PUC

Série: Monografias em Ciência da Computação
Nº 14/77

O MAPEAMENTO DA ESPECIFICAÇÃO CONCEITUAL EM BANCO DE DADOS GERENCIADO PELO TOTAL

POR

Rubens N. Melo
e
Leonardo Lellis

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente 225 — ZC 19
Rio de Janeiro — Brasil

Série: Monografias em Ciência da Computação
Nº 14/77

O MAPEAMENTO DA ESPECIFICAÇÃO CONCEITUAL EM BANCO DE DADOS GERENCIADO PELO TOTAL*

Por

Rubens N. Melo

e

Leonardo Lellis

Editor da Série: Michael F. Challis Agosto 1977

* Trabalho parcialmente patrocinado pela FINEP

Para obter cópias dirija-se a:

Rosane T. L. Castilho
Chefe Setor de Documentação e Informação
Deptº de Informática - PUC/RJ
Rua Marques de São Vicente, 209 - Gávea
20.000 - Rio de Janeiro - RJ - BRASIL

RESUMO:

A partir da especificação conceitual de um sistema de informações apoiado em banco de dados, feita através da linguagem PSL, aborda-se o problema de obtenção do esquema interno do banco de dados do sistema usando o DBMS TOTAL. Inicialmente as idéias básicas da especificação conceitual de sistemas de informação são apresentadas. Em seguida algumas características relevantes do sistema TOTAL são abordadas e, finalmente, é sugerida uma maneira de mapear a especificação conceitual do banco de dados do sistema para um esquema interna usando TOTAL.

PALAVRA CHAVE:

Especificação conceitual, sistemas de informação, esquema interno, DBMS TOTAL.

ABSTRACT:

The object of this paper is to propose an approach to obtain the internal schema of the data base of a system which has been conceptually specified in PSL using the TOTAL DBMS. First, the basic ideas of the conceptual specification of an information system are presented. Then the relevant characteristics of the TOTAL system are described and finally a method is suggested to map a conceptual specification of the system's data base to an internal schema using TOTAL.

KEY WORDS:

Conceptual specification, information system, internal schema, TOTAL DBMS.

SUMÁRIO

1. INTRODUÇÃO

Este trabalho aborda o problema da obtenção do esquema interno do Banco de Dados, gerenciado pelo TOTAL, de um sistema de informação, a partir de sua especificação conceitual feita em PSL [6,7].

Um resumo das idéias básicas de Especificação Conceitual de Sistema de Informação apoiados em Bancos de Dados [3] é descrito no capítulo 2. Neste trabalho a parte de especificação correspondente a descrição conceitual dos dados é enfatizada e o quadro de conceitos utilizados para esta descrição é apresentado na seção 2.2.

O uso da PSL como linguagem de especificação conceitual foi discutido em [3]. Supondo que a especificação tenha sido feita em PSL (ou similar) e que sua forma objeto tenha sido armazenada num Banco de Dados, como ocorre em PSA, este trabalho sugere no capítulo 4 um método para mapear a descrição conceitual dos dados do sistema para um esquema interno de um Banco de Dados gerenciado pelo TOTAL.

Para ser mais auto-suficiente este trabalho apresenta também no capítulo 3 um resumo sobre as estruturas de arquivos do sistema do TOTAL e suas facilidades para a definição do esquema interno de Banco de Dados.

2 - INTRODUÇÃO A ESPECIFICAÇÃO CONCEITUAL PARA SISTEMAS DE INFORMAÇÃO

2.1 - CONCEITOS BÁSICOS

Os sistemas de informação apoiados em banco de dados representam o resultado de uma transição de enfoque na área de sistemas de informação. Este novo enfoque pode ser caracterizado por dois pontos fundamentais: a independência de dados e a separação arquivos-programas [1,2] .

Uma forma de caracterização da idéia de sistemas de banco de dados integrados é que este seja uma fonte de dados estruturados para modelar um sistema real e integrados para satisfazer os requisitos de todas as aplicações [3] . Como resultado desta integração temos uma redução da redundância de dados e um controle melhor dos dados evitando a inconsistencia de dados.

Este novo enfoque implica em uma necessidade de integração de análise, especificação e projeto dos sistemas de informação apoiados em bancos de dados. Desta necessidade de integração surge a idéia da especificação conceitual de sistemas de informação apoiados em banco de dados.

A especificação conceitual pode tornar possível a transformação sistemática dos requisitos dos usuários em problemas para técnicos em processamentos de dados e possibilita ver o banco de dados como uma fonte de informações sem tomar conhecimento como a informação é armazenada nos meios físicos.

Desta forma o usuário pode utilizar o banco de dados num modelo orientado para a realidade, em termos que são compatíveis com a sua própria concepção do mundo real.

A especificação conceitual ajuda tanto na especificação da estrutura semântica quanto na documentação da formalização conceitual, facilitando a comunicação entre o especificador e o implementador.

Esta especificação deve descrever o conjunto de informações que representa as entidades, suas propriedades e relações e também deve conter informações sobre as mudanças de estado do banco de dados ao longo do tempo.

Desta forma o esquema proposto em [3] para a especificação conceitual de sistemas de informação apoiados em bancos de dados deve consistir de:

1 - ESQUEMA CONCEITUAL DO BANCO DE DADOS

1.1 - DESCRIÇÃO CONCEITUAL DOS DADOS

1.2 - DESCRIÇÃO DAS RESTRIÇÕES DE INTEGRIDADE

2 - DESCRIÇÃO DA EVOLUÇÃO DO BANCO DE DADOS

Os objetivos principais deste esquema para especificação conceitual do sistema são os seguintes: ser um modelo informacional do sistema real, servir de referência tanto para a análise quanto para implementação e servir de documentação do sistema.

2.2 - UM MODELO PARA ESPECIFICAÇÃO CONCEITUAL

Analisando os aspectos do mundo real que normalmente são descritos e utilizados em sistemas de informação concluimos que "uma certa classe de aplicações" pode ser descrita através de ENTIDADES que possuem propriedades elementares que são denominadas ELEMENTOS e de RELAÇÕES entre estas entidades.

Desta forma os conceitos que utilizaremos na especificação conceitual são:

ENTIDADES - são objetos reais ou abstratos que devem ser tratados pelo sistema de informação. Exemplo: um "projeto" de uma determinada empresa.

ELEMENTOS - são propriedades que descrevem as entidades do sistema. Exemplo: entre as propriedades que descrevem um"projeto",poderíamos ter: "número do projeto", "descrição", "valor global".

RELAÇÕES - são conecções de interesse entre as entidades, podendo existir elementos associados a estas relações. Exemplo: entre a entidade "agente financeiro" e a entidade "projeto" existe a relação "financia". Um elemento que pode estar associado a esta relação seria o "valor do financiamento".

Existem dois aspectos principais que devem ser considerados quando tratamos uma relação, são eles:

i) conectividade - indica quantas entidades de cada tipo podem estar associadas em uma relação. Exemplo de conectividade: "many-to-many". Esta conectividade ocorre na relação "financia" entre "agentes financeiros" e "projetos"; isto quer dizer que um "agente financeiro " pode financiar mais de um "projeto", e por sua vez um "projeto" pode ser financiado por mais de um "agente financeiro".

ii)N-ariedade - indica quantos tipos de entidades estão envolvidos em uma relação. Exemplo: a relação "financia" do exemplo anterior tem N-ariedade 2, pois envolve duas entidades.

Quanto aos tipos de entidades que podem estar envolvidos em uma relação devemos considerar três pontos importantes:

i)uma relação pode ser feita entre entidades de tipos diferentes.Exemplo: a relação "financia" envolve duas entidades de tipos diferentes.

ii)uma relação pode ser feita entre entidades de mesmo tipo.Exemplo: a relação "pré-requisito" envolve duas entidades do tipo "disciplina".

iii)pode existir mais de um tipo de relação entre duas determinadas entidades. Exemplo: entre as entidades "agente financeiro" e"projetos" além da relação "financia" poderíamos ter a relação "patrocina"a relação financia envolve "agentes financeiros" que devem ser reembolsados de seu financiamento e a relação "patrocina" envolve "agentes financeiros " que não serão reembolsados.

3 - O DBMS TOTAL

3.1 - ORGANIZAÇÃO DE ARQUIVOS

O TOTAL [4] é um sistema de gerência de banco de dados que utiliza dois tipos de arquivos e que permite a interligação entre todos os arquivos. Desta forma o banco de dados pode ser visto como uma estrutura "network".

Cabe observar, conforme veremos a seguir, que para existir a interligação entre todos os arquivos, alguns tipos de ligação devem ser mantidos por programas do usuário, em complementação aos que são mantidos automaticamente pelo próprio sistema TOTAL.

Os dois tipos de arquivos que são utilizados no TOTAL são:

- arquivos MESTRES
- arquivos VARIÁVEIS

Arquivos Mestres

São arquivos armazenados em dispositivos de acesso direto, pré-formatados através do programa FORMAT, onde os registros são manipulados (inseridos, lidos, regravados e deletados) através de randomização de uma chave primária.

Arquivos Variáveis

São arquivos armazenados em dispositivos de acesso direto, pré-formatados, onde os registros formam listas duplamente encadeadas. Os ponteiros para acesso ao começo ou fim da lista são obtidos pela leitura implícita de um registro mestre associado a esta lista. Existe também a possibilidade de acesso direto a um registro de um arquivo variável caso seja conhecida a posição relativa ao início do arquivo.

RELACIONAMENTO ENTRE ARQUIVOS

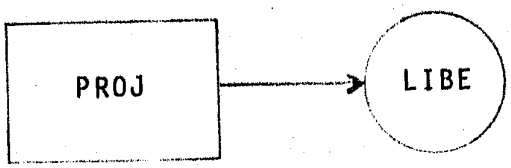
Relacionamentos primários

São relacionamentos mantidos pelo próprio sistema TOTAL.

Com este tipo de relacionamento podemos ligar arquivos de tipos diferentes, isto é, podemos:

i) relacionar um mestre a um variável

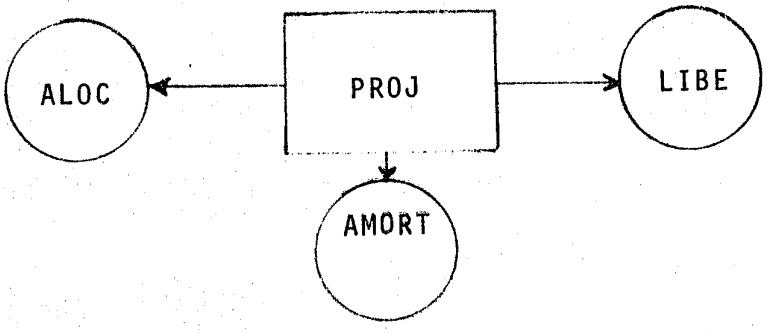
exemplo:



Neste exemplo temos um arquivo mestre de "projetos" ligado a um arquivo variável de "liberações de verbas para este projeto". Desta forma a cada registro mestre temos associada uma cadeia de registros variáveis podendo conter, zero, um ou mais registros de "liberações" para aquele "projeto". Dois "projetos" distintos podem ter cadeias de tamanho igual ou diferente.

ii) relacionar um mestre a mais de um variável

exemplo:



Neste exemplo temos um arquivo mestre de "projetos" ligado a três variáveis, um contendo as "liberações dos projetos", ou

outro contendo as "amortizações dos projetos" e um terceiro contendo as "alocações de recursos humanos para os projetos". Não existe nenhuma relação entre o número de registros nas três cadeias referentes a um determinado "projeto", isto é, um determinado "projeto" pode ter uma cadeia de zero ou mais "liberações", uma cadeia de zero ou mais "amortizações" e finalmente uma cadeia de zero ou mais "alocações de recursos" sendo que o número de registros em cada cadeia pode ser igual ou diferente das demais cadeias.

iii) relacionar um variável a mais de um mestre

Neste exemplo temos um arquivo variável contendo o "histórico de cada empregado" e o "histórico de cada cargo" numa determinada empresa, isto é, para cada "empregado" temos uma cadeia no arquivo variável "histórico", contendo um registro para cada cargo que o empregado ocupou na empresa. Por outro lado para cada cargo que existe na empresa temos uma cadeia no arquivo variável histórico, contendo um registro para cada empregado que ocupou aquele cargo. É importante observar que, um determinado registro do arquivo variável, não precisa necessariamente estar ligado a uma cadeia do arquivo "empregado" e a uma cadeia do arquivo "cargos". Desta forma poderíamos ter um funcionário que tivesse ocupado um cargo ainda não cadastrado e ele teria esta ocupação registrada na sua cadeia do arquivo variável "histórico" sem que este registro fizesse parte de alguma cadeia pertencente a um registro do arquivo "cargos".

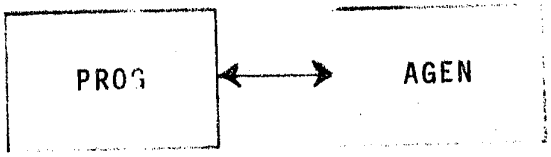
Relacionamentos secundários

São relacionamentos mantidos por programa

Com este tipo de relacionamento podemos ligar arquivos de mesmo tipo, isto é, podemos:

i) relacionar um mestre a um mestre

exemplo:

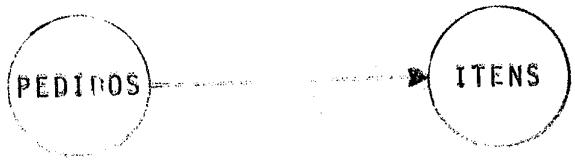


Neste exemplo temos um arquivo mestre de "projetos" associado a um arquivo mestre de "agentes financeiros" num relacionamento um-para-um.

Para este tipo de ligação colocamos no registro do primeiro mestre a chave primária do registro do segundo mestre e nesse a chave primária do registro do primeiro mestre.

Cabe observar que, com este tipo de relacionamento, podemos ligar um arquivo mestre a ele mesmo. Um exemplo deste tipo de relacionamento seria o caso em que no registro do arquivo de "empregados" de uma certa empresa existisse um campo "chefe" que contivesse a chave primária do registro do chefe deste empregado.

ii) relacionar um variável a um variável



Neste exemplo temos um arquivo variável de "pedidos" associado a um arquivo variável de "itens do pedido".

3.2 - ESQUEMA DE DEFINIÇÃO

Na conceituação apresentada pela CODASYL [5]. O esquema de um banco de dados consiste de uma completa descrição do banco de dados. Este esquema inclui o nome e a descrição de todos os arquivos do banco, assim como a descrição do relacionamento entre estes arquivos. Nesta mesma conceituação encontramos o conceito de sub-esquema, que consiste da descrição do arquivos utilizados por um ou mais programas especificos.

Analisando o DBMS TOTAL concluimos que êle possui o conceito de esquema, mas entretanto não possui conceito de sub-esquema. A seguir damos uma descrição geral da implementação do conceito de esquema adotada pelo TOTAL. (detalhes desta implementação são encontrados em [4]).

Uma das vantagens da utilização do conceito de esquema é diminuir o esforço requerido no desenvolvimento de uma aplicação tirando do programador a tarefa de descrever os arquivos utilizados em sua aplicação.

Para isto a descrição lógica e física das características do banco de dados é feita por um programa separado dos programas de aplicação. Este programa, quando executado, produz um módulo que contêm a descrição do banco de dados. Este módulo é armazenado e os programas do usuário que necessitem utilizar o banco fazem apenas uma referência a este módulo e com isto o sistema passa a conhecer a descrição dos arquivos a serem utilizados pelo programa de aplicação.

A descrição do banco é feita através de um "programa" escrito em linguagem própria, a DATA BASE DEFINITION LANGUAGE (DBDL). Esta descrição é compilada pelo programa DBGEN que analisa os comandos escritos na DBDL. Caso exista algum erro nesta definição o programa emite mensagens acusando; caso contrário o programa gera o DATA BASE DESCRIPTOR MODULE (DBMOD) (o esquema). Este módulo será armazenado em disco magnético para depois ser referenciado pelos comandos da DATA MANAGEMENT LANGUAGE (DML) dos programas de aplicação.

O "programa" de definição do banco de dados possui quatro partes:

i) o prologo,

ii) a seção de definição dos arquivos mestres,

iii) a seção de definição dos arquivos variáveis

iv) o epílogo.

As seções de definição dos arquivos mestres e variáveis, por sua vez, são subdivididas em sub-seções que fazem uma definição lógica e física de cada arquivo.

O"programa" inicia com o comando "BEGIN-DATA-BASE-DEFINITION":, que avisa o sistema que neste ponto vai começar a definição de um banco de dados. Seguindo este comando do prólogo ficam comandos que são de caráter geral, são eles: o comando de especificação do nome do banco de dados, o comando de especificação do tamanho do maior registro a ser gravado no arquivo " log" e o comando de especificação dos nomes das areas de entrada e saída a serem utilizadas durante o processamento.

A seção seguinte contém a definição de todos os arquivos mestre do banco de dados. Para cada arquivo mestre deve existir uma sub-seção que inicia com o comando "BEGIN-MASTER-DATA-SET:"; seguindo este comando estão os comandos que especificam o nome do arquivo mestre e o nome da área de entrada e saída a ser utilizada durante o processamento.

A seguir, iniciando pelo comando "MASTER-DATA",estão os comandos que fazem a descrição lógica do registro do arquivo mestre. Estes comandos especificam o tamanho em bytes do campo "root", o nome e tamanho do campo que será utilizado como chave primária do arquivo, os "linkpaths" e finalmente os campos de dados(DATA-ELEMENTS). Finalizando a descrição do registro lógico esta o comando "END-DATA".

A seguir estão os comandos que fazem a descrição física do arquivo mestre. Estes comandos especificam o tipo de unidade de armazenamento, o número total de registros no arquivo, o tamanho em bytes de cada registro lógico, o número total de trilhas que serão utilizadas pelo arquivo e finalmente o número de blocos por trilhas

Para indicar o fim da definição de um arquivo mestre , após os comandos que fazem a descrição física, está o comando "END-MASTER-DATA-SET":. Conforme dissemos anteriormente, se existirem outros arquivos mestres, para cada um deles se repitirá o ciclo descrito acima.

A seção seguinte contém a definição de todos os arquivos variáveis do banco de dados. Da mesma forma que na seção de definição dos mestres, para cada arquivo variável deve existir uma sub-seção que inicia com o comando "BEGIN-VARIABLE-ENTRY-DATA-SET", seguindo este comando estão os comandos que especificam o nome do arquivo variável e o nome da área de entrada e saída a

a ser utilizada durante o processamento.

A seguir, iniciando pelo comando "BASE-DATA", estão os comandos que fazem a descrição lógica do registro do arquivo variável. Estes comandos especificam o nome e tamanho da chave de cada mestre associado a este variável, os "linkpaths" desses mestres para este variável e os campos de dados (DATA-ELEMENTS). Neste ponto existe uma diferença básica entre definição lógica de um registro mestre e de um registro variável. Enquanto para o mestre só pode existir uma formatação lógica para todos os registros do arquivo, para o variável podem existir múltiplos formatos lógicos (todos de mesmo tamanho físico). Desta forma serão definidos nesta seção os deversos formatos que poderão ser adotados. Finalizando a descrição do registro lógico está o comando "END-DATA".

A seguir da mesma forma que para o arquivo mestre estão os comandos que fazem a descrição física do arquivo variável . Além das informações já descritas para o mestre existirá também uma informação de carga máxima do cilindro ("cylinder load limit")

Para indicar o fim da definição de um arquivo variável, após os comandos que fazem a descrição física, está o comando "END-VARIABLE-DATA-SET". Conforme dissemos anteriormente para cada arquivo variável se repetirá o ciclo de instruções acima.

O"programa" de definição do banco de dados termina com epílogo, que consiste do comando "END-DATA-BASE-GENERATION".

Como resultado da compilação do "programa" de definição do banco de dados temos a criação do DBMOD (o esquema) que será

será armazenado em memória secundária e quando um programa de aplicação for utilizar o banco de dados basta fazer uma referência ao nome do módulo e a definição lógica e física dos arquivos passa a ser conhecida pelo sistema (através da carga do respectivo DBMOD).

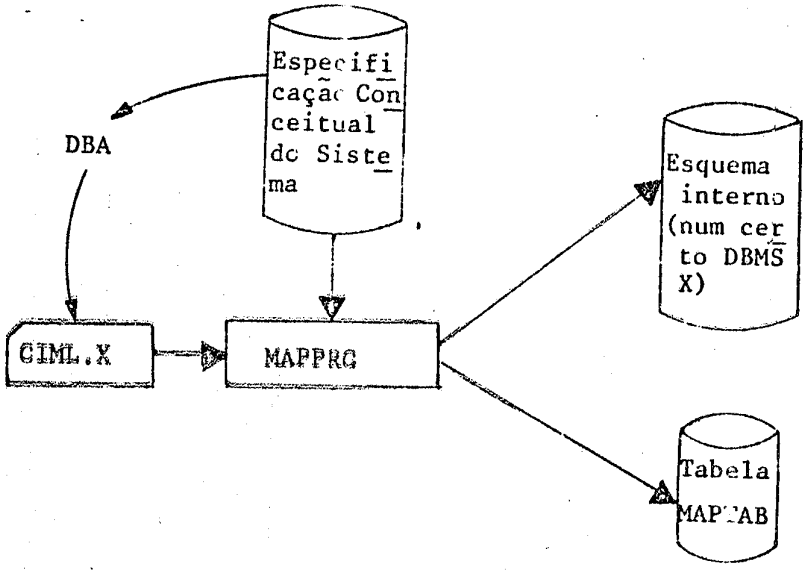
4 - O MAPEAMENTO DA ESPECIFICAÇÃO CONCEITUAL PARA UM ESQUEMA INTERNO DO B.D USANDO O DBMS TOTAL

No capítulo dois apresentamos um modelo para especificação conceitual. Em [3] a linguagem PSL (Problem,Statement Language) do sistema PSL/PSA [6,7] é considerada como linguagem para especificação conceitual.

O sistema PSL/PSA recebe a especificação conceitual descrita em PSL e a armazena num banco de dados que pode ser posteriormente acessado por um conjunto de programas chamado PSA (Problem Statement Analizer) que produzem relatórios sobre esta especificação.

Nesse capítulo abordaremos o problema do mapeamento da especificação conceitual armazenada no banco de dados do PSA para o esquema interno do DBMS TOTAL.

Um esquema proposto por Melo [8] para este mapeamento é ilustrado pela figura abaixo:



Para este mapeamento devemos criar uma linguagem CIML-TOTAL (Conceptual Internal Mapping Language) que descreve o mapeamento a ser realizado. O programa APPRG tem como entrada esta descrição do mapeamento e a especificação conceitual do sistema que esta armazenada no banco de dados do PSA e produz o esquema interno (DBMOD) para TOTAL e uma tabela MAPTAB que contêm informações sobre este mapeamento.

A seguir apresentamos uma proposta para a linguagem CIML-TOTAL.

Mapeamento das entidades

Um tipo de entidade pode ser mapeado de dois modos diferentes dependendo da existência ou não de elementos que podem ocorrer mais de uma vez para uma determinada entidade. Esse seria o caso do tipo de entidade "empregado" que tivesse como uma de suas propriedades elementares o elemento "nome do filho".

Para o primeiro caso o mapeamento do tipo de entidade será feito para um arquivo mestre, Desta forma uma entidade será mapeada para um registro mestre

Para o segundo caso o mapeamento do tipo de entidade será feito para um arquivo mestre associado a alguns arquivos variáveis; um arquivo variável para cada elemento que tenha possibilidade de ocorrencia múltipla Desta forma uma entidade será mapeada para um registro mestre associado a algumas cadeias de registros variáveis, uma cadeia para cada elemento que tenha possibilidade de ocorrencia múltipla.

O comando CIML-TOTAL para entidades seria:

```
ASSIGN ENTITY entity name
       TO MASTER FILE file name,
  [IF ANY ELEMENT OCCURS {MORE THAN} n TIME
                         {LESS THAN}
   THEN {USE LINK TO VARIABLE      }  ];
        {USE REPEATING ELEMENT     }
```

O DBMS TOTAL necessita que um nome de arquivo seja composto por quatro caracteres. Desta forma no comando acima o nome do arquivo mestre por exemplo seria formado pelos quatro primeiros caracteres do nome da entidade e os nomes dos arquivos variáveis pelos quatro primeiros caracteres do nome do elemento que vai ser mapeado para esse arquivo, existindo ainda a possibilidade de outras regras para formação dos nomes.

Mapeamento dos elementos

Os elementos serão mapeados para "DATA-ELEMENTS ". Os elementos que não podem ter ocorrencia múltipla serão mapeados para "DATA-ELEMENTS" do arquivo mestre e os que podem ter ocorrencia múltipla acima de um valor n especificado, serão mapeados para "DATA-ELEMENTS" dos arquivos variáveis.

Conforme vimos no capítulo anterior o DBMS TOTAL não guarda nenhuma informação sobre o tipo dos "DATA-ELEMENTS" que serão armazenados no banco de dados, desta forma no

mapeamento dos elementos só estaremos preocupados com o tamanho em bytes das informações que serão associadas a cada "DATA-ELEMENT".

O nome de um "DATA-ELEMENT" do DBMS TOTAL deve ser formado por oito carecteres, desta forma no mapeamento acima poderemos adotar a convenção de formar o nome de um "DATA-ELEMENT" da seguinte forma: os quatro primeiros bytes serão iguais aos do nome do arquivo mestre para o qual será feito o mapeamento do tipo de entidade ao qual pertence o elemento a ser mapeado e os quatro últimos caracteres deverão ser os quatro primeiros caracteres do nome do elemento a ser mapeado.

O comando CIML.TOTAL para elementos seria:

```
ASSIGN ELEMENT  element name
    TO DATA-ELEMENT  data-element name;
```

Mapeamento das relações

As relações entre as entidades serão mapeadas através de arquivos do tipo variável. Se existirem dados associados a uma determinada relação eles deverão ser mapeados para "DATA-ELEMENTS" do arquivo variável que implementa a relação.

Uma análise deste mapeamento, do ponto de vista dos aspectos importantes sobre relações abordados no capítulo dois, nos permite as seguintes observações:

i) conectividade permitida: "many-to-many"

Partindo de um registro do arquivo mestre que implementa a primeira entidade da relação podemos ter uma cadeia de zero ou mais registros no arquivo variável, logo uma entidade do primeiro tipo pode estar associada a diversas entidades do segundo tipo. De forma análoga partindo do registro que implementa a segunda entidade podemos ter uma cadeia de zero ou mais registros no arquivo variável, logo uma entidade do segundo tipo pode estar associada a deversas entidades do primeiro tipo. Desta forma a conectividade permitida para as relações que serão mapeadas é many-to-many.

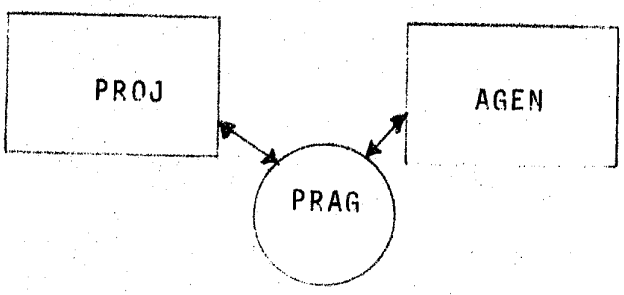
ii) N-ariedade permitida: "qualquer"

Do ponto de vista do DBMS TOTAL com o mapeamento acima podemos implementar relações de qualquer N-ariedade pois para cada entidade que participe da relação teremos um arquivo mestre associado ao variável que implementa a relação, pois conforme vimos no capítulo anterior o DBMS TOTAL permite que um arquivo do tipo variável seja ligado a um número qualquer de arquivos mestres.

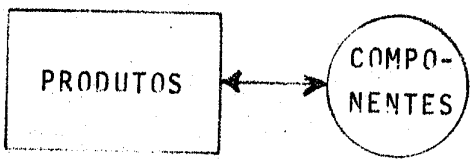
Entretanto, cabe observar que a PSL só permite relações de N-ariedade dois. Desta forma embora o modelo conceitual adotado permita relações de N-ariedade maior que dois, a utilização da PSL impõe esta restrição e o especificador ao ter que especificar uma relação de N-ariedade maior que dois terá que subdividí-la em duas ou mais relações de N-ariedade dois.

Quanto aos tipos de entidades que podem estar envolvidos em uma relação, concluimos que os três pontos importantes apresentados no capítulo dois são atendidos, ou seja:

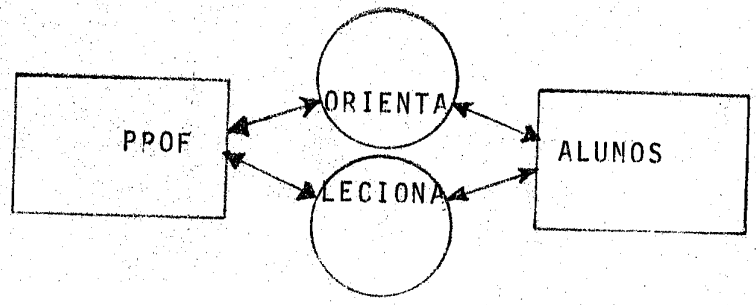
i) podemos ter relações entre entidades de tipos diferentes, isto corresponde ao caso mais simples que é temos dois arquivos mestres diferentes ligados pelo variável que implementa a relação exemplo:



ii) podemos ter relações entre entidades do mesmo tipo, isto corresponde ao caso de termos dois tipos de ligação entre um arquivo mestre e o variável que implementa a relação. Exemplo:



iii) podemos ter mais de um tipo de relação entre duas entidades, pois podemos ter dois arquivos mestres ligados por mais de um arquivo variável (um para cada relação entre as duas entidades representadas pelos arquivos mestres). Exemplo:



O comando CIML.TOTAL para relações seria:

```
ASSIGN RELATION  relation name
    TO VARIABLE FILE  file name ,
   [ASSDAT  element 1  TO  data-element 1,
            element 2  TO  data-element 2,
            ..............................];
```

REFERÊNCIAS

1 - DATE, C.J. An introduction to database systems. Addison Wesley, 1975.

2 - MARTIN, J. Computer data-base organization. Englewood Cliffs, N. J., Prentice-Hall, 1975.

3 - GUSMÃO, E. D. Especificação Conceitual de Sistemas de Informação apoiados em Banco de Dados. Tese de mestrado apresentada ao departamento de Informática da PUC, 1976.

4 - OS TOTAL REFERENCE MANUAL, Cincon, Ohio, 1976.

5 - CODASYL DATA BASE TASK GROUP REPORT, New York, ACM, 1971.

6 - TEICHROEW, D & HERSHEY III, "An introduction to PSL/PSA", ISDOS Working Paper nº 86, The University of Michigan, March 1974.

7 - HERSHEY et al., "Problem Statement Language Version 3.0, Language Reference Manual, "ISDOS Working Paper nº 68. The University of Michigan, May 1975.

8 - MELO, RUBENS N. "On the Mapping From the Conceptual Specification to an Internal Model in Data Base Design" (a ser publicado).