho não foi analisada a possibilidade da aplicação dos dois métodos conjuntamente, embora tenhamos utilizado nos exemplos a dinstinção entre os pedidos indicando o termo POR PREDICADOS.

CAPÍTULO XII

CONSIDERAÇÕES ADICIONAIS:EXTENSÃO PARA NÍVEIS DE LEITURA

12.1. Níveis de Leitura

Com a finalidade do aumento da concorrência o usuário pode especificar em seu pedido de proteção para leitura, três níveis. Se o usuário especificar em seu pedido de proteção para leitura o nível 1, então a ele será garantido que, nenhum outro usuário poderá efetuar modificações nos pontos protegidos por este pedido de proteção. Se o usuário especificar em seu pedido de proteção para leitura o nível-2, então outro usuário poderá efetuar modificações nos pontos relativos a este pedido, porém o sistema garantirá ao usuário que realizou o pedido de nível 2, que as leituras efetuadas serão sempre leituras em construções integras. Se o usuário especificar o nível 3 de proteção para leitura, outro usuário poderá realizar modificações nos pontos relativos a este pedido, e ao usuário que requerer o nível 3 para leitura, o sistema não dará garantia de que a leitura efetuada nas construções nos pontos referenciados neste pedido serão integras.

O aumento de concorrência decorre do fato, ou que para os níveis 2 e 3 o sistema não mantem os pontos especificados para estes níveis de leitura protegidos contra usuários alteradores, podendo assim estes usuários alteradores utilizarem pontos comuns com usuários leitores de níveis 2 e 3. A justificativa para a existência dos níveis 2 e 3, é baseada no fato de que, o usuário especificando estes níveis podem requerer um grande número de pontos e tê-los imediatamente. Por

exemplo para efetuar estatísticas no acervo onde resultados exatos em cada ponto não são necessários.

Daremos a seguir qual o tratamento realizado por cada método, quando da especificação destes níveis de leitura.

NÍVEL-1

- MÉTODO-1

Para pedidos de proteção para leitura de nível 1, o sistema realiza o cálculo da área reclamada e marca os pontos com o identificador do pedido de proteção. Estes pontos serão mantidos protegidos contra alteradores até que haja a liberação.

- MÉTODO-2

Para pedidos de proteção para leitura de nível-1, o sistema inserirá na tabela de pedidos e sempre verificará a compatibilidade deste pedido contra alteradores. Após ser garantido pelo sistema o começo das operações devido a este pedido. Todas as instruções dentro do bloco de proteção para este pedido que referenciam pontos deste pedido, nestes pontos, será colocada a marca do identificador de proteção, que será mantida até que haja a liberação destes pontos.

NIVEL - 2

- MÉTODO 1

Ao encontrar um pedido de proteção para leitura de nível 2, o sistema calcula a área reclamada e para cada ponto desta área reclamada colocar a marca do identificador do pedido, para que quando o

usuário que mantem este pedido desejar realizar um acesso, o sistema possa garantir que as construções nos pontos com este identificador estejam integras para que a leitura possa ser efetuada. Porém para efeitos de verificação de conflito a área reclamada associada a um pedido de leitura nível 2 não será considerada. Neste nível de leitura os pontos são considerados como pontos de uma área livre.

- MÉTODO 2

Será encarado pelo sistema como dito para o nível 1, com a exceção de que os pedidos de leituras de nível 2 não serão inseridos na tabela para verificação de conflitos No entanto os pontos referenciados nestes pedidos levam a marco do identificador de pedido para que o sistema possa garantir uma leitura integra.

NÍVEL 3

Para efetuar leitura de nível 3, o usuário não indica no seu pedido nenhum endereço de ponto pois a ele é permitido acessar qualquer ponto no acervo. Sendo assim nenhuma marca é colocada nos pontos referenciados nas instruções realizadas por este usuário. Seu pedido não será considerado em nenhum caso para efeitos de verificação de conflito.

12.2. DEAD LOCK

O problema de tratamento de Dead lock, está ligado ao aspecto de se permitir ou não encaixamentos entre blocos de proteção.Se for permitido ter-se blocos de proteção en-

caixados, significa que um usuário pode mantendo uma área protegida em seu poder, requisitar mais pontos para proteção como por exemplo podemos ver através do exemplo 12.1

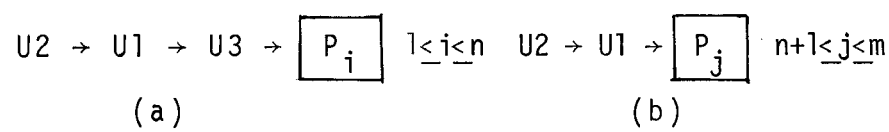
EXEMPLO 12.1: Considere o procedimento mostrado abaixo para um usuário U1.

```
U1: ESTABELECER PROTEÇÃO PRIM
    SOBRE{<end_1>,...,<end_n>} PARA ALTERAR
    <instrição-1>
         .
         .
         .
    <instrução-n>
     ESTABELECER PROTEÇÃO SEG
     SOBRE{<end_m>,...,<end_e>} PARA LER
     <instrução-m+1>
          .
          .
          .
     <instrução-m>
     ABANDONAR PROTEÇÃO PRIM
     <instrução-m+1>
          .
          .
          .
     <instrução-k>
     ABANDONAR PROTEÇÃO SEG
```

Neste exemplo verificamos que se os pontos a serem protegidos devido ao segundo pedido de proteção, não estiverem disponíveis para uma proteção, o usuário U1, deverá esperar até que o sistema, possa atender o seu pedido de proteção. Mas por já possuir uma área protegida devido as execuções das instruções até antes do segundo pedido de proteção e como o u-

Para o caso (a) da figura 12.1, temos que o tratamento de dead lock a ser utilizado é o de detenção e recuperação , e para o caso (b), o sistema não necessita tomar providências quanto ao problema de dead lock.

Para o caso (b), temos que quando utilizado o método 1, para não ter-se que avaliar muitas vezes as áreas reclamadas derivadas dos pedidos de proteção para os usuários que estão a espera do atendimento, o sistema pode quando da primeira avaliação das áreas reclamadas e da verificação da existência de conflito, realizar uma fila nos pontos participantes dessas áreas reclamadas a espera de uma liberação desses pontos por outros usuários. Como por exemplo, se tivermos os pontos: P1, ..., Pn, participantes das áreas reclamadas por U1 e U2, e mantidos protegidos por um usuário U3, e os pontos P_{n+1},..., P_m participantes também das áreas reclamadas de U1 e U2, e que não estão protegidos, teríamos o apresentado na figura 12.2.



(a) (b)

FIGURA 12.2

Para os casos (a) e (b) diríamos que U3 e U1 possuem os pontos P_i e P_j respectivamente. Com este tratamento de filas em pontos, é possível a ocorrência de dead lock entre dois usuários que possuem pontos e sendo estes pontos não protegidos. Para solução deste dead lock indicamos as referências |4|,|11| e |12|.

suário U1, não realizou a liberação dessa área protegida, então este usuário deverá esperar o atendimento do segunto pedido de proteção, com a sua área protegida não liberada, isto é, ainda em seu poder. Esse fato permite que ocorra uma situação de dead lock, na fase de manipulação, que deve ser resolvida através de um procedimento que detete e solucione dead lock. Este procedimento, quando detetado um dead lock, então, deve escolher um ou mais usuários e cancelar as proteções por eles realizadas e assim liberando a pontos para eles protegidos.

Caso se deseje que nunca ocorra dead lock na fase de manipulação então a solução é simplesmente não permitir encaixamento entre blocos de proteção. Com esta providência sendo exigida, nenhum usuário pode requerer uma proteção para uma determinada área, antes que libere a área protegida que está em seu poder.

Temos então esquemáticamente o mostrado na figura 12.1, onde as linhas representam blocos de proteção, e que tipo de tratamento deve ser realizado pelo sistema

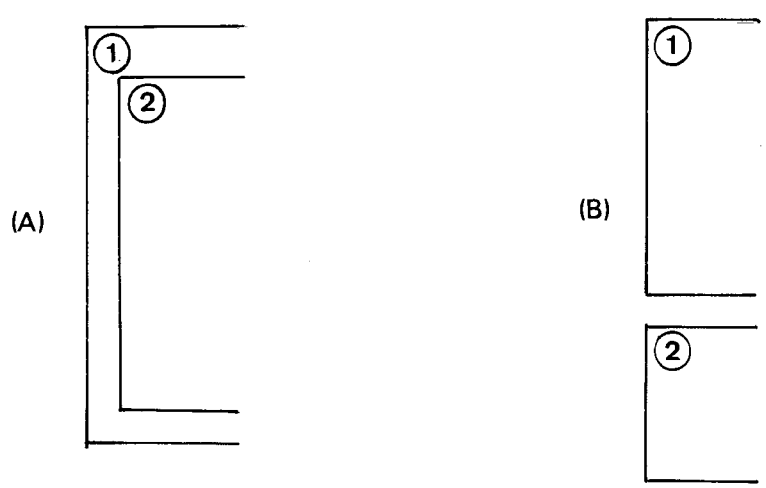


FIGURA 12.1

CAPÍTULO XIII

CONCLUSÕES E ASPECTOS ADICIONAIS

Enumeraremos neste capítulo as conclusões e alguns aspectos adicionais concernentes ao trabalho apresentado.

13.1. Ambos os métodos apresentados podem ser aplicados para qualquer estrutura de informação permitida pelos conceitos definidos em IMC.

13.2. Não foi analisado neste trabalho, quais seriam as consequências devido as representações das informações relativamente a aplicação dos métodos apresentados. Isto é, aspectos implementacionais.

13.3. Os conceitos e métodos apresentados, para o tratamento de proteção servem também para aplicação de tratamento de autorização.(Quanto à compatibilidade de pontos endereçados nas operações com pontos aos quais foi atribuida uma certa autorização).

13.4. Não analisamos a possibilidade de um usuário realizar o que chamamos de redução de proteção. Isto é, se um determinado usuário estabelece uma proteção para alterar e no interior do bloco de proteção ele deseja para os pontos protegidos para alteração devido ao seu pedido de proteção, que o sistema reduza a proteção nesses pontos para uma proteção para leitura.

13.5. Não foi apresentado neste trabalho, o que inicialmente

estava previsto, a aplicação dos conceitos apresentados para a modelagem do tratamento de proteção realizados por sistemas encontrados na literatura como os sistemas R, CODASYL e INGRES.

ANEXO I

<termo de proteção>::=

<id-proteção> SOBRE {<termo de conjunto de pontos>}[POR PREDICADOS]

PARA { LER { [NÍVEL-1] / [NÍVEL-2] / [NÍVEL-3] } / ALTERAR { [NÍVEL-1] / [NÍVEL-2] / [NÍVEL-3] } } [COM CONEXÃO]

<estabelecer proteção>::=

ESTABELECER PROTEÇÃO <termo de proteção>

E PARA <termo de proteção>

<considerar transação> ::=

CONSIDERAR TRANSAÇÃO

<inserir construção>::=

INSERIR <obter construção> EM <termo de conjunto de pontos>

<retirar construção>::=

RETIRAR [<termo de tipificação>] DE <termo de conjunto de pontos>

<substituir construção>::=

SUBSTITUIR [<termo de tipificação>]EM <termo de conjunto de pontos>
POR < obter construção >

<modificar nominação>::=

MODIFICAR [<termo de tipificação>]EM <termo de conjunto de pontos>
COM <obter nominação>

<substituir nome>::=

SUBSTITUIR NOME EM <end ponto> POR <obter nome>

<termo de conjunto de pontos>::=

((<termo de conjunto de pontos>))

1<end ponto>

2<termo de conjunto de pontos>{U|/| } <termo de conjunto de pontos>

<obter construção>::=

|OBTER| |<termo de tipificação>|<termo de construção>

<termo de construção>::=

1 C <end ponto>

2 COLEC <end ponto>

3 CALC <id cálculo> SOBRE <end ponto>

4 <obter real> {+|-|*|/|↑} <obter real>

5 CARD <obter coleção>

6 ESTREIT <obter tab>{DE|PARA} atributo ,,,

7 COMPOR <bloco de composição>

8 <termo elementar de valbool>

9 <obter valbool>

10 <obter valbool>{∧|∨|↔|→ <obter valbool>

11 PARA CADA <end ponto> ((<obter valbool>))

12 EXIST <end ponto>

<bloco de composição>::=

((|<obter construção>|...))|

((|<obter nome>:=<obter construção|...))

<termo elementar de valbool>::=

1 <obter construção> = <obter construção>

2 <obter coleção> {⊂|⊆|⊃|⊇} <obter coleção>

3 <obter construção> {=|<|≤|>|≥} <obter construção>

<lit valbool>::=

VERDADEIRO | FALSO

<verbete de coerência>::=

<d tipo> : TIPO DE <d pretipo><verbete de consistência>

<verbete de consistência>::=

TAL QUE

COMPOS <termo de composição>

[CONEX <termo de conexão>]

<termo de composição>::=

{<lit nome>→< d tipo>|}$_{,,,}$|<d tipo>

<termo de conexão>::=

[AJUSTAR]<obter valbool>[AJUSTAR] Λ<termo de conexão>

<abandonar proteção>::=

ABANDONAR { PROTEÇÃO <id-pro> | TODAS AS PROTEÇÕES }

REFERÊNCIAS BIBLIOGRÁFICAS

|1| - G.RICHTER - "O Sentido E O Valor Do Banco de Dados", Dados e Idéias, p.2-14, 1977.

|2| - R.DURCHHOLZ,G.RICHTER - "Information Management Concepts (IMC) For Use With DBMS Interfaces", Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems, Freudnstadt, Germany, p.49-72, 1976.

|3| - J.N. GRAY, R.A.LORIE, G.R.PUTZOLU, I.L.TRAIGER - "Granularity of Locks And Degrees Of Consistency In A Shared Data Base", Proceedings of the IFIP of the Working Conference on Modelling in Data Base Management Systems, Freundenstadt, Germany, p.365-394, 1976.

|4| - G.GARDARIN, S.SPACCAPITRA - "Integrity Of Data Bases: A General Lockout Algorithm With Deadlock Avoidance", Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems, Freundenstadt, Germany, p.395-412, 1976.

|5| - R.W.ENGLES - "Currency And Concurrency in COBOL Data Base Facility", Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems, Freundenstadt, Germany, p.339-364, 1976.

|6| - R.DURCHHOLZ, G.RICHTER - "Concepts For Data Base Management Systems", Proceedings of the IFIP Working Conference on Data Base Management, Cargèse, Corsica, p.97-122, 1974.

|7| - G.C.EVEREST - "Concurrent Update Control And Data Base Integrity", Proceedings of the IFIP Working Conference on Data Base Manegement, Cargèse, Corsica, p.241-270, 1974.

|8| - G.RICHTER - "On The Relationship Between Information And Data", Proceeding 5th Informatik Symposium, IBM Germany , Bad Homburg v.d.H, p.21-43, 1975.

|9| - M.M.ASTRAHAN, D.D. CHAMBERLIN, W.F.KING, I.L.TRAIGER - "System R - A Relational Data Base Management System", Proceedings 5th Informatik Symposium, IBM Germany, Bad Homburg v.d.H., p.139-148, 1975.

|10|- S.C. KLEENE - "Introduction to Matemathematics", Van Nostrand Company, p.204, 1952.

|11|- R.BAYER - "On The Integrity Of Data Bases And Resource Locking", Proceedings 5th Informatik Symposium, IBM Germany, Bad Homburg v.d.H, p.339-361.

|12|- D.D.CHAMBERLIN, R.F.BOYCE, I.L.TRAIGER - "A Deadlock - Free Scheme For Resource Locking In A Data Base Enviroment", Information Proceedings, p.340-343, 1974.

|13|- K.P.ESWARAN, J.N.GRAY, R.A.LORIE, I.L.TRAIGER - "On The Notions Of Consistency and Predicate Locks in a Data Base System", IBM Research Report RI 1487, Dec.30, 1974.

|14|- D.B.LOMET - "A Pratical Deadlock Avoidance Algorithm For Data Base Systems", SIGMOD Proceedings, Toronto, August, p.122-127, 1977.

|15| - R.C.HOLT - "Comments On Prevention Of System Deadlocks", Comm.ACM 14,1(Jan.1971), p.36-38.

|16| - E.G.COFFMAN, M.J.ELPHICK, A.SHOSHANI - "System Deadlocks", Computing Surveys, Vol.3, Nº 2, JUNE. p.67-78, 1971.

|17| - M.STONEBRAKER - "Implementation Of Integrity Constraints And Views By Query Modification", Proc.ACM SIGMOD Conf., San Jose, Calif., May 1975, p.65-78.

|18| - M.M.ASTRAHAN, M.W.BLASGEN, D.D.CHAMBERLIN, K.P.ESWARAN, J.N.GRAY, P.P.GRIFFITHS, W.F.KING, R.A.LORIE, P.R. NcJONES, J.W.MEHL, G.R.PUTZOLU, I.L.TRAIGER,B.W.WADE, V.WATSON - "System R:Relational Approach to Database Management", ACM Transacation on Database Systems, Vol. I, Nº 2, June 1976, p.97-137.

|19| - M.STONEBRAKER, E.WONG, P.KREPS, G.HELD - "The Design And Implementation Of INGRES", ACM Transaction on Database Systems, vol.1, Nº 3, September 1976, p.189-222.

|20| - CADASYL - DDL Journal Of Development, 1973.

|21| - CODASYL - DML Journal of Development, 1976.

|22| - G.RICHTER, J.CUNHA PEREIRA, J.M.V.CASTILHO - Projeto Miniban : Relatório Final Da Segunda Etapa, Parte A Parte B, Rio de Janeiro, CNPq/DIBIBRÃS/GMD/UFRGS, Abril 1978.

|23| - G.RICHTER,J.M.V.CASTILHO - "Uma Interface Para Sistemas de Informação: LOBAN - Linguagem de Operação de Banco de Dados em Anais do 11º CNPD,Outubro 1978,RJ,p.347-354.