UNIVERSIDADE FEDERAL DA PARAÍBA

CENTRO DE CIÊNCIAS E TECNOLOGIA

CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

PROJETO E IMPLEMENTAÇÃO DE ESQUEMAS EXTERNOS BASEADOS NO MODELO DE ENTIDADES E RELACIONAMENTOS

LUCIANO ROMERO SOARES DE LIMA

CAMPINA GRANDE

DEZEMBRO - 1988

LUCIANO ROMERO SOARES DE LIMA

PROJETO E IMPLEMENTAÇÃO DE ESQUEMAS EXTERNOS BASEADOS NO MODELO DE ENTIDADES E RELACIONAMENTOS

Dissertação apresentada ao Curso de MESTRADO EM SISTEMAS E COMPUTAÇÃO da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do Grau de Mestre.

ÁREA DE CONCENTRAÇÃO : CIÊNCIA DA COMPUTAÇÃO

MASCUS COSTA SAMPAIO

Orientador

GIUSEPPE MANGIOVI

Co-Orientador

CAMPINA GRANDE - PB

DEZEMBRO-1988

L732p Lima, Luciano Romero Soares de
Pojeto e implementacao de esquemas externos baseados no modelo de entidades e relacionamentos / Luciano Romero Soares de Lima. - Campina Grande, 1988.
133 f. : il.

Dissertacao (Mestrado em Ciencias e Computacao) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Banco de Dados 2. Gerenciadores de Banco de Dados 3. Dissertacao I. Sampaio, Marcus Costa, M.Sc. II. Mongiovi, Giuseppe, M.Sc. III. Universidade Federal da Paraiba - Campina Grande (PB) IV. Título

CDU 004.65(043)

PROJETO E IMPLEMENTAÇÃO DE ESQUEMAS EXTERNOS BASEADOS NO MODELO DE ENTIDADES E RELACIONAMENTOS

LUCIANO ROMERO SOARES DE LIMA

Dissertação Aprovada em 27/12/88

MARCUS COSTA SAMPAIO - M.Sc

- PRESIDENTE -

GIUSEPPE MONGIOVI - M.Sc

- EXAMINADOR -

DANIEL ALBERTO MENASCÉ - Ph.D

- EXAMINADOR -

Campina Grande - Pb

DEZEMBRO - 1988

À Gabriela, minha esposa,

a Rafael meu filho e

a Mariana minha filha.

AGRADECIMENTOS

Aos professores Marcus Costa Sampaio e Giuseppe Mangiovi pela orientação, incentivo e dedicação com que acompanharam o desenvolvimento deste trabalho.

Aos professores do DSC, que direta ou indiretamente contribuiram com sugestões e incentivos durante a realização deste trabalho.

Aos funcionários do DSC, em especial, aos do Laboratório de Microcomputadores, pelo apoio material as fases de desenvolvimento deste trabalho.

Aos colegas do Mestrado em Informática e de outras áreas, pelo incentivo e colaboração.

À Aarão Aata e Vicente Manuel pelo apoio na edição e diagramação deste texto.

S U M Á R I O

LISTA DE ILUSTRAÇÕES

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 5

Capítulo 6

Capítulo 7

R E S U M O

Este trabalho tem como objetivo dotar os mais diversos gerenciadores de banco de dados de um Modelo Conceitual padronizado, em que cada usuário, ou grupo de usuários, tem acesso somente a uma parte do banco de dados (Esquema Externo), cujo alcance depende das permissões dadas. Portanto, qualquer operação de manipulação de dados (consulta e manutenção) é realizada obedecendo à respectiva definição do esquema externo.

Neste sentido, apresentamos aqui, uma proposta de modelagem de dados para o nível conceitual e para o nível externo, bem como descrevemos as linguagens de definição e manipulação de dados utilizadas no nosso modelo e as implementações realizadas.

1. INTRODUÇÃO

Observando o mundo dos Gerenciadores de Banco de Dados (GBD's) existentes, verifica-se que a maioria não define e nem implementa a arquitetura de sistemas de banco de dados que foi proposta, em fins de 1979, pelo comitê ANSI/X3/SPARK [4], o que provoca uma considerável diminuição nas potencialidades de utilização destes GBD's. Esta arquitetura trata um Banco de Dados (BD) através de três níveis gerais: o Nível Externo, o Nível Conceitual e o Nível Interno, conforme mostra a figura 1.1.

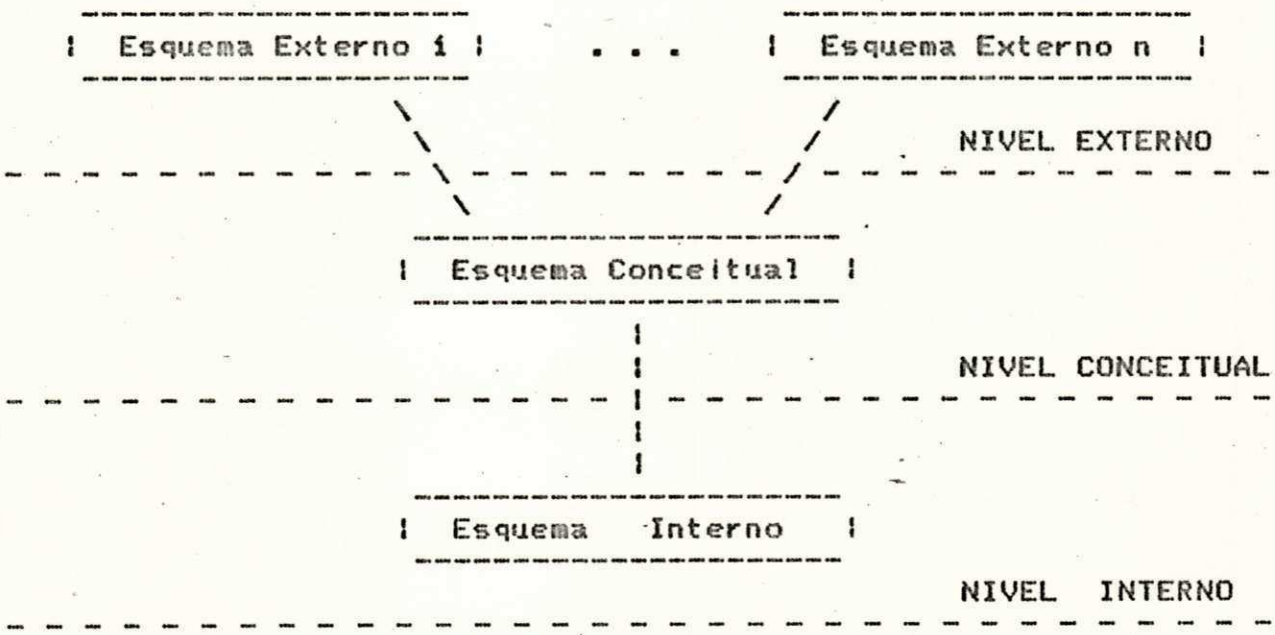


FIGURA 1.1 - ARQUITETURA DE SISTEMAS DE BD DEFINIDA PELO COMITÉ ANSI/X3/SPARK

O Nível Externo é o conjunto de todos os esquemas externos, onde cada esquema externo representa uma visão lógica da parte do BD que um usuário ou grupo de usuários possui. O conceito de esquemas externos é importante por permitir a personalização da

utilização do BD,pois a maioria dos usuários não está interessada no total do BD, mas somente numa porção restrita do mesmo.

O Nível Conceitual é composto pelo Esquema Conceitual, que retrata a visão lógica geral do BD definida pela totalidade da comunidade de usuários.

O Nível Interno é composto pelo Esquema Interno, que retrata a forma como os dados do BD estão realmente armazenados e como são gerenciados.

Pretendemos desenvolver um sistema que permita capacitar vários GBD's existentes com a proposta dos três níveis, e que possam ser usados independentemente do modelo de dados particular de cada GBD.

A arquitetura geral do sistema que estamos propondo é composta de duas partes distintas e integradas: O BD Conceitual e o BD Operacional, conforme mostra a figura 1.2.

O BD Conceitual é a parte padronizada da arquitetura do sistema. É o BD no qual o usuário vê a organização lógica dos seus dados. Ele é composto de:

- **Esquema Conceitual (EC)**: tem o mesmo significado conforme definido em [4]. O modelo de dados adotado pelo EC será analisado no próximo capítulo.

- **Esquemas Externos (EE)**: têm o mesmo significado conforme definido em [4], cujo modelo de dados é o mesmo adotado pelo EC.

O BD Operacional é um BD armazenado em um ou mais computadores e é composto de:

- **Esquema Operacional (EO)**: descreve a organização lógica dos dados de um BD operacional. Tradicionalmente é baseado nos modelos Hierárquico [7], de Rede [6] e Relacional [5].

- **GBD Operacional**: gerencia o BD Operacional (armazenamento de dados, acesso aos dados, entre outros). Comercialmente existem vários GBD's, diferindo entre si por adotarem, ou modelos operacionais diferentes, ou estratégias de implementação diferentes para um modelo comum. Como exemplo, podemos citar o SBD/TS[9], o ORACLE [11], o INGRES[12], o ADABAS[13], etc.

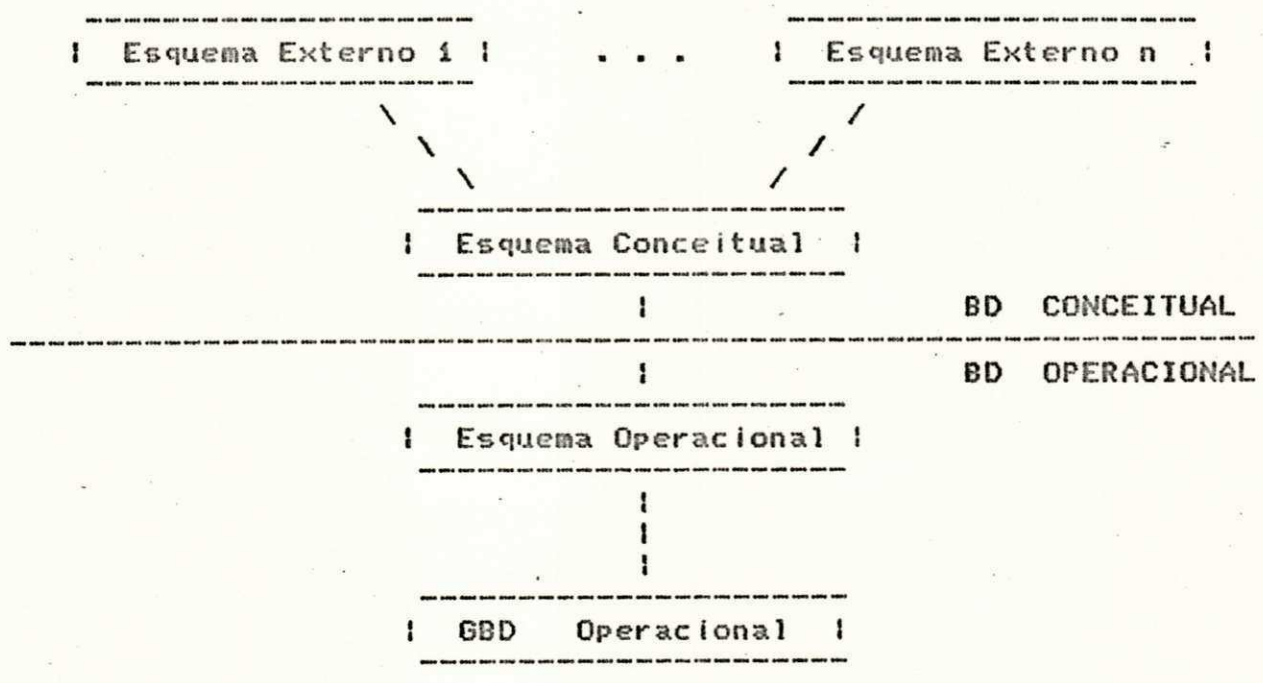


FIGURA 1.2 - ARQUITETURA GERAL DO SISTEMA

Destacamos desta arquitetura do sistema dois elementos principais:

- O Mapeador EE X EC - realiza o mapeamento dos componentes de um EE para os seus correspondentes no EC. Como um EE e o EC pertencem à parte padronizada da arquitetura do sistema, este mapeador funciona independente do GBD utilizado na execução da tarefa. Aproveitando-se desta independência, o sistema proposto realiza, aqui, otimizações que tornam as operações, na fase operacional, mais eficientes.

- **O Mapeador EC X EO** - realiza o mapeamento dos componentes do EC para os seus correspondentes no EO. Como o EO depende da escolha de um GBD operacional, esta parte é um componente mutável dentro da arquitetura do sistema.

Os detalhes acerca do sistema proposto serão objetos de discussão dos capítulos subsequentes.

O **Capitulo 2** apresenta uma descrição geral do Modelo Conceitual de Dados adotado no nosso trabalho, bem como, descreve os elementos de uma metodologia de desenvolvimento de sistemas de BD usando o nosso modelo como suporte.

O **Capitulo 3** apresenta uma descrição detalhada das linguagens utilizadas pelo sistema proposto, na definição de dados do EC e na definição e manipulação de dados de um EE.

O **Capítulo 4** apresenta uma descrição sucinta das estruturas internas que compõem o EC.

O **Capítulo 5** apresenta uma descrição dos detalhes da implementação da linguagem de definição de dados de um EE.

O **Capítulo 6** apresenta uma descrição dos detalhes da implementação da consulta na linguagem de manipulação de dados de um EE.

O **Capítulo 7** apresenta uma descrição detalhada da operacionalização do modelo conceitual adotado pelo sistema, utilizando como estudo de caso o GBD operacional SBD/TS.

O capítulo 8 resume as principais conclusões tiradas a partir do desenvolvimento do trabalho, bem como apresenta possíveis trabalhos que possam ser desenvolvidos a partir deste.

2 - O MODELO CONCEITUAL DE DADOS

A necessidade de se abstrair fatos e informações fornecidos pelo mundo real, que possam ser utilizados de modo satisfatório numa determinada aplicação, propiciou o surgimento dos Modelos de Dados [1,2,3]. Portanto, podemos afirmar que um modelo de dados é uma representação lógica das informações e fatos do mundo real na forma abstrata de dados e de suas operações.

Antes de descrevermos o nosso modelo de dados, faremos uma descrição sumária de alguns modelos existentes, sendo que, em primeiro lugar, abordaremos os modelos de dados tradicionalmente adotados pelos GBD's comerciais atuais e, posteriormente, falaremos sobre o MODELO DE ENTIDADES E RELACIONAMENTOS (MER) de Chen[8], que ao nosso ver, é a base das várias propostas de modelos de dados surgidas até agora, inclusive a nossa.

2.1 - O Modelo de Dados Relacional

O modelo de dados relacional [5] é atualmente o modelo de dados mais utilizado pelos GBD's comerciais existentes. Seus conceitos foram estabelecidos no início da década de 70 por Codd, e os principais são os seguintes:

- **Relação(R)**: tabela de dados valorados, bidimensional, cujo conjunto de valores horizontais é chamado de TUPLA, e cada componente de uma tupla é chamado de ATRIBUTO, cujo conjunto de valores fica na vertical. Cada atributo de uma relação deve ser nomeado e conter no máximo um valor, sendo que o atributo ou o

conjunto de atributos que diferencia uma tupla de outra, não deve possuir valores NULOS (CHAVE PRIMÁRIA DA RELAÇÃO). Cada relação, numa modelagem relacional, deve possuir um nome único que a diferencie das outras relações existentes.

Um exemplo de relação está na figura 2.1.

NUM_PROF	NOME_PRF	SALARIO	NUM_DEP
12345	LUCIANO	45000,00	0023
12471	MARCUS	45000,00	0022
12482	GIUSSEPE	45000,00	0023

Observa-se que NUM_PROF é a chave primária da relação PROFESOR

FIGURA 2.1 - RELAÇÃO PROFESSOR

- **Domínio de um atributo numa relação(D)**: conjunto de todos os valores que um atributo pode ter dentro de uma relação.

- **Relacionamentos entre tuplas de relações**: tuplas de uma relação se relacionam com outras tuplas de outra relação, através da existência de um atributo ou conjunto de atributos, que tenham os mesmos domínios e em algum momento, valores comuns entre si.

Se um atributo ou conjunto de atributos de uma relação "i" for uma chave primária de uma relação "j", chama-se, então, este atributo ou conjunto de atributos de CHAVE ESTRANGEIRA da relação "i".

A popularidade do modelo de dados relacional deve-se, principalmente, ao fato dele utilizar, como base, a teoria matemática das relações, largamente conhecida e formalizada, o

que contribui para uma assimilação mais fácil. Além disto, a modelagem de dados em tabelas bidimensionais é bastante natural e se apresenta no cotidiano da vida há muito tempo.

A maior desvantagem do modelo de dados relacional reside na sua limitação semântica, no tocante, principalmente, a modelagem de relacionamentos entre relações, que se dá de forma indireta (não fica explícito no diagrama relacional), além de produzir uma redundância de vários dados.

São exemplos de GBD's relacionais comerciais o ORACLE [11], o INGRES [12], o ADABAS [13], entre outros.

2.2 - O Modelo de Dados CODASYL

O modelo de dados CODASYL [6] teve seu relatório final divulgado em abril de 1971, contendo não somente as especificações sobre a estrutura lógica e física dos dados, como também, as linguagens de definição de dados do Esquema Conceitual, dos Esquemas Externos e a linguagem de manipulação de dados. Aqui, só abordaremos, sumariamente, sobre os conceitos estruturais lógicos da modelagem de dados, usando o modelo CODASYL:

- **Item de Dado**: menor unidade de dado existente no modelo. Chama-se **Ocorrência de Item de Dado**, o valor do mesmo num determinado instante.

- **Registro**: conjunto referenciável de um ou mais itens de dado. Chama-se de **Tipo de Registro**, um registro nomeado com a

definição dos seus itens de dado e de Ocorrência do Registro os valores assumidos pelos itens de dado de um tipo de registro num dado instante.

- **Conjunto de Registros**: conjunto referenciável de tipos de registros. Cada conjunto contém um tipo de registro especificado como DONO e um ou mais tipos de registros especificados como MEMBROS. Chama-se **Ocorrência de Conjunto de Registros** quando acontecer uma ocorrência de um tipo registro dono e uma ou mais ocorrências do tipo registro membro.

Portanto, os tipos registros se relacionam entre si através de relacionamentos 1:1 ou 1:n, sendo que estes relacionamentos aparecem na modelagem de forma direta.

Vejamos um exemplo de uma modelagem de dados usando o modelo de dados CODASYL, na figura 2.2.

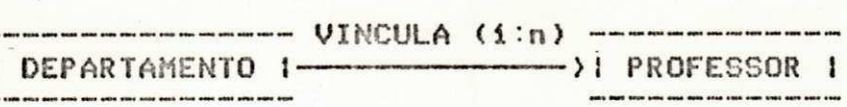


FIGURA 2.2 - MODELAGEM COM O MODELO DE DADOS CODASYL, USANDO NOTAÇÃO BACHMAN [14]

Então, uma modelagem de dados CODASYL será um conjunto de registros e seus relacionamentos, representando os dados e suas integrações numa aplicação.

O modelo CODASYL apresenta como principal vantagem a explicitação dos relacionamentos entre os registros, sem a necessidade de se ter dados redundantes, sendo que consideramos a impossibilidade de representar relacionamentos n:m diretamente

entre os registros envolvidos, uma limitação semântica importante, além de algumas características que o aproximam da parte implementável do BD, como por exemplo, a utilização do conceito de PONTEIROS.

Um exemplo de GBD que utiliza os conceitos do modelo de dados CODASYL é o DMSII da UNISYS [23]. O GBD/TS, utilizado neste trabalho, é um GBD que usa os conceitos estendidos do modelo CODASYL.

Existe, ainda, um modelo de dados chamado HIERÁRQUICO [7], que é um caso particular do modelo de dados CODASYL e por isto não teceremos maiores comentários sobre ele. O GBD IMS [16] da IBM utiliza este modelo de dados.

2.3 - O Modelo de Entidades e Relacionamentos

Além dos modelos de dados descritos anteriormente, nos últimos anos têm aparecido muitas propostas de modelos semânticos de BD [3,15,16]. Todos, em última análise, são enriquecimentos de um modelo original, o Modelo de Entidades e Relacionamentos (MER) de Peter Chen [8]. Os principais conceitos estabelecidos pelo MER são:

- **Entidade**: representa, abstratamente, qualquer fato ou informação identificável fornecida pelo mundo real, que precisa ser utilizada numa determinada aplicação. São exemplos de entidades um funcionário de uma empresa, um departamento de uma universidade, um contribuinte da Receita Federal, entre outras.

Chamamos de **Conjunto de Entidades**, o conjunto de todas as entidades com características semelhantes, como por exemplo o o conjunto de funcionários, o conjunto de departamentos, o o conjunto de contribuintes, etc. Um conjunto de entidades é representado por um **retângulo** nomeado, conforme mostra a figura 2.3.

FIGURA 2.3 - REPRESENTAÇÃO DE CONJUNTOS DE ENTIDADES NO MER

- **Atributos**: são partes nomeadas componentes dos conjuntos de entidades, que guardam informações específicas sobre uma determinada entidade, e, portanto, as diferenciam entre si. Por exemplo, o nome, a matrícula,o endereço e o salário são atributos que caracterizam o conjunto de FUNCIONÁRIOS. Os atributos são representados conforme mostra a figura 2.4.

FIGURA 2.4 - REPRESENTAÇÃO DOS ATRIBUTOS DO CONJUNTO DE ENTIDADES FUNCIONÁRIOS NO MER

- **Relacionamentos**: são associações entre duas ou mais entidades. Por exemplo, entre a entidade 'dsc' e a entidade 'func1', pode existir o relacionamento 'vincula'. Chamamos **Conjunto de Relacionamentos**, ao conjunto de todos os

relacionamentos entre entidades de um conjunto de entidades e as entidades de outro ou outros conjuntos de entidades. Um conjunto de relacionamentos é nomeado e representado por um losango, conforme mostra a figura 2.5.

FIGURA 2.5 - CONJUNTO DE RELACIONAMENTOS VINCULA

Existem três classes de relacionamento no MER:

a) Uma entidade **X** relacionada com uma outra entidade **Y** e vice-versa (chama-se relacionamento da **classe 1:1**).

b) Uma entidade X relacionada com várias outras entidades e cada uma dessas últimas, relacionadas somente com X (chama-se relacionamento da **classe 1:n**).

c) Várias entidades relacionadas com várias outras entidades (chama-se relacionamento da **classe n:m**).

As entidades de um conjunto de entidades podem se relacionar com entidades do mesmo conjunto de entidades (chama-se Auto-Relacionamento). Neste caso, uma mesma entidade deve exercer papéis diferentes no relacionamento.

2.4 - O Modelo de Entidades e Relacionamentos Adaptado (MERA)

O modelo proposto MERA, é, basicamente, o MER. Fizemos algumas alterações importantes nele com o intuito de esclarecer melhor o significado dos relacionamentos entre as entidades e de permitir a definição de mais regras de integridade . As

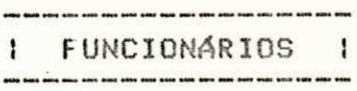
razões para nos mantermos próximos do MER são de ordem prática: seria inviável implementar um modelo mais rico em semântica por questões de eficiência, ainda mais utilizando um GBD para micro-computadores, como está sendo nosso caso.

2.4.1 - Entidades

Têm o mesmo significado do MER, inclusive a simbologia adotada. Existe, ainda, o conceito de **Entidade Virtual**, que é uma entidade sem atributos, que participa da modelagem devido a alguma necessidade particular criada pela aplicação envolvida, sendo representada por um círculo(veja o exemplo da figura 2.14).

2.4.2 - Atributos

Têm o mesmo significado do MER, sendo representados conforme mostra o exemplo da figura 2.6.



FUNCIONÁRIOS (NOME, MATRÍCULA, ENDEREÇO, SALÁRIO)

FIGURA 2.6 - REPRESENTAÇÃO DOS ATRIBUTOS DO CONJUNTO DE ENTIDADES FUNCIONÁRIOS NO MERA

2.4.3 - Relacionamentos

São associações entre duas ou mais entidades, sendo que o conjunto de todos os relacionamentos entre entidades com um mesmo significado, é chamado de **Conjunto de Relacionamentos.** Obrigatoriamente, cada conjunto de relacionamentos do MERA tem um

sentido. Por esta razão, usa-se uma **seta** para representá-lo de um conjunto de entidades **para** um outro(ou o mesmo) conjunto de entidades .

As classes possíveis de um conjunto de relacionamentos são:

a) Uma entidade do conjunto de entidades origem (e) relacionada com uma única entidade do conjunto de entidades destino (**para**) e vice-versa (chama-se relacionamento da **classe 1:1**).

b) Uma entidade do conjunto de entidades origem (**de**) relacionada com várias outras entidades do conjunto de entidades destino (**para**) e cada uma destas entidades destino não se relacionando com qualquer outra entidade origem (chama-se relacionamento da **classe 1:n**).

Vejamos um exemplo comparativo do MER (figura 2.7) com o MERA (figura 2.8):

FIGURA 2.7 - EXEMPLO DE MODELAGEM COM O MER.

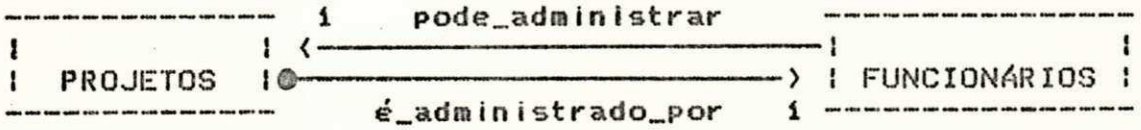


FIGURA 2.8 - EXEMPLO DE MODELAGEM COM O MERA.

Como pode ser visto, no MERA a semântica dos relacionamentos é ressaltada, ou seja, entre os dois conjuntos de entidades **PROJETOS** e **FUNCIONÁRIOS**, há pelo menos dois relacionamentos: cada

um deles tem significado diferente (**pode_administrar**, **é_administrado_por**), dependendo do **sentido** do relacionamento (indicado pela seta). Como observação, a existência dos relacionamentos nos dois sentidos não é obrigatória, podendo ser num só sentido, dependendo do uso que se quer fazer do BD. Ainda em relação à figura 2.8, verificamos: que o conjunto de entidades origem (de) é **PROJETOS** em **é_administrado_por** e é **FUNCIONÁRIOS** em **pode_administrar**; que o conjunto de entidades destino (para) é **FUNCIONÁRIOS** em **é_administrado_por** e **PROJETOS** em **pode_administrar**; que não consideramos atributos de relacionamentos; que na origem de cada conjunto de relacionamentos está subentendida a cardinalidade **UM** ("um (implícito) funcionário pode administrar um (explícito) projeto"; "um (implícito) projeto é administrado por um (explícito) funcionário"); o símbolo ● indica a existência de uma restriçao de integridade, que será explicada na sub-seção 2.4.5.2.

Finalizando, o MERA suporta um relacionamento de classe mais geral, o **n:m**, que é tratado como a composição de dois relacionamentos 1:n, cujos sentidos são opostos, formando um par de relacionamentos opostos e dependentes. Esta representação do relacionamento n:m visa aumentar a capacidade de semântica da descrição do BD. As figuras 2.9 e 2.10 mostram um exemplo comparativo do relacionamento n:m no MER e no MERA. O símbolo ⚷, ligando os relacionamentos **aloca** e **pode_participar_de**, indica que estes relacionamentos formam um relacionamento n:m no MERA (um par de relacionamentos opostos e dependentes).

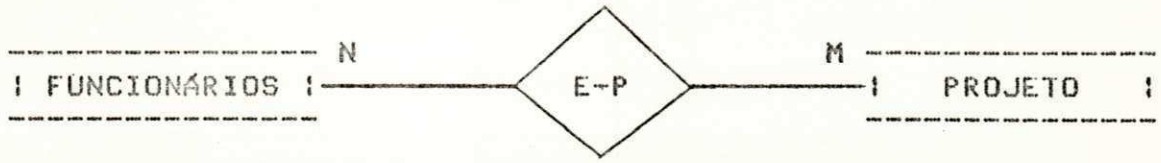


FIGURA 2.9 - EXEMPLO DE RELACIONAMENTO N:M NO MER.

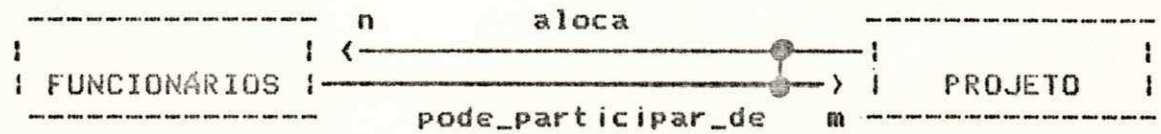


FIGURA 2.10 - MESMO EXEMPLO DA FIGURA 2.9 COM O MERA.

2.4.5- Restrições de Integridade do MERA

São regras que devem ser obedecidas pelas operações de manutenção do BD, com a finalidade de preservar a qualidade dos dados armazenados [2,3,10].

O MERA, na sua proposta atual, prevê dois tipos de restrições de integridade:

a) Restrição de Integridade de Valor.

b) Restrição de Integridade de Relacionamento.

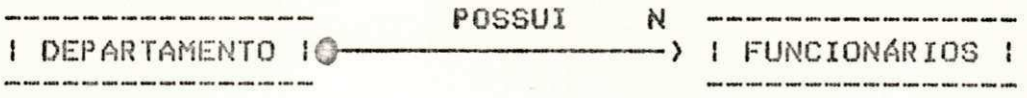
2.4.5.1- Restrição de Integridade de Valor (RIV)

É uma restrição de integridade que prevê a existência de atributos cujos valores não podem ser nulos. Isto ocorre em duas situações distintas:

a) Na definição dos atributos das entidades que funcionarão como CHAVE DE ORDENAÇÃO de um conjunto de relacionamentos 1:n. A chave de ordenação é um atributo do conjunto de entidades destino num relacionamento 1:n, cujos valores serão utilizados para ordenar as entidades.

b) Quando o usuário quiser ter a garantia de que um determinado atributo, que não é chave de ordenação, nunca terá valor NULO armazenado no BD, ele indicará no esquema através de um traço sublinhando o atributo.

Vejamos um exemplo de definição de RIV na figura 2.11.



DEPARTAMENTO (ND, NOME_DEP)
FUNCIONÁRIOS (NE, NOME_FUN, SALÁRIO)
CHAVE_POSSUI (NE)

FIGURA 2.11 - DEFINIÇÃO DE RIV's NO MERA

Observamos que NF do conjunto de entidades FUNCIONÁRIOS possui uma RIV por ser chave de ordenação do conjunto de relacionamentos POSSUI, enquanto ND no conjunto de entidades DEPARTAMENTO e SALÁRIO no conjunto de entidades FUNCIONÁRIOS possuem uma RIV imposta pelo usuário do BD.

2.4.5.2 - Restrição de Integridade de Relacionamento (RIR)

É uma restrição de integridade que prevê a existência de conjuntos de relacionamentos que obrigarão as entidades do conjunto de entidades origem a sempre estarem participando dos seus relacionamentos. Este conceito é chamado de RELACIONAMENTO TOTAL em [3] e adotaremos o simbolo ● (também presente em [3]) colocado ao lado do conjunto de entidades que exige a RIR.

No exemplo da figura 2.8, podemos dar a seguinte interpretação: "Nenhum projeto pode existir sem ser administrado

por um funcionário".

2.4.6- Metodologia de Projeto de BD usando o MERA

Descrevemos aqui, sumariamente, os elementos básicos para o desenvolvimento de sistemas de BD usando o MERA como suporte. Para isto, modelaremos um sistema de BD para uma rede estadual de ensino, que servirá de exemplo em todo o restante deste trabalho.

2.4.6.1 - Projeto do Esquema Conceitual

O primeiro passo importante é conhecermos os objetivos gerais do problema, com a intenção de detectarmos os fatos ou informações que poderão ser utilizados como entidades.

Por exemplo, o desenvolvimento de um sistema de BD para uma rede estadual de ensino deveria possibilitar a realização dos seguintes objetivos:

a) Administrar o funcionamento de todos colégios estaduais, conhecendo-se os seus professores, os seus alunos, os seus funcionários e as suas disciplinas.

b) Controlar o nível de qualificação dos professores, no intuito de administrar a ascensão funcional por mérito.

A partir destes objetivos, poderemos concluir que são necessários os seguintes conjuntos de entidades: Professores, Colégios, Disciplinas, Estudantes, Qualificações e Funcionários.

Definidos os conjuntos de entidades, faz-se necessário, agora, conhecermos quais os atributos que deverão compôr cada

conjunto de entidades. Para isto, precisaremos estudar a amplitude de utilização do sistema.

Por exemplo,se no estudo da utilização do sistema de BD para a rede estadual de ensino, tivéssemos as seguintes operações:

a) Emissão da Folha de Pagamento pela Secretária de Finanças.

b) Contrôle das administrações escolares pela Secretária de Educação.

c) Fiscalização do cumprimento legal do período letivo pela Inspetoria de Ensino.

Estas operações nos levaria a supôr a necessidade dos seguintes atributos: SALÁRIO e NUMERO-DEPENDENTES nos conjuntos de entidades PROFESSORES e FUNCIONÁRIOS (item a); DIRETOR e VICE-DIRETOR no conjunto de entidades COLÉGIOS (item b); DURAÇÃO no conjunto de entidades DISCIPLINA (item c).

Portanto, através da análise destas e de outras informações, seríamos capazes de definir todos os atributos de cada conjunto de entidades.

Definidos os conjuntos de entidades e seus atributos, faz-se necessário avaliarmos como as entidades se relacionam entre si, suas cardinalidades e suas chaves de ordenação.

Por exemplo, analisando o papel do conjunto de entidades PROFESSORES no BD, verifica-se que todos devem estar vinculados a algum colégio, então, existe um relacionamento entre o conjunto de entidades COLÉGIOS e o conjunto de entidades PROFESSORES. Percebe-se, ainda , que um colégio vincula um ou mais

professores, e que cada professor só está em um colégio, caracterizando assim, um relacionamento 1:n. Conhecido o relacionamento entre um colégio e os professores e como a cardinalidade é 1:n, deve-se definir para o relacionamento uma chave de ordenação. Através de discussão e análise com os usuários, pode-se optar por definir dois conjuntos de relacionamentos (VINCULA1 e VINCULA2) entre os conjuntos de entidades COLÉGIOS e PROFESSORES, sendo que, um conjunto de relacionamentos com chave de ordenação dada pelo número do professor (VINCULA1) e outro conjunto de relacionamentos com chave de ordenação dada pelo nome do professor (VINCULA2).

Portanto, através deste tipo de análise definiremos todos os conjuntos de relacionamentos, suas cardinalidades e suas chaves de ordenação possíveis e necessárias ao BD.

Ao fim deste passo, teremos o corpo inicial do esquema conceitual do BD, sendo necessário agora analisarmos quais as RIV's e RIR's que serão incorporadas ao BD conceitual proposto.

Por exemplo, analisando os dados de uma entidade professor, constata-se que o seu número, o seu nome e o seu salário são informações permanentes e fundamentais para a existência da entidade, portanto, nunca devem ter valores nulos. Sendo assim, verifica-se a existência de RIV's nestes atributos. Como o número e nome do professor foram usados como chave de ordenação, eles já possuem uma RIV implícita, bastando, portanto, criar uma RIV explícita para o atributo salário. Assim, partindo deste tipo de

análise, verificaremos todas RIV's existentes no BD.

Finalmente, o encerramento do ciclo de construção do esquema conceitual do BD se dá com a necessidade de se definir as RIR's.

Por exemplo, analisando o papel da entidade professor no BD, poder-se-ia constatar que não deveria existir professores que não estivessem diretamente vinculados a um colégio. Sendo assim, é importante definimos uma RIR para o conjunto de relacionamentos entre PROFESSORES e COLÉGIOS.

Nesta altura, teríamos um primeiro esquema conceitual completo do BD, que deveria ser novamente analisado, através de uma interação com os usuários, até que fosse considerado adequado aos objetivos estabelecidos (figura 2.12).

A seguir mostramos como são descritos as entidades, os atributos e as chaves no MERA:

```
PROF (NP, NOME_PRF, SALÁRIO, ENDEREÇO, N_DEPEND)
COLÉGIOS (NC, NOME_COL, DIRETOR, VICE_DIR, DATA_FUN)
DISC (ND, NOME_DSC, DURAÇÃO,)
ESTUD (NE, NOME_ESTUD, DAT_NASC)
FUNCION (NF, NOME_FUN, SALÁRIO, ENDEREÇO, N_DEPEND)
QUALIFIC (NQ, NOME_QUALIF, NÍVEL_QUA)
CHAVE_VINCULA1 (NP)
CHAVE_VINCULA2 (NOME_PRF)
CHAVE_POSSUI (NE)
CHAVE_ENSINA (ND)
CHAVE_ALOCA (NF)
CHAVE_CURSA (ND)
CHAVE_EHCURSAD (NE)
CHAVE_TEM (NQ)
CHAVE_PERTENCE (NF)
```

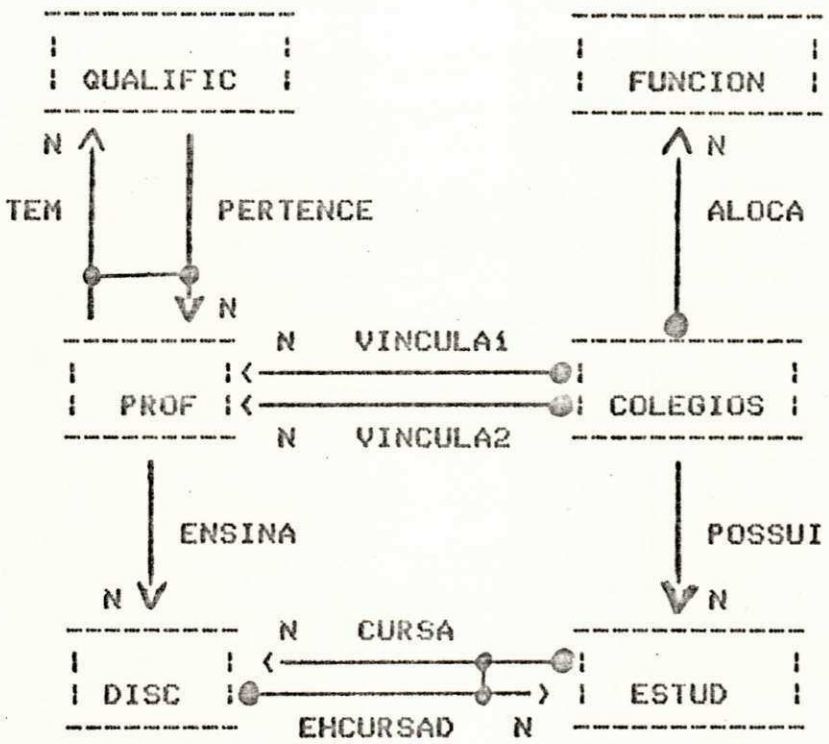


FIGURA 2.12 - ESQUEMA CONCEITUAL 'Colégios'.

2.4.6.2- Projeto de um Esquema Externo.

De posse do esquema conceitual, projetado a partir das necessidades gerais de toda a comunidade de usuários, iremos projetar os esquemas específicos (externos) que reflitam as necessidades particulares de cada usuário ou grupo de usuários.

Um EE obedece às seguintes regras de construção:

- Um conjunto de entidades num EE é um sub-conjunto vertical e/ou horizontal (Projeção e/ou Seleção) de um conjunto de entidades num EC.

- Cada relacionamento num EE tem um relacionamento correspondente no EC.

administração da política salarial.

Portanto, através de análises deste tipo definiremos todos os atributos dos conjuntos de entidades que servirão ao EE, bem como os valores possíveis.

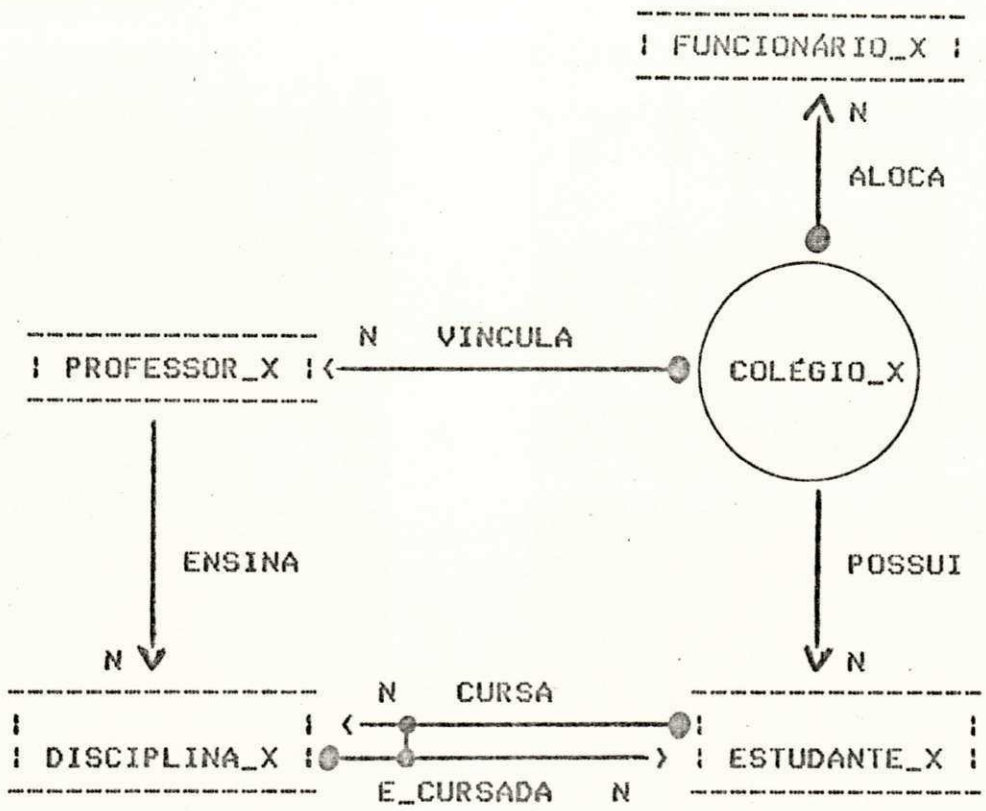
Finalizando o cíclo de construção do EE, analisaremos quais dos conjuntos de relacionamentos do EC deverão servir ao EE.

Por exemplo, no EC temos dois conjuntos de relacionamentos entre professores e colégios, que diferem entre si devido as suas chaves de ordenação. Se na análise das necessidades da administração do colégio X se constatasse que só seria preciso um tipo de conjunto de relacionamentos que possuisse chave de ordenação por nome de professor, não precisaríamos, portanto, do conjunto de relacionamentos vinculai projetado para o EE.

Ao fim destes passos, teremos um projeto inicial do EE. Este EE deve ser reanalisado até que as necessidades específicas dos usuários ou grupos de usuários sejam satisfeitas.

Vale salientar, que durante a construção de um EE, pode-se constatar deficiências no EC proposto e aí teremos que reanalisar as necessidades gerais da comunidade de usuários e partirmos para sua reestruturação.

Mostramos, na figura 2.13, o exemplo do EE desenvolvido.



PROFESSOR_X (NUM_PROF, NOME_PROF, ENDEREÇO)
DISCIPLINA_X (NUM_DISC, NOME_DISC, DURAÇÃO)
ESTUDANTE_X (NUM_ESTUD, NOME_ESTUD, DAT_NASC)
FUNCIONÁRIO_X (NUM_FUNC, NOME_FUNC, ENDEREÇO)
CHAVE_VINCULA (NOME_PROF)
CHAVE_POSSUI (NUM_ESTUD)
CHAVE_ENSINA (NUM_DISC)
CHAVE_E_CURSADA (NUM_ESTUD)
CHAVE_CURSA (NUM_DISC)

Figura 2.14 - ESQUEMA EXTERNO COLÉGIO_X

3. INTERFACES COM O USUÁRIO

Á cada dia se consolida a proposta de transformar o uso do computador em algo fácil, que possa ser usado largamente por pessoas sem experiência em computação(usuários finais).

Foi tentando atender a esta proposta que projetamos interfaces entre o usuário e o MERA, que oferecessem todas as facilidades aos usuários finais.

Na abordagem que adotamos, primeiro define-se o esquema conceitual, depois os esquemas externos e então com um esquema externo definido, pode-se manipular o BD. Por isto, tivemos que definir três tipos de interfaces (linguagens):

LINDEC - Linguagem de Definição de Dados do EC.

LINDEX - Linguagem de Definição de Dados de um EE.

LIMADEX - Linguagem de Manipulação de Dados de um EE.

Neste capítulo, mostraremos os elementos básicos das três linguagens e como eles se apresentam para o usuário. Vale salientar que, a partir deste ponto do trabalho, chamaremos EC de **Esquema** e EE de **Sub-esquema** e tudo que estiver em negrito, nas telas que iremos apresentar, são informações fornecidas pelo usuário.

3.1) LINDEC

É uma linguagem interativa, baseada em menus e telas programadas, que será utilizada pelo Administrador do Banco de Dados (ABD) na definição do esquema do BD desejado.

Na LINDEC, os nomes dos seus componentes serão sempre compostos de um a oito caracteres alfa-numéricos, sendo o primeiro caractere uma letra (IDENTIFICADOR-LINDEC).

3.1.1) Identificação de um Esquema do BD

Um novo esquema de BD, para ser definido, deve ser nomeado através de um IDENTIFICADOR-LINDEC único para o dono do esquema (um ABD ou um grupo de ABD's).

Portanto, no caso da definição do nome do esquema da figura 2.12, teríamos:

```
+-----------------------------------------+
|  ESQUEMA  DO  BD                        |
|                                         |
|     NOME: Colégios                      |
|                                         |
|-----------------------------------------|
|      <mensagens do sistema>             |
+-----------------------------------------+
```

FIGURA 3.1 - TELA DE DEFINIÇÃO DO NOME DO ESQUEMA

3.1.2) Definição dos Conjuntos de Entidades e seus Atributos

Devidamente identificado o esquema, faz-se necessário definir os seus conjuntos de entidades e os seus atributos.

Para cada conjunto de entidades :

a) define-se um nome, que deve ser um IDENTICADOR-LINDEC único entre os conjuntos de entidades do esquema;

b) definem-se os atributos do conjunto de entidades. Para cada atributo :

b.1) define-se um nome, que deve ser um IDENTIFICADOR-LINDEC

único entre os atributos de um mesmo conjunto de entidades;

b.2) define-se o tipo de atributo. Os tipos disponíveis são:

- INTEIRO1 - indica um conjunto de valores **inteiros positivos** na faixa de **0 a 256.**
- INTEIRO+ - indica um conjunto de valores **inteiros positivos.**
- INTEIRO2 - indica o conjunto de valores **inteiros.**
- REAL - indica o conjunto de valores **reais.**
- DATA - indica um tipo da forma **dd/mm/aa** onde **'dd'** guarda os dias do mês **(01 a 31)**, **'mm'** guarda os meses do ano **(01 a 12)** e **'aa'** guarda os anos **(0 a 99).**
- LÓGICO - indica o tipo **booleano**, cujos valores são **FALSO** ou **VERDADEIRO.**
- CARACTER n - indica o tipo **caracter**, sendo que **n** indica a quantidade de caracteres do atributo.
- CADEIA n - indica o tipo **string**, onde **n** é número máximo de caracteres da cadeia.

b.3) define-se a existência ou não das RIV's dos atributos.

As figura 3.2 e 3.3 mostram as telas de definição dos conjuntos de entidades e seus atributos, baseadas no conjunto de entidades **PROF** e seus atributos (figura 2.12).

3.1.3) Definição dos Conjuntos de Relacionamentos

Para cada conjunto de relacionamentos :

a) define-se um nome, que deve ser um IDENTIFICADOR_LINDEC

único entre os conjuntos de relacionamentos;

b) define-se entre os conjuntos de entidades, qual será o conjunto de entidades **ORIGEM** e o conjunto de entidades **DESTINO**;

c) define-se o tipo do conjunto de relacionamentos, 1:1 ou 1:n;

d) define-se, caso o tipo de relacionamento seja 1:n, o atributo do conjunto de entidades **DESTINO** que deve ser **CHAVE DE ORDENAÇÃO**;

e) define-se a existência ou não de uma RIR no conjunto de relacionamentos;

f) define-se, entre os tipos de relacionamentos 1:n, aqueles que formarão o par de conjuntos de relacionamentos opostos e dependentes (representação do tipo de relacionamento n:m).

A figura 3.4 mostra a tela de definição dos conjuntos de relacionamentos do esquema **COLÉGIOS** (figura 2.12).

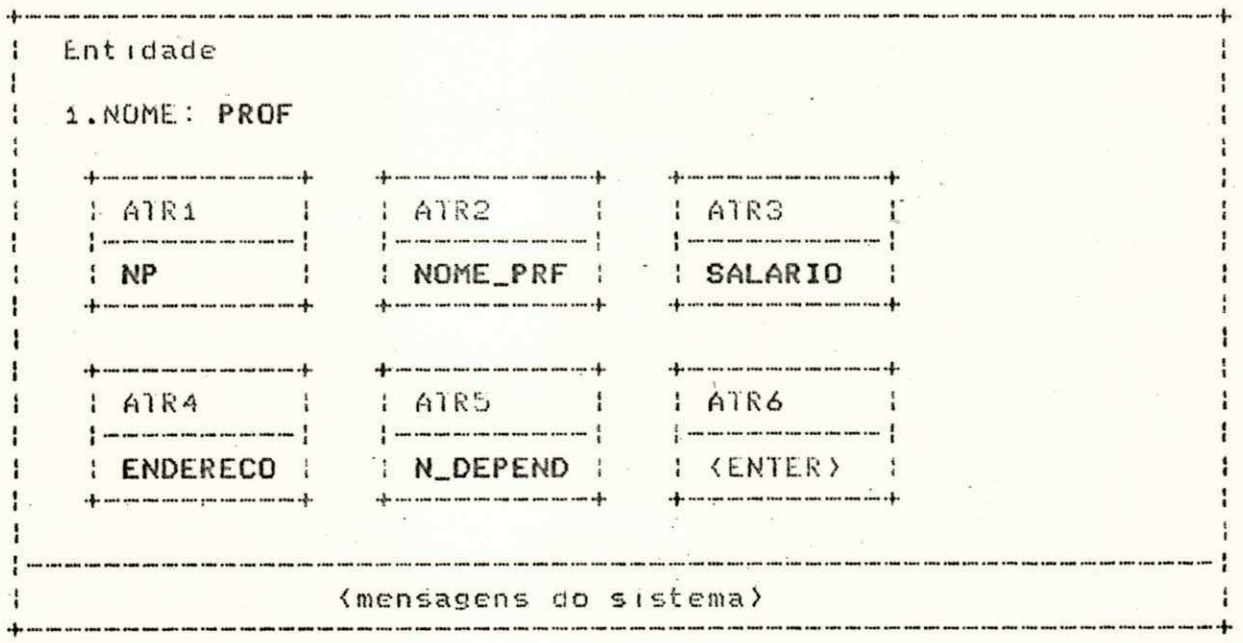


FIGURA 3.2 - TELA DE DEFINIÇÃO DE UM CONJUNTO DE ENTIDADES E SEUS ATRIBUTOS NUM ESQUEMA

```
+-----------------------------------------------------------------------+
| Escolha o tipo dos atributos e as RIV's                               |
|                                                                       |
| 1.NP          : INTEIRO+              RIV : NAO                       |
| 2.NOME_PRF    : CADEIA [20]           RIV : NAO                       |
| 3.SALARIO     : REAL                  RIV : SIM                       |
| 4.ENDERECO    : CADEIA [40]           RIV : NAO                       |
| 5.N_DEPEND    : INTEIRO1              RIV : NAO                       |
|                                                                       |
|-----------------------------------------------------------------------|
| Confirma          ↓ Próximo          ↑ Anterior                       |
+-----------------------------------------------------------------------+
```

FIGURA 3.3 - TELA DE DEFINIÇÃO DOS TIPOS DOS ATRIBUTOS E DA EXISTÊNCIA DA RIV

```
+-----------------------------------------------------------------------+
| Relacionamentos                                                       |
| +-------------------------------------------------------------------+ |
| | N | NOME      | DE        | RIR | PARA     | CHAVE     | CLASSE | |
| |---|-----------|-----------|-----|----------|-----------|--------| |
| | 1 | VINCULA1  | COLÉGIOS  |  s  | PROF     | NP        |  1:n   | |
| |---|-----------|-----------|-----|----------|-----------|--------| |
| | 2 | VINCULA2  | COLÉGIOS  |  s  | PROF     | NOME_PRF  |  1:n   | |
| |---|-----------|-----------|-----|----------|-----------|--------| |
| | 3 | ENSINA    | PROF      |  n  | DISC     | ND        |  1:n   | |
| |---|-----------|-----------|-----|----------|-----------|--------| |
| | 4 | CURSA     | ESTUD     |  s  | DISC     | ND        |  1:n   | |
| |---|-----------|-----------|-----|----------|-----------|--------| |
| | 5 | EHCURSAD  | DISC      |  s  | ESTUD    | NE        |  1:n   | |
| |---|-----------|-----------|-----|----------|-----------|--------| |
| | 6 | POSSUI    | COLÉGIOS  |  n  | ESTUD    | NE        |  1:n   | |
| |---|-----------|-----------|-----|----------|-----------|--------| |
| | 7 | ALOCA     | COLÉGIOS  |  s  | FUNCION  | NF        |  1:n   | |
| |---|-----------|-----------|-----|----------|-----------|--------| |
| | 8 | TEM       | PROF      |  n  | QUALIFIC | NP        |  1:n   | |
| |---|-----------|-----------|-----|----------|-----------|--------| |
| | 9 | PERTENCE  | QUALIFIC  |  n  | PROF     | NQ        |  1:n   | |
| |---|-----------|-----------|-----|----------|-----------|--------| |
| |10 | <ENTER>   |           |     |          |           |        | |
| +-------------------------------------------------------------------+ |
|                                                                       |
| Pares de Relacionamentos Opostos e Dependentes                        |
|                                                                       |
| 1. (CURSA,EHCURSAD)                                                   |
| 2. (TEM,PERTENCE  )                                                   |
| 3. (<ENTER>       )                                                   |
|                                                                       |
|-----------------------------------------------------------------------|
|                    (mensagens do sistema)                             |
+-----------------------------------------------------------------------+
```

FIGURA 3.4 - TELA DE DEFINIÇÃO DOS CONJUNTOS DE RELACIONAMENTOS DE UM ESQUEMA

3.1.4) Definição dos Usuários do Esquema

Para cada usuário do esquema :

a) define-se um nome, que deve ser um IDENTIFICADOR-LINDEC;

b) define-se uma senha, que deve ser um IDENTIFICADOR-LINDEC;

Deve-se observar, que o par (nome_do_usuário, senha) tem de ser único entre os usuários do esquema.

Vejamos a tela fornecida pelo sistema para a definição dos usuários do esquema (ABD's):

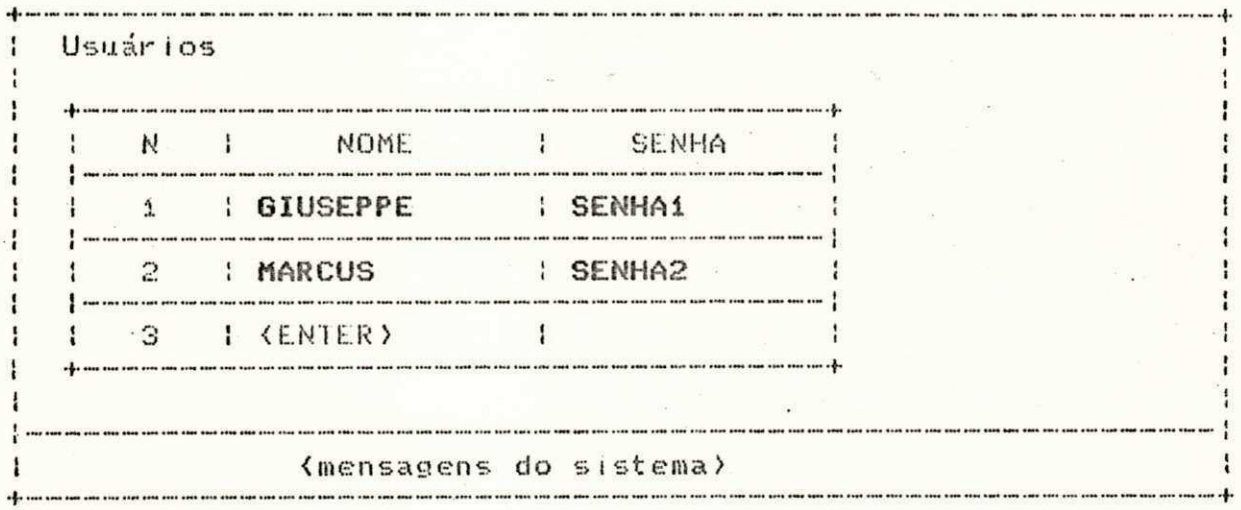


FIGURA 3.5 - TELA DE DEFINIÇÃO DOS USUÁRIOS

3.1.5) Confirmação do Projeto do Esquema

Definido o esquema, o sistema acionará um módulo que permitirá ao ABD realizar as seguintes tarefas:

a) **Listar** os componentes do esquema desenvolvido;

b) **Modificar** qualquer componente do esquema desenvolvido.

c) **Gravar** o esquema desenvolvido.

3.1.6) Detectação de Erros no Desenvolvimento do Esquema

Todos os passos,descritos anteriormente, são monitorados por sub-módulos que detectam erros e inconsistências nas informações fornecidas à LINDEC. A cada informação incorreta, o sistema emite um **alerta sonoro** e uma **mensagem de erro** explicativa, solicitando novamente a informação ao ABD. Os principais erros detectados são: duplicação de nomes dos componentes; definição indevida de chaves de ordenação e de tipos de atributos; tentativa de definição do relacionamento n:m com dois relacionamentos 1:n indevidos; entre outros.

3.2) LINDEX

É uma linguagem interativa, baseada em menus e telas programadas, que será utilizada, pelo ABD, na definição de um sub-esquema.

Na LINDEX, os nomes dos seus componentes serão sempre compostos de um a quinze caracteres alfanúmericos, sendo o primeiro caractere uma letra (IDENTIFICADOR-LINDEX).

3.2.1) Identificação do Usuário e do Esquema

A LINDEX, ao ser ativada, solicita do ABD o seu nome e a sua senha, que serão checados para verificar a sua existência. Confirmado a existência do usuário, o sistema solicitará o nome do esquema a ser utilizado e verificará se ele pertence ou não ao usuário identificado. Em caso afirmativo, o sistema carregará todas as informações necessárias do esquema para a memória.

Vejamos, através do esquema definido na figura 3.1 e de seus usuários definidos na figura 3.5, a tela fornecida pelo sistema para a identificação do usuário e do esquema.

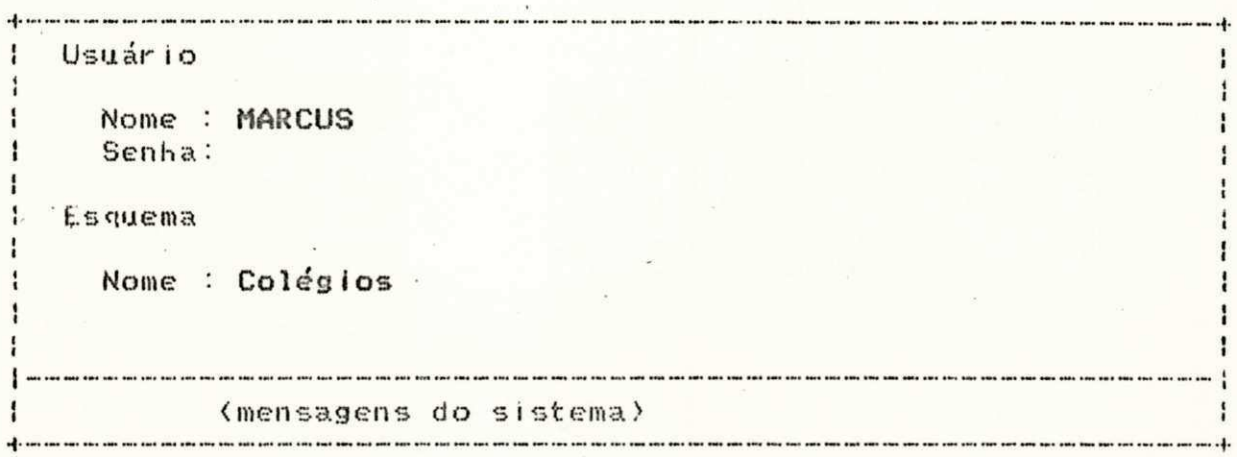


FIGURA 3.6 - TELA DE IDENTIFICAÇÃO DO USUÁRIO E DO ESQUEMA

3.2.2) Operações da LINDEX

Devidamente identificados o usuário e o esquema, o sistema ativa a tela de operações da LINDEX, conforme mostra a figura 3.7.

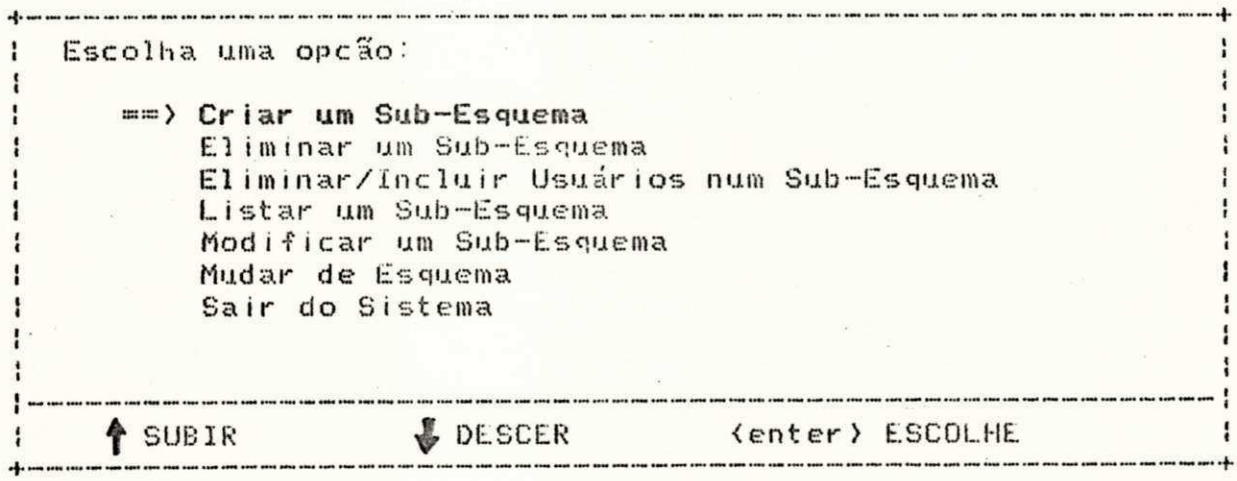


FIGURA 3.7 - TELA DE APRESENTAÇÃO DAS OPERAÇÕES DA LINDEX

3.2.2.1) Criar um Sub-Esquema

Esta operação permite ao usuário do esquema criar um novo sub-esquema. Vejamos como isto é possível:

- Identificação do Sub-Esquema

Um novo sub-esquema deve ser, primeiramente, identificado através de um nome, que deve ser um IDENTIFICADOR-LINDEX único para um usuário, ou grupo de usuários, dono do sub-esquema.

Portanto, no caso da definição do nome do sub-esquema da figura 2.13 (COLÉGIO_X), teríamos:

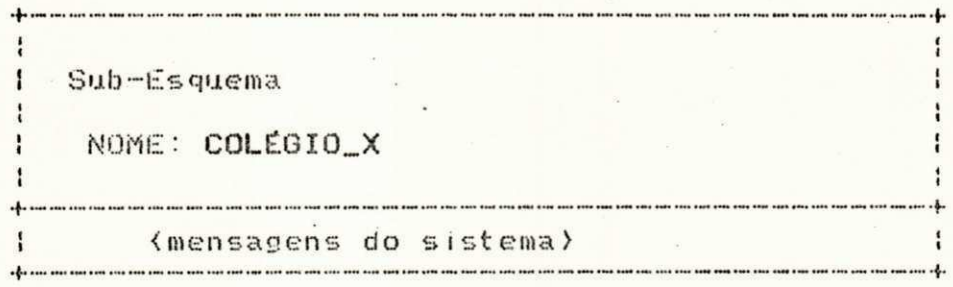


FIGURA 3.8 - TELA DE DEFINIÇÃO DO NOME DO SUB-ESQUEMA

- Definição dos Conjuntos de Entidades e seus Atributos

Para cada conjunto de entidades num sub-esquema :

a) define-se um nome, que deve ser IDENTIFICADOR-LINDEX único entre os conjuntos de entidades do sub-esquema;

b) define-se o seu correspondente no esquema;

c) define-se, através de uma projeção (sub-conjunto vertical), os atributos do esquema que participarão do sub-esquema. Estes atributos poderão ter nomes diferentes dos seus correspondentes

no esquema, sendo que, cada nome deve ser um IDENTIFICADOR-LINDEX único dentro do conjunto de entidades;

d) define-se, através de uma expressão de seleção (sub-conjunto horizontal), quais os conjuntos de valores dos atributos que poderão participar do sub-esquema. A sintaxe de uma expressão de seleção é definida pela seguinte Backus Normal Form (BNF)[22]:

⟨expr_seleção⟩ := ⟨expressão⟩;

⟨expressão⟩ := ⟨expressão⟩ ! ⟨termo⟩ / ⟨termo⟩

⟨termo⟩ := ⟨termo⟩ & ⟨fator⟩ / ⟨fator⟩

⟨fator⟩ := (⟨expressão⟩) / ⟨expressão_lógica⟩

⟨expressão_lógica⟩ := atributo ⟨operador_lógico⟩ ⟨valor⟩

⟨op_lógico⟩ := = / < / <> / > / <= / >=

⟨valor⟩ := ⟨constante de um tipo de atributo válido na LINDEX⟩

onde: & - conectivo **and**

! - conectivo **or**

A figura 3.9, apresenta a tela de definição dos conjuntos de entidades de um sub-esquema, usando o conjunto de entidades **COLÉGIO_X** (figura 2.13). Esta tela mostra que **COLÉGIO_X** foi definido como um conjunto de entidades virtual (não possui atributos, só interessando a condição NOME_COL = 'X'), que irá selecionar as entidades de PROFESSOR_X, FUNCIONÁRIO_X e ESTUDANTE_X, via os relacionamentos respectivos. Já para PROFESSOR_X (figura 3.10), o usuário terá que definir a projeção de **PROF** (figura 2.12) sobre os atributos NP, NOME_PRF e ENDEREÇO, escrevendo, em vez de ⟨ENTER⟩, NUM_PROF, NOME_PROF e ENDEREÇO, respectivamente.

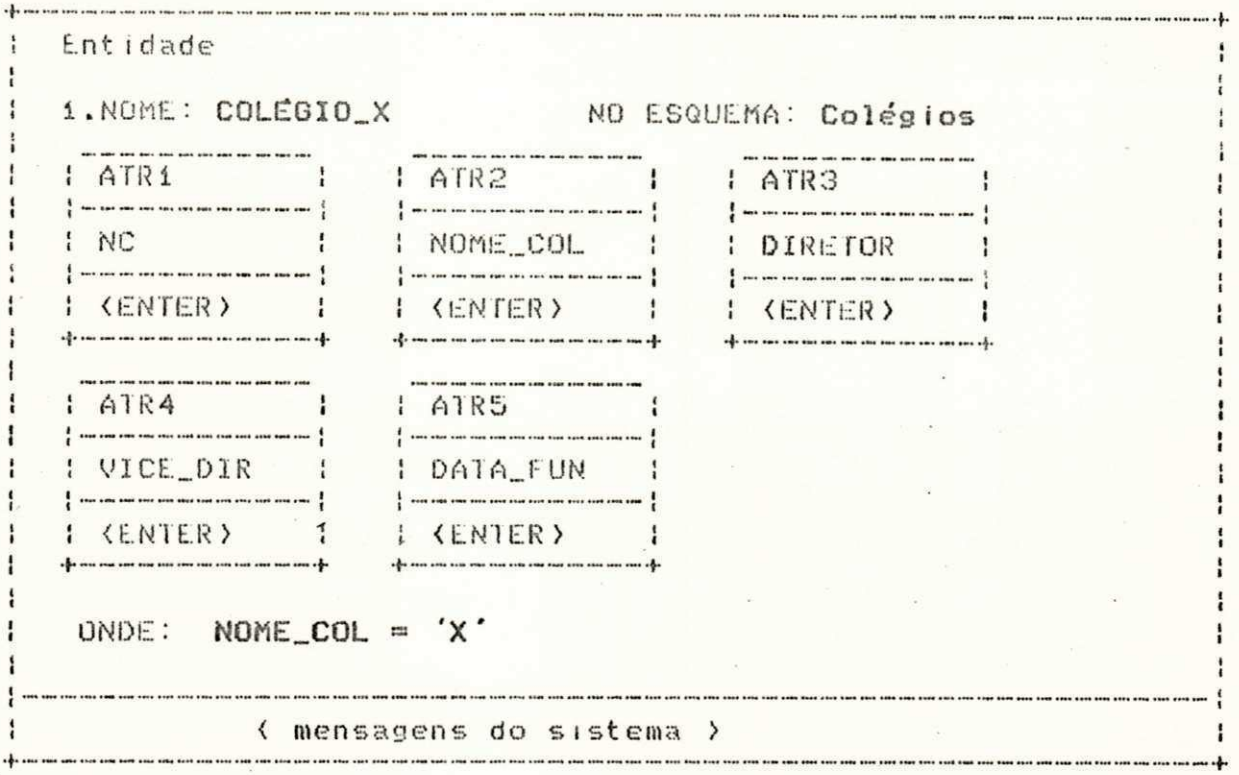


FIGURA 3.9 - TELA DE DEFINIÇÃO DE UM CONJUNTO DE ENTIDADES E OS SEUS ATRIBUTOS.

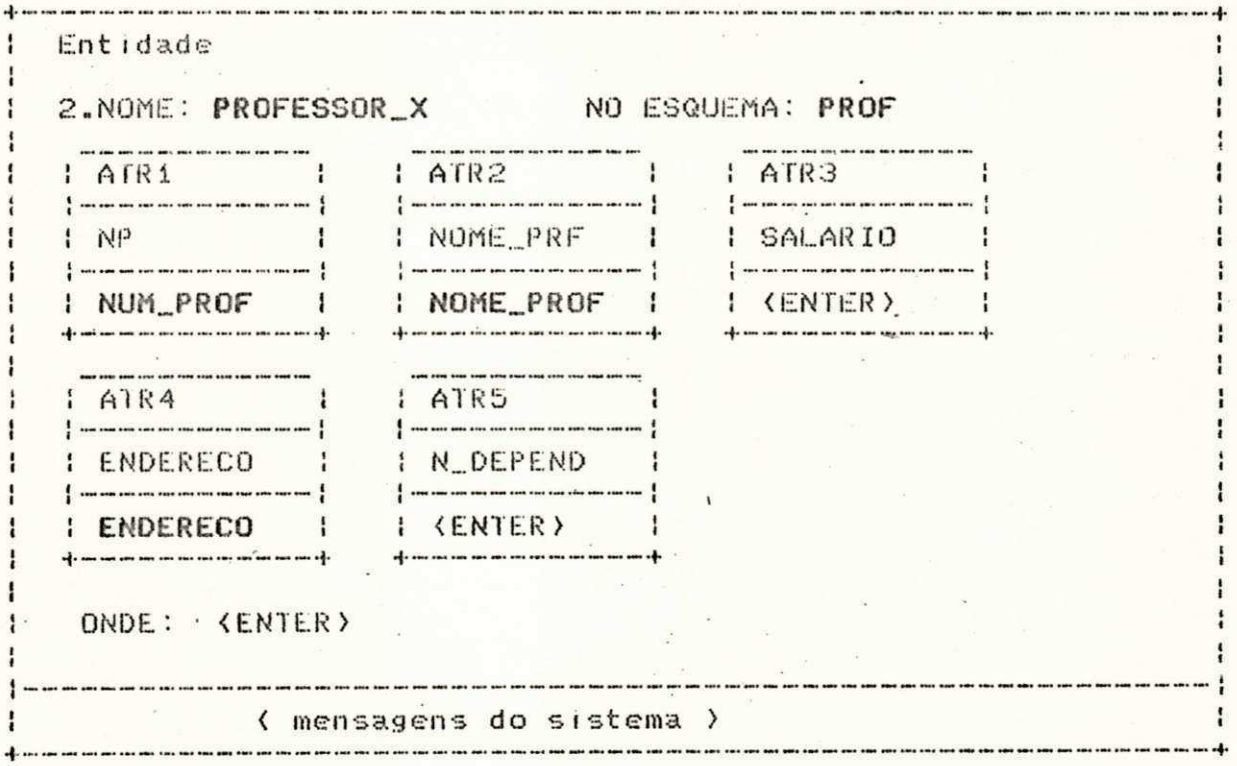


FIGURA 3.10 - TELA DE DEFINIÇÃO DE UM CONJUNTO DE ENTIDADES E SEUS ATRIBUTOS.

Observe, que não foi definida uma condição de seleção, para PROFESSOR_X, significando que todos os professores do COLÉGIO_X devem entrar neste sub-esquema.

- **Definição dos Conjuntos de Relacionamentos**

Para cada conjunto de relacionamentos num sub-esquema :

a) define-se um nome, que deve ser IDENTIFICADOR-LINDEX único entre os conjuntos de relacionamentos do sub-esquema;

b) define-se os conjuntos de entidades origem e destino dentre os conjuntos de entidades definidos no sub-esquema;

c) caso exista mais de um conjunto de relacionamentos equivalentes no esquema, é necessário indicar qual é conjunto de relacionamentos equivalente;

d) identificado o conjunto de relacionamentos equivalente no esquema, o sistema indicará o tipo do conjunto de relacionamentos correspondente.

A figura 3.11, apresenta a tela de definição dos conjuntos de relacionamentos de uma sub-esquema, baseada no sub-esquema **COLÉGIO_X**. Observa-se, que para evitar qualquer problema de ambiguidade na definição do conjunto de relacionamentos VINCULA, foi necessário indicar qual o conjunto de relacionamentos correspondente no esquema, no caso, VINCULA1.

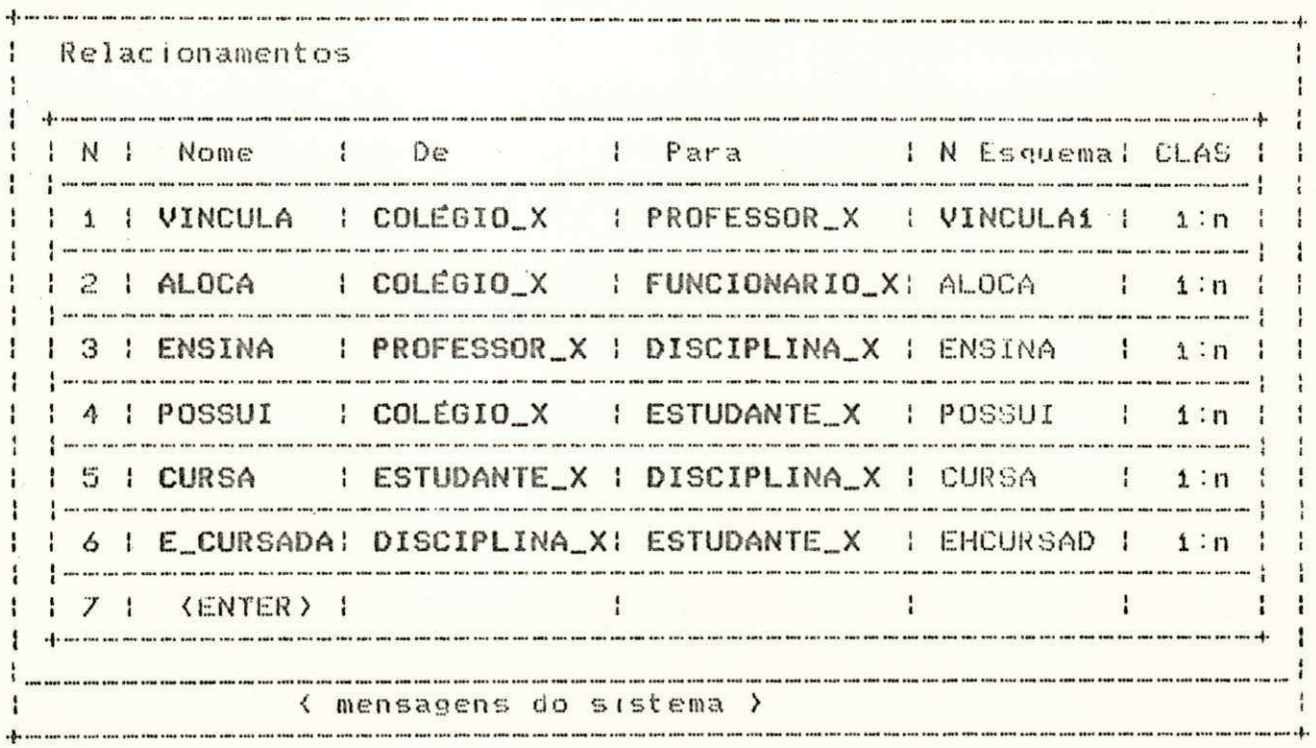


FIGURA 3.11 - TELA DE DEFINIÇÃO DOS CONJUNTOS DE RELACIONAMENTOS DE UM SUB-ESQUEMA.

- Definição dos Usuários do Sub-Esquema

Para cada usuário do sub-esquema :

a) define-se um nome, que deve um IDENTIFICADOR-LINDEX;

b) define-se uma senha, que deve ser um IDENTIFICADOR-LINDEX, sendo que, o par (nome_do_usuário,senha) deve ser único entre os usuários do sub-esquema.

c) define-se um tipo. Existem dois tipos disponíveis: **NG** (Não Grava) é aquele usuário que só pode consultar o BD; **LG** (Lê e Grava) é aquele usuário que pode consultar, modificar, incluir e/ou excluir no BD, sendo que as últimas operações só serão permitidas se o usuário tiver **acesso irrestrito** (sem projeção horizontal ou vertical) à entidade-objeto da operação.

Vejamos a tela fornecida pelo sistema para a definição dos usuários do sub-esquema COLÉGIO_X (figura 3.12).

FIGURA 3.12 - TELA DE DEFINIÇÃO DOS USUÁRIOS DE UM SUB-ESQUEMA.

- **Confirmação do Sub-Esquema Desenvolvido**

Definido o sub-esquema, o sistema oferece ao projetista do sub-esquema as seguintes operações:

a) **Listar** todos os componentes definidos para o sub-esquema.

b) **Modificar** qualquer componente do sub-esquema definido.

c) **Gravar** o sub-esquema definido.

- **Detectação dos Erros na Definição de um Sub-Esquema**

Este sub-módulo apresenta o mesmo padrão de monitoriamento de erros descrito na sub-seção 3.1.6. Aqui, os principais erros detectados são: duplicidade de nomes dos componentes do sub-esquema; tentativa de definição de um componente do sub-esquema sem equivalente no esquema; incompatibilidade de tipos numa expressão de seleção; entre outros.

3.2.2.2) Eliminar um Sub-Esquema

Esta operação desativa, das tabelas internas da LINDEX, todos os componentes de um sub-esquema existente.

3.2.2.3) Eliminar/Incluir Usuários num Sub-Esquema

Esta operação desativa ou inclui usuários na tabela de usuários da LINDEX.

3.2.2.4) Listar um Sub-Esquema

Esta operação mostra os componentes de um sub-esquema.

3.2.2.5) Modificar um Sub-Esquema

Esta operação permite a alteração de qualquer componente de um sub-esquema existente.

3.3) LIMADEX

É uma linguagem interativa, composta de menus e telas programadas, projetada para que os usuários finais possam manipular dados de um sub-esquema.

3.3.1) Identificando o Usuário e o Sub-Esquema

A LIMADEX, ao ser ativada, solicita do usuário seu nome e senha, que serão checados para verificar a sua existência. Confirmada a existência do usuário, o sistema solicitará o nome do sub-esquema a ser utilizado e verificará se ele pertence ou não ao usuário identificado. Em caso afirmativo, o sistema

carregará todas as informações do sub-esquema para a memória.

Vejamos, através do sub-esquema definido na figura 3.8, a tela fornecida pelo sistema para a identificação do usuário e do sub-esquema (figura 3.13).

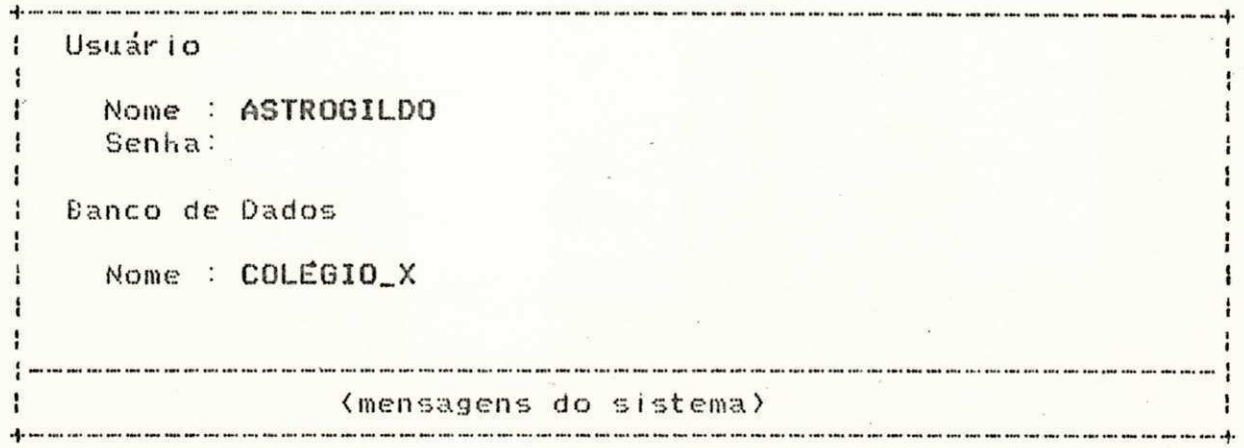


FIGURA 3.13 - TELA DE IDENTIFICAÇÃO DO USUÁRIO E DO SUB-ESQUEMA

3.3.2) Operações da LIMADEX

Após a identificação do usuário e do sub-esquema, é ativada a tela de operações da LIMADEX se o usuário for do tipo LG, senão o sistema ativará a tela da operação **CONSULTA.**

Vejamos, na figura 3.14, a tela de operações da LIMADEX:

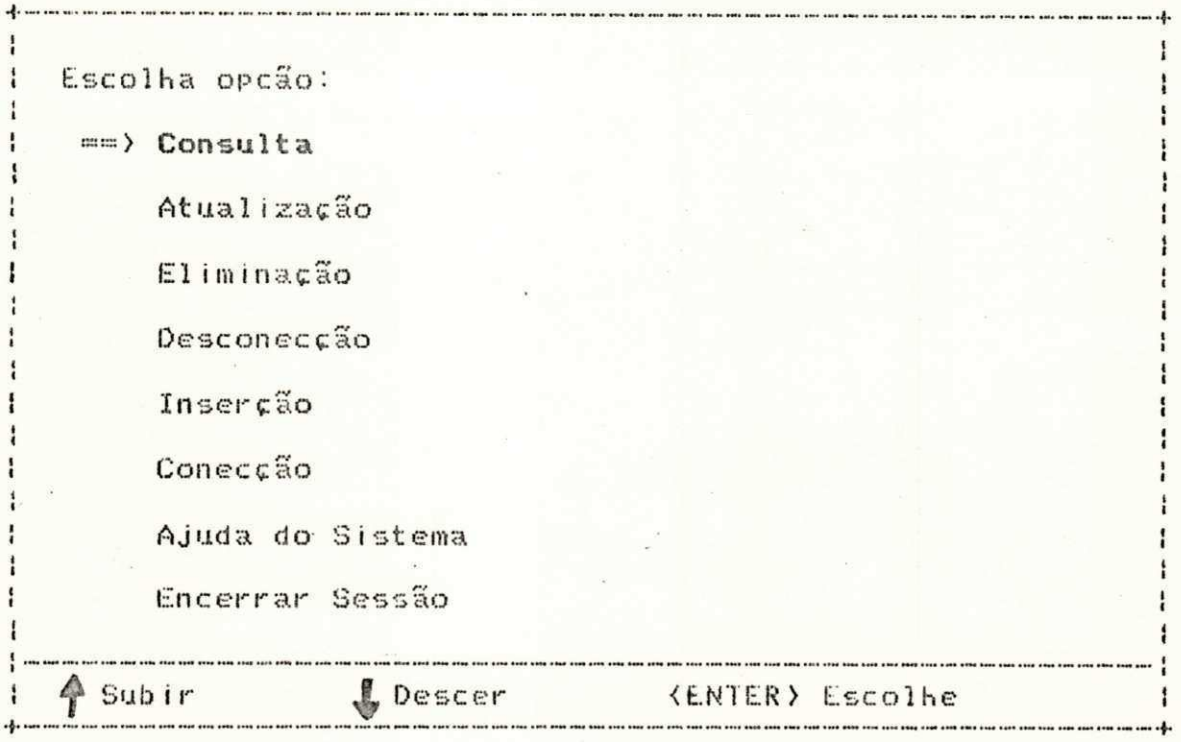


FIGURA 3.14 - TELA DE APRESENTAÇÃO DAS OPERAÇÕES DA LIMADEX

3.3.2.1) Consulta

Esta operação permite o acesso aos dados armazenados no BD, conforme a visão que o usuário tenha deste BD. Existem três sub-operações possíveis, conforme mostra a figura 3.15.

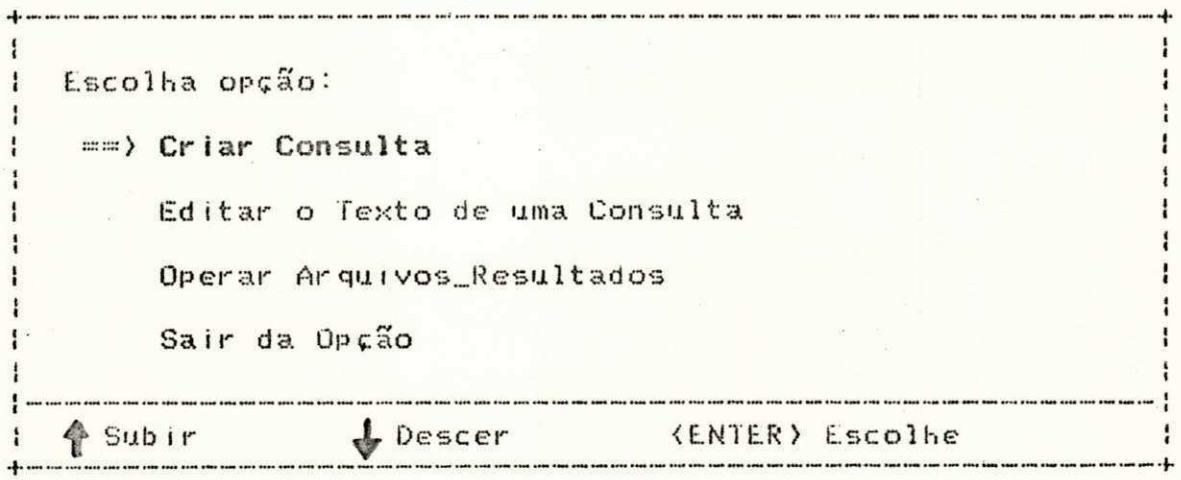


FIGURA 3.15 - TELA DE APRESENTAÇÃO DAS SUB-OPERAÇÕES DA CONSULTA

- Criar uma Consulta

Esta sub-operação cria uma nova consulta ao BD. Uma nova consulta possuirá os seguintes elementos:

a)**O Caminho da Consulta (CC)** - indica a sequência de conjuntos de entidades e conjuntos de relacionamentos participantes da consulta, sendo que um conjunto de entidades só poderá possuir no máximo dois conjuntos de relacionamentos presentes no CC, não sendo permitida a definição de um CAMINHO BIFURCADO. Entende-se por CAMINHO BIFURCADO aquele que possui um ou mais conjuntos de entidades que se relacionam com tres ou mais conjuntos de entidades, conforme mostra a figura 3.16. Vale ressaltar que consultas com caminhos bifurcados poderão ser resolvidas usando a sub-operação Operar Arquivos Resultados.

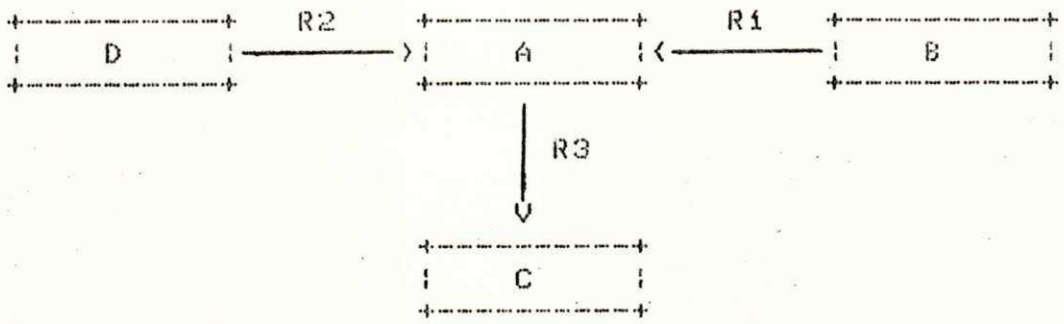


FIGURA 3.16 - EXEMPLO DE UM CAMINHO BIFURCADO

b)**A População da Consulta (PC)** - é o sub-conjunto das entidades e relacionamentos de **CC** que torna o **Predicado da Consulta (PDC)** sempre verdadeiro. **PDC** é uma expressão booleana, com BNF igual à da expressão descrita na LINDEX, envolvendo os atributos dos conjuntos de entidades do CC.

c)O **Resultado da Consulta (RC)** - é um sub-conjunto vertical dos conjuntos de entidades do **PC**. O conjunto de atributos de RC chama-se **Lista-Alvo.**

Para exemplificar, iremos mostrar a sequência de telas fornecidas pelo sistema para atender à seguinte consulta: **"Listar a matrícula e o nome dos professores com matrículas maiores do que 68, que ensinam disciplinas com mais de 2 meses de duração e que estas sejam cursadas por estudantes com matrículas entre 121 e 212".**

Para realizar a consulta solicitada, o usuário poderia definir o seguinte caminho:
PROFESSOR_X - ENSINA ---> DISCIPLINA_X <--- CURSA - ESTUDANTE_X, através da sequência de telas, conforme mostra a figura 3.17.

TELA1

```
+-----------------------------------------------------------------------+
|  Escolha uma Entidade                                                 |
|                                                                       |
|  ===> PROFESSOR_X                                                     |
|                                                                       |
|       DISCIPLINA_X                                                    |
|                                                                       |
|       ESTUDANTE_X                                                     |
|                                                                       |
|       COLÉGIO_X                                                       |
|                                                                       |
|       FUNCIONÁRIO_X                                                   |
|                                                                       |
|       Nenhuma das Entidades                                           |
|                                                                       |
|-----------------------------------------------------------------------|
| ↑ Subir ↓ Descer (Pg Dn) Prox_Pag (Pg Up) Pag_Ant (ENTER) Escolhe     |
+-----------------------------------------------------------------------+
```


TELA 2

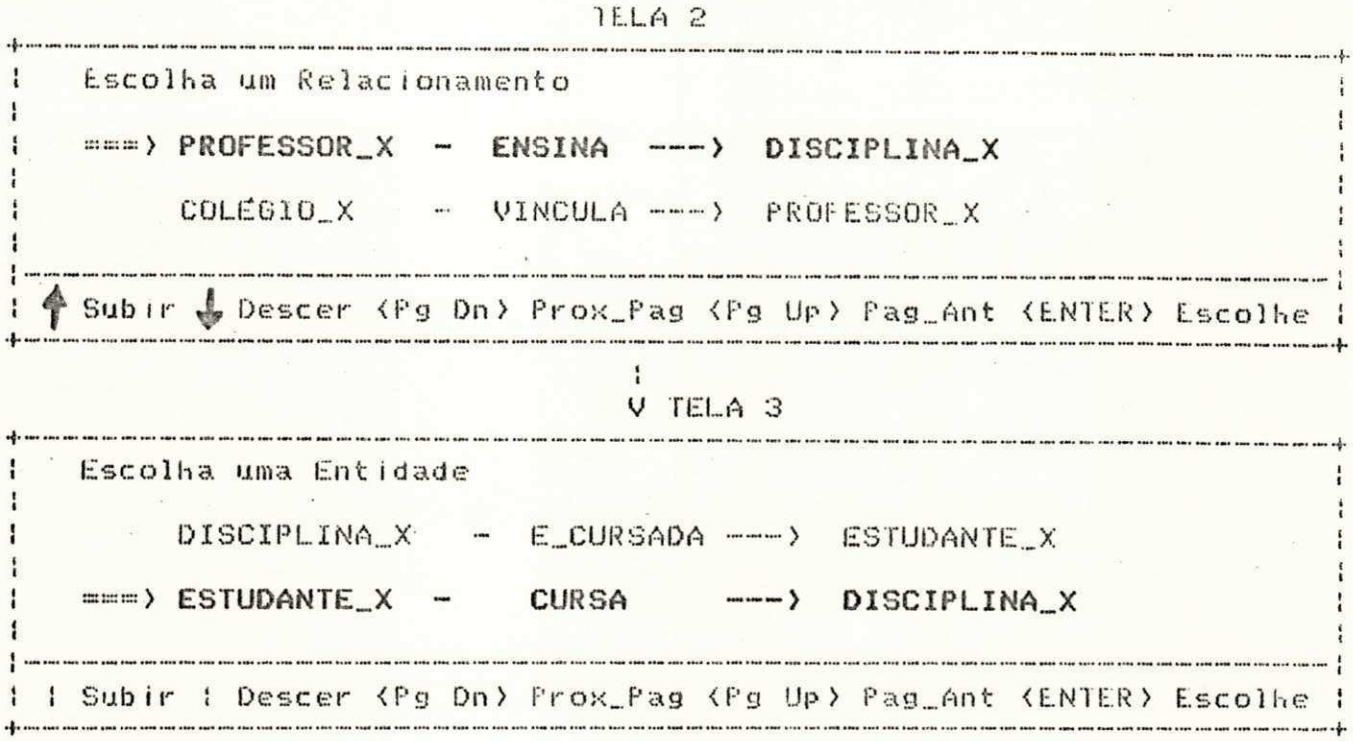


FIGURA 3.17 - TELAS DE DEFINIÇÃO DO CAMINHO DA CONSULTA

Definido o **CC**, o sistema apresentará o caminho selecionado e pedirá a sua confirmação. Caso o usuário não confirme o CC selecionado, a consulta será encerrada. A figura 3.18 apresenta a tela de confirmação do **CC**.

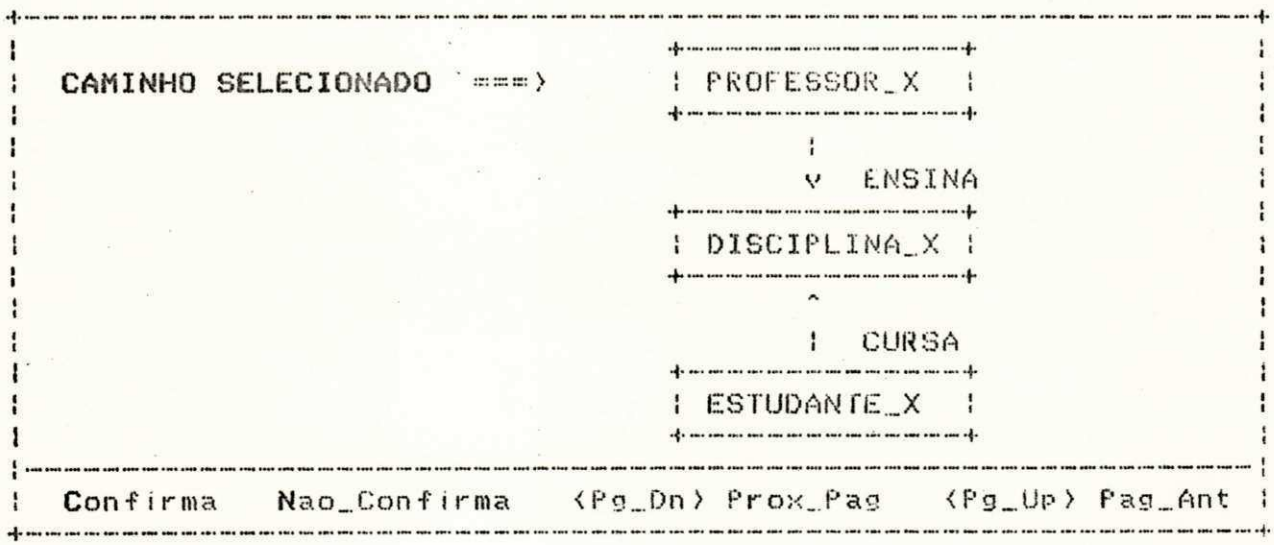


figura 3.18 - TELA DE CONFIRMAÇÃO DO CC SELECIONADO

Com **CC** selecionado e confirmado, o otimizador da consulta (ver capítulo 6) otimizará este caminho para:

PROFESSOR_X - ENSINA --) DISCIPLINA_X - E_CURSADA --) ESTUDANTE_X

Para satisfazer a consulta é preciso,agora, definir a **Lista-Alvo** (no exemplo, **NUM_PROF** e **NOME_PROF**), bem como, o **Predicado da Consulta.** Vejamos as telas fornecidas pelo sistema, para a definição destes elementos em PROFESSOR_X, nas figuras 3.19 e 3.20 respectivamente.

```
+-------------------------------------------------------------------+
|  Recupere:                                                        |
|                                                                   |
|    Marque atributos de PROFESSOR_X a serem recuperados:           |
|                                                                   |
|        NUM_PROF                                                   |
|   ===) NOME_PROF                                                  |
|        ENDERECO                                                   |
|                                                                   |
|                                                                   |
|-------------------------------------------------------------------|
| Para_Escolha Sair_da_Consulta  Marca_Atributo  ↑ Subir            |
| ↓ Descer  (Pg_Up) Pag_Ant   (Pg_Dn) Prox_Pag   Desmarca_Atributo  |
+-------------------------------------------------------------------+
```

FIGURA 3.19 - TELA DE DEFINIÇÃO DA LISTA_ALVO

```
+-------------------------------------------------------------------+
|   Expressão para seleção de Atributos de PROFESSOR_X              |
|                                                                   |
|     NUM_PROF  ) 68;                                               |
|                                                                   |
|                                                                   |
|-------------------------------------------------------------------|
| NUM_PROF      |              |              |                     |
| NOME_PROF     |              |              |                     |
| ENDERECO      |              |              |                     |
|-------------------------------------------------------------------|
|   FIM DE ATRIBUTOS. APERTE UMA TECLA                              |
+-------------------------------------------------------------------+
```

FIGURA 3.20 - TELA DE DEFINIÇÃO DO PREDICADO DA CONSULTA.

Definidos a Lista-Alvo e o PC, é preciso, agora, indicar os meios de saída do RC, bem como, um nome para os arquivos que

guardarão o programa e os dados gerados pela consulta. A tela destas definições se encontra na figura 3.21.

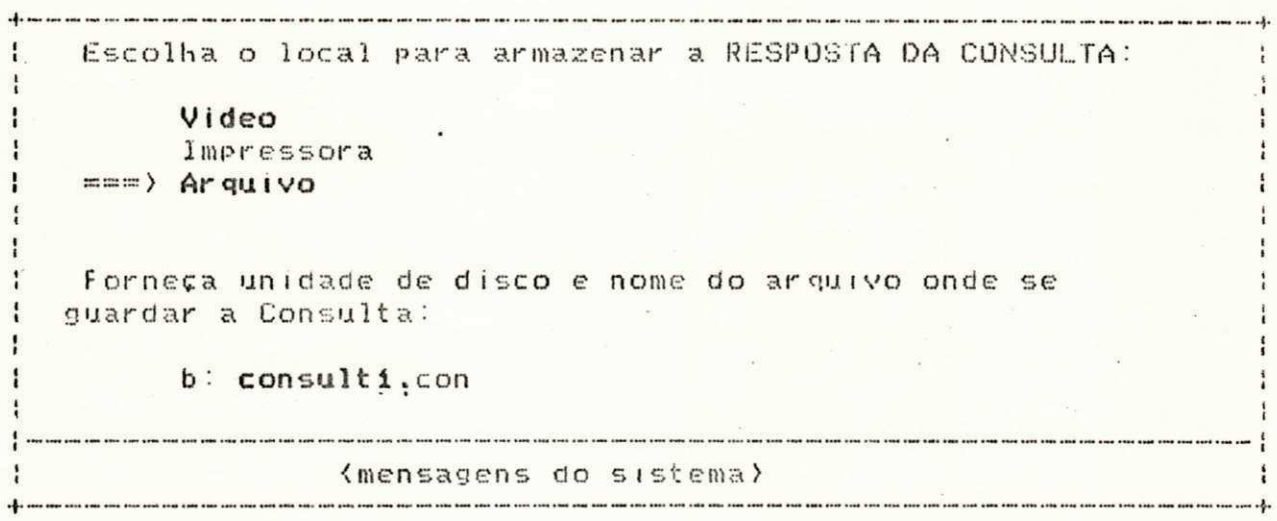


FIGURA 3.21 - TELA DE DEFINIÇÃO DOS MEIOS DE SAIDA DA CONSULTA E DO ARQUIVO DE CATALOGAÇÃO DA CONSULTA

d)Editar o Texto de uma Consulta

Ao encerrar-se a definição de uma consulta,o sistema permite ao usuário escrever um texto, de no máximo 15 linhas, descrevendo a consulta realizada. Este texto será catalogado e todas as vezes que se executar a consulta gerada ele será apresentado no vídeo.

O editor de texto, que realiza esta tarefa, é simples e apresenta um conjunto mínimo de comandos de edição, tais como, ^S (Retorna um caracter a partir do cursor),^D (Avança um caracter a partir do cursor), ^A (Vai para o início da linha do cursor),etc.

e)Detectando os Erros na Elaboração da Consulta

Este sub-módulo segue o mesmo padrão de monitoriamento de erros descrito na sub-seção 3.1.6. A definição de uma consulta,

via LIMADEX, apresenta poucos pontos de captação de informações incorretas. Os únicos pontos possíveis de erros são: na definição da expressão de seleção (operador inválido, tipo de operação inválida, atributo inexistente, etc) e na definição do arquivo da consulta(nome do arquivo inexistente,unidade não disponível,etc).

- **Operar Arquivos-Resultados**

Esta sub-operação permite que saídas de operações de consultas guardadas em arquivos possam ser manipuladas formando uma nova consulta, devendo existir compatibilidade entre as **Listas-Alvos** envolvidas[10].

Três operadores podem ser utilizados para operar arquivos-resultados:

a) **INTERSEÇÃO (&)** - seleciona os elementos cujos valores são comuns entre os arquivos-resultados envolvidos na operação.

b) **UNIÃO (!)** - seleciona os elementos cujos valores aparecem em um ou outro arquivo-resultado envolvido na operação.

c) **DIFERENÇA (-)** - seleciona os elementos cujos valores aparecem em um e não aparecem no outro arquivo-resultado envolvido na operação.

Estes tipos de operadores são importantes, pois permitem a realização de consultas que, para se concretizarem, necessitam de vários caminhos que possuam listas-alvos comuns.

Para realizarmos esta sub-operação, definimos,primeiramente, a expressão envolvendo os arquivos-resultados, cuja a BNF é:

⟨expr_arqresultado⟩ := ⟨expressão⟩;

⟨expressão⟩ := ⟨expressão⟩ ! ⟨termo⟩ / ⟨expressão⟩ - ⟨termo⟩ /

⟨termo⟩

⟨termo⟩ := ⟨termo⟩ & ⟨fator⟩ / ⟨fator⟩

⟨fator⟩ := (⟨expressão⟩) / ⟨identificador⟩

⟨identificador⟩ := nome_arquivo_resultado

Definida a expressão, definem-se os meios de saída, os arquivos da consulta gerada e um texto explicativo.

Por exemplo, se tivéssemos três arquivos ARQ1, ARQ2 e ARQ3 compatíveis entre si e desejássemos criar um quarto arquivo resultado ARQ4 (baseado na operação **ARQ1 & (ARQ2 ! ARQ3) - (ARQ1 & ARQ2)**), teríamos que utilizar a seguinte sequência de telas, conforme mostra a figura 3.22.

TELA 1

```
+-----------------------------------------------------------------------+
|  Escreva a operação:                                                  |
|                                                                       |
|  ARQ1 & (ARQ2 ! ARQ3) - (ARQ1 & ARQ2)                                |
|                                                                       |
|                                                                       |
|-----------------------------------------------------------------------|
|                  (mensagens do sistema)                               |
+-----------------------------------------------------------------------+
                              |
                              V
                            TELA2
```

TELA 2

```
+-----------------------------------------------------------------------+
|  Escolha o local para armazenar a RESPOSTA DA CONSULTA:               |
|                                                                       |
|       Video                                                           |
|       Impressora                                                      |
|  ===> Arquivo                                                         |
|                                                                       |
|  Forneça a unidade de disco e nome do arquivo onde se                 |
| guardar a Consulta:                                                   |
|                                                                       |
|       b: ARQ4     .con                                                |
|                                                                       |
|-----------------------------------------------------------------------|
|                 <mensagens do sistema>                                |
+-----------------------------------------------------------------------+
```

FIGURA 3.22 - TELAS DE DEFINIÇÕES DA OPERAÇÃO OPERAR ARQUIVOS RESULTADOS.

3.3.2.2) Atualização

Esta operação permite aos usuários **LG** e que possuam uma **visão irrestrita** de um conjunto de entidades,acessar os atributos armazenados no BD e realizar as modificações desejadas.

Para realizar esta operação, faz-se necessário definir os seguintes elementos:

a)**O Caminho da Atualização (CA)** - indica o conjunto de entidades que será atualizada (conjunto de entidades fonte), bem como, seus conjuntos de relacionamentos e conjuntos de entidades relacionados (se existir e for necessário) que influenciarão na operação de atualização. As telas fornecidas pelo sistema, para a definição do CA, são idênticas às telas 1 e 2 apresentadas na figura 3.17, sendo que, no caso da definição do **CA**, é permitida a escolha de mais de uma opção na tela 2.

b)**A População da Atualização (PA)** - é o sub-conjunto do conjunto de entidades fonte, que torna o **Predicado da Atualização**

(PDA) sempre verdadeiro. O PDA é uma expressão booleana com BNF igual a descrita na sub-seção 3.2.2.1, item "d", que envolve separadamente os atributos de conjuntos de entidades do CA. As telas fornecidas pelo sistema, para a definição do PDA, são idênticas a da figura 3.20. Apesar do PDA apresentar uma forma idêntica ao PDC, seu significado é diferente:

a) A expressão booleana definida no conjunto de entidades fonte indica que as entidades que satisfaçam a expressão poderão ter seus atributos atualizados.

b) A expressão booleana definida nos conjuntos de entidades relacionadas indica que as entidades do conjunto de entidades fonte que se relacionem com entidades que satisfaçam a expressão booleana poderão ter seus atributos atualizados.

Finalizando a operação, o sistema solicitará do usuário quais os atributos das entidades do conjunto de entidades fonte que poderão sofrer modificações. A tela fornecida pelo sistema para esta escolha é equivalente a mostrada na figura 3.19.

Ao encerrar a definição da operação, o sistema permitirá ao usuário a edição de um texto descrevendo-a, conforme visto na operação consulta.

De posse de todas as informações fornecidas pelo usuário, o sistema deverá gerar um programa na linguagem de manipulação do BD operacional, que realize as seguintes tarefas:

a) Seleciona os atributos das entidades do conjunto de entidades fonte que satisfaçam o PDA, expondo-os para o usuário.

b) Solicita ao usuário um novo valor para cada atributo das entidades do conjunto de entidades fonte. Vale salientar, que um <ENTER>, num dos atributos, significa que, este não será modificado.

c) Para cada grupo de valores de atributos de uma entidade do conjunto de entidades fonte, haja modificação nos atributos armazenados no BD.

3.3.2.3) Eliminação

Esta operação permite aos usuários **LG** e que possuam uma **visão irrestrita** de um conjunto de entidades, acessar os atributos armazenados no BD e eliminá-los.

Para realizar esta operação, faz-se necessário definir os seguintes elementos:

a)**O Conjunto de Entidades Fonte (CEF)** - indica o conjunto de entidades que terá algumas de suas entidades eliminadas. Vale salientar, que se um conjunto de entidades for escolhido como **CEF** e for **DESTINO** num conjunto de relacionamentos que possua uma **RIR**, a eliminação só será efetivada se pelo menos uma das entidades não for eliminada ou se a entidade do conjunto de entidades ORIGEM tenha sido eliminada primeiro. A tela fornecida pelo sistema para a definição do CEF é idêntica a tela 1 apresentada na figura 3.17.

b)**A População da Eliminação (PE)** - é o sub-conjunto do **CEF** , que torna o **Predicado da Eliminação (PDE)** sempre verdadeiro. O **PDE** é uma expressão booleana com BNF igual a descrita na sub-

seção 3.2.2.1, item "d", envolvendo atributos do CEF. As telas fornecidas pelo sistema para a definição do PDE são idênticas a da figura 3.20 e apresenta o seguinte significado: "elimine do CEF as entidades que tornem o PDE verdadeiro".

Finalizando a operação, o sistema solicitará ao usuário um texto descrevendo-a, conforme visto na operação consulta.

De posse de todas as informações fornecidas pelo usuário, o sistema deverá gerar um programa na linguagem de manipulação do BD Operacional, que realize as seguintes tarefas:

a) Seleciona as entidades do CEF, que satisfaçam o PDE, expondo-as para o usuário.

b) Verifica se a entidade pode ser eliminada e em caso afirmativo, pede confirmação do usuário.

c) Para cada entidade confirmada, desativa os seus atributos armazenados no BD.

3.3.2.4) Desconecção

Esta operação permite aos usuários LG e que possuam uma **visão irrestrita** de um conjunto de entidades, acessar as entidades armazenadas no BD e desrelacioná-las da entidade do conjunto de entidades origem com a qual estejam relacionadas.

Para realizar esta operação, faz-se necessário definir os seguintes elementos:

a)**O Caminho da Desconecção (CD)** - indica o conjunto de entidades que terá entidades desconectadas (conjunto de entidades

fonte), os seus conjuntos de relacionamentos e os conjuntos de entidades que sofrerão a desconecção. Vale salientar, que se o conjunto de relacionamentos envolvido possuir uma RIR, a desconecção só será efetuada, se existir outras entidades relacionadas com a entidade do conjunto de entidades ORIGEM. As telas fornecidas pelo sistema para a definição do **CD** são idênticas as telas 1 e 2 apresentadas na figura 3.17.

b)**A População da Desconecção (PD)** - é o sub-conjunto do conjunto de entidades fonte, que torna o **Predicado da Desconecção (PDD)** sempre verdadeiro. O **PDD** é uma expressão booleana com BNF igual a descrita na sub-seção 3.2.2.1, item "d", que envolve separadamente os atributos de conjuntos de entidades do **CD**. As telas fornecidas pelo sistema para definição do **PDD** são idênticas a da figura 3.20 e apresenta o seguinte significado: "Desconecte dos seus respectivos relacionamentos as entidades do conjunto de entidades fonte que tornem o **PDD** verdadeiro".

Finalizando a operação, o sistema solicitará ao usuário um texto descrevendo-a, conforme visto na operação consulta.

De posse de todas as informações fornecidas pelo usuário, o sistema deverá gerar um programa na linguagem de manipulação do BD Operacional, que realize as seguintes tarefas:

a) Seleciona as entidades do conjunto de entidades fonte, que satisfaçam PDD, expondo-as para o usuário.

b) Verifica se a entidade pode ser desconectada e em caso afirmativo, pede confirmação do usuário.

c) Desconecta cada entidade confirmada do seu respectivo

relacionamento com a entidade do conjunto de entidades **ORIGEM.**

3.3.2.5) Inserção

Esta operação permite aos usuários **LG** e que possuam uma **visão irrestrita** de um conjunto de entidades, criar novas entidades para este conjunto de entidades.

Para realizar esta operação, faz-se necessário definir os seguintes elementos:

a)**O Caminho da Inserção (CI)** - indica o conjunto de entidades que terá novas entidades inseridas (conjunto de entidades fonte), bem como, os seus conjuntos de relacionamentos e os conjuntos de entidades relacionadas (se existir e for necessário) que influenciarão na operação de inserção. As telas fornecidas pelo sistema para a definição do **CI** são idênticas as telas 1 e 2 apresentadas na figura 3.17, sendo que,no caso da definição do **CI** é permitido a escolha de mais de uma opção na tela 2. Vale ressaltar, que se o conjunto de entidades fonte for **ORIGEM** de um conjunto de relacionamentos que possua uma RIR,será obrigatório a presença deste no **CI**, bem como, a definição das entidades do conjunto de entidades DESTINO que se relacionarão com ela.

b)**A População da Inserção (PI)** - é um sub-conjunto de todos os conjuntos de entidades que se relacionem com o conjunto de entidades fonte, que estejam no **CI** e que tornem o **Predicado da Inserção (PDI)** sempre verdadeiro. O **PDI** é uma expressão booleana com BNF igual a descrita na sub-seção 3.2.1.2, item 'd', envolvendo somente atributos de cada conjunto de entidades que se

relacione com o conjunto de entidades fonte separadamente. As telas fornecidas pelo sistema são equivalentes a da figura 3.20.

Finalizando a operação, o sistema solicitará ao usuário um texto descrevendo-a, conforme visto na operação consulta.

De posse de todas as informações fornecidas pelo usuário, o sistema deverá gerar um programa na linguagem de manipulação do BD operacional, que realize as seguintes tarefas:

a) Solicita dos usuários os dados dos atributos da nova entidade que será inserida no conjunto de entidades fonte.

b) Mostra o conjunto de entidades e suas entidades que se relacionarão com a nova entidade do conjunto de entidades fonte.

3.3.2.6) Conecção

Esta operação permite aos usuários LG e que possuam uma **visão irrestrita** de um conjunto de entidades, acessar as entidades armazenadas no BD e relacioná-las com outras entidades.

Para realizar-se esta operação faz-se necessário definir os seguintes elementos:

a)**O Caminho da Conecção (CCO)**- indica o conjunto de entidades que terá entidades conectadas (**conjunto de entidades fonte**), os seus conjuntos de relacionamentos e os conjuntos de entidades que sofrerão a conecção. As telas fornecidas pelo sistema para a definição do CC são equivalentes as telas 1 e 2 apresentadas na figura 3.17.

b)**A População da Conecção (PCO)**- é o sub-conjunto do conjunto

de entidades fonte, que torna o Predicado da Conecção (PDCO) sempre verdadeiro.O **PDCO** é uma expressão booleana com BNF igual a descrita na sub-seção 3.2.2.1, item 'd', que envolve separadamente os atributos de conjuntos de entidades do **CCO**. As telas fornecidas pelo sistema são idênticas as da figura 3.20 e apresenta o seguinte significado: "Conecte as entidades do conjunto de entidades fonte aos seus respectivos relacionamentos, que tornem o **PDCO** verdadeiro".

Finalizando a operação, o sistema solicitará ao usuário um texto descrevendo-a, conforme visto na operação consulta.

De posse de todas as informações fornecidas pelo usuário, o sistema deverá gerar um programa na linguagem de manipulação do BD Operacional, que realize as seguintes tarefas:

a) Seleciona as entidades do conjunto de entidades fonte, que satisfaçam PDCO, expondo-as para o usuário.

b) Verifica se a entidade pode ser conectada e em caso afirmativo, pede confirmação do usuário.

c) Conecta cada entidade confirmada ao seu respectivo relacionamento com a entidade do conjunto de entidades **ORIGEM**.

4.ESTRUTURAS INTERNAS DO ESQUEMA CONCEITUAL

Apesar da implementação da LINDEC não estar dentro do escopo deste trabalho, fez-se necessária a definição de suas estruturas internas, onde as informações do esquema estarão armazenadas, devido à necessidade de utilização destas informações na LINDEX e na consulta, via LIMADEX.

As estruturas internas, que suportam o esquema, são:

a) **Tabela de Esquemas de Dados**

b) **Tabela de Entidades dos Esquemas de Dados**

c) **Tabela de Atributos dos Esquemas de Dados**

d) **Tabela de Relacionamentos dos Esquemas de Dados**

e) **Tabela de Usuários dos Esquemas de Dados**

4.1 - A Tabela de Esquemas de Dados (tab_esq)

É a estrutura de dados que guarda as informações da definição de um esquema de dados. Apresenta o seguinte formato:

flag_esq	nome_esq	lst_reg	lst_rel	lst_usr
byte	string[8]	integer	integer	integer

FIGURA 4.1 - FORMATO DE UM REGISTRO DE TAB_ESQ

- Campo **flag_esq** - indica se o registro de tab_esq se encontra disponível ou não.

- Campo **nome_esq** - guarda o nome do esquema.

- Campo **lst_reg** - apontador para uma lista encadeada de registros, que guarda informações sobre os conjuntos de entidades definidos para o esquema (veja figura 4.2).

- Campo lst_rel - apontador para uma lista encadeada de registros, que guarda informações sobre os conjuntos de relacionamentos definidos para o esquema (veja figura 4.2).

- Campo lst_usr - apontador para uma lista encadeada de registros, que guarda informações sobre os usuários definidos para o esquema (veja figura 4.2).

Vale ressaltar, que, neste trabalho, a ordem dos elementos numa lista encadeada será sempre dada pela ordem de chegada dos elementos na lista.

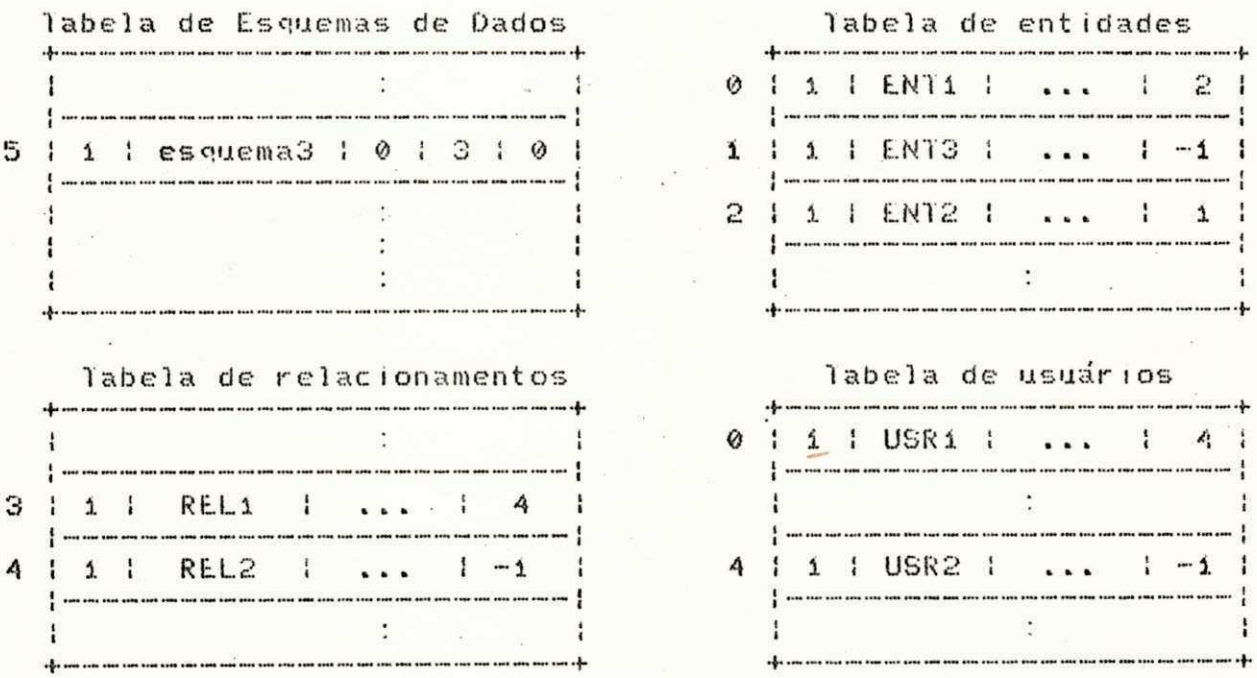


FIGURA 4.2 - EXEMPLO DAS ESTRUTURAS INTERNAS DE UM EC.

4.2 - A Tabela de Entidades dos Esquemas de Dados (tab_rege)

É a estrutura de dados que guarda as informações das definições dos conjuntos de entidades de um esquema de dados. Apresenta o seguinte formato:

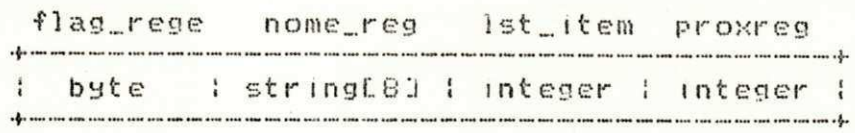


FIGURA 4.3 - FORMATO DE UM REGISTRO DE TAB_REGE

- Campo **flag_reg** - indica se o registro de tab_rege se encontra disponível ou não.

- Campo **nome_reg** - guarda o nome do conjunto de entidades.

- Campo **lst_item** - apontador para uma lista encadeada de registros, que guarda informações sobre os atributos que formam o conjunto de entidades (veja figura 4.4).

- Campo **proxreg** - apontador para o próximo registro de tab_rege,que guarda informações sobre outro conjunto de entidades que forma o esquema de dados.

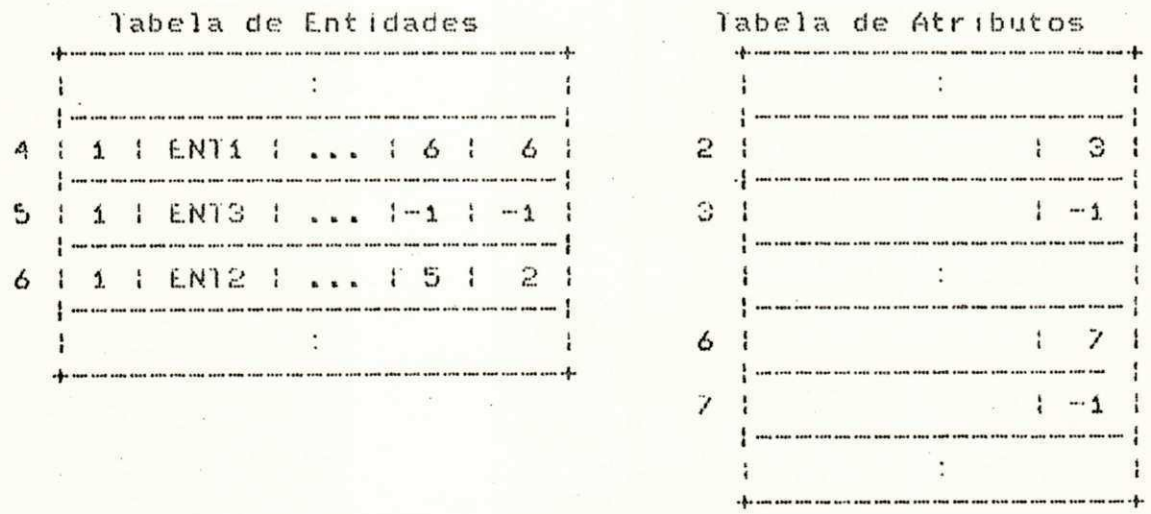


FIGURA 4.4 - EXEMPLO DA TAB_REGE E SUA INTERAÇÃO COM A TAB_ITE

4.3 - A Tabela de Relacionamentos dos Esquemas de Dados(tab_rele)

É a estrutura de dados que guarda as informações das definições dos conjuntos de relacionamentos que formam o esquema

de dados. Apresenta o seguinte formato:

flag_rele	nome_rel	nomeregd	nomeregm	ord_itmenb
byte	string[8]	string[8]	string[8]	string[8]

tipo	RIR	proxrel	vinculo
string[3]	char	integer	char

FIGURA 4.5 - FORMATO DE UM REGISTRO DE TAB_RELE

- Campo **flag_rele** - indica se o registro de tab_rele se encontra disponível ou não.

- Campo **nomerel** - guarda o nome do conjunto de relacionamentos (veja figura 4.6).

- Campo **nomeregd** - guarda o nome do conjunto de entidades ORIGEM do conjunto de relacionamentos (veja figura 4.6).

- Campo **nomeregm** - guarda o nome do conjunto de entidades DESTINO do conjunto de relacionamentos (veja figura 4.6).

- Campo **ord_itmemb**- guarda o nome do atributo do conjunto de entidades DESTINO, que será **CHAVE DE ORDENAÇÃO** do conjunto de relacionamentos (veja figura 4.6).

- Campo **tipo** - indica o tipo do conjunto de relacionamentos (**1:1** ou **1:n**) (veja figura 4.6).

- Campo **RIR** - indica a existência ou não de uma RIR no conjunto de relacionamentos (veja figura 4.6).

- Campo **proxrel** - apontador para o próximo registro de tab_rele, que guarda informações sobre outro conjunto de relacionamentos que forma o esquema de dados (veja figura 4.6).

- Campo **vinculo** - indica se o conjunto de relacionamentos

faz parte ou não de um par de conjuntos relacionamentos 1:n opostos e dependentes no esquema (veja figura 4.6).

Tabela de Relacionamentos

	:								
3	1	REL1	ENT1	ENT2	ATR21	1:n	s	4	i
4	1	REL2	ENT2	ENT1	ATR11	1:n	s	7	i
	:								
7	1	REL3	ENT1	ENT1	ATR11	1:1	n	-1	i
	:								

FIGURA 4.6 - EXEMPLO DE TAB_RELE

4.4 - Tabela de Atributos dos Esquemas de Dados(tab_ite)

É a estrutura de dados que guarda as informações das definições dos atributos que formam os conjuntos de entidades. Apresenta o seguinte formato:

flag_ite	nomeitem	tipoitem	tam_item	proxitem	RIV
byte	string[8]	byte	integer	integer	char

FIGURA 4.7 - FORMATO DE UM REGISTRO DE TAB_ITE

- Campo **flag_ite** - indica se o registro de tab_ite se encontra disponível ou não.

- Campo **nome_item** - guarda o nome do atributo.

- Campo **tipoitem** - identifica qual o tipo de atributo (veja figura 4.8).

- Campo **tam_item** - identifica qual o tamanho, em bytes, do atributo (veja figura 4.8).

- Campo **proxitem** - apontador para o próximo registro de tab_ite, que guarda informações sobre outro atributo que forma o conjunto de entidades (veja figura 4.8).

- Campo **RIV** - indica a existência ou não uma RIV no atributo (veja figura 4.8).

Tabela de Atributos

		:					
2	1	ATR11	2	1	3	s	
3	1	ATR12	7	30	-1	n	
		:					
6	1	ATR21	2	2	7	s	
7	1	ATR22	5	1	-1	n	
		:					

FIGURA 4.8 - EXEMPLO DE TAB_ITE

4.5 - Tabelas de Usuários dos Esquemas de Dados (tab_usre)

E a estrutura de dados que guarda as informações sobre os usuários que serão donos do esquema de dados. Apresenta o seguinte formato:

flag_usre	nome_usre	senha	proxusr
integer	string[8]	string[8]	integer

FIGURA 4.9 - FORMATO DE UM REGISTRO DE TAB_USRE

- Campo **flag_usre** - indica se o registro de tab_usre se encontra disponível ou não.

- Campo **nome_usre** - guarda o nome do usuário do esquema.

- Campo **senha** - guarda o nome da senha do usuário.

- Campo **proxusr** - apontador para o próximo registro de tab_usre, que guarda informações sobre outro usuário, dono do esquema de dados.

A figura 4.10 mostra um exemplo da tabela de usuários do esquema de dados.

Tabela de Usuário

		:		
3	1	GIUSEPPE	SENHA1	6
		:		
		:		
	1	MARCUS	SENHA2	-1
		:		
		:		

FIGURA 4.10 - EXEMPLO DE TAB_USRE

5 - ASPECTOS DE IMPLEMENTAÇÃO DA LINDEX

A LINDEX e o módulo Consulta da LIMADEX foram implementados na linguagem PASCAL. As razões desta opção foram: ser a linguagem PASCAL uma linguagem que favorece a modularidade, a estruturação, a portabilidade e a eficiência de execução; apresentar uma interface com o SBD/TS; e a nossa fluência na sua utilização.

5.1 - Estruturas Internas dos Sub-Esquemas

As informações sobre os sub-esquemas são guardadas em sete tabelas:

a) Tabela de Sub-Esquemas

b) Tabela de Entidades dos Sub-Esquemas

c) Tabela de Atributos dos Sub-Esquemas

d) Tabela de Relacionamentos dos Sub-Esquemas

e) Tabela de Usuários dos Sub-Esquemas

f) Tabela de Expressões e Tabela de Átomos dos Sub-Esquemas

5.1.1 - A Tabela de Sub-Esquemas (tab_sube)

É a estrutura de dados que guarda as informações da definição de um sub-esquema. Apresenta o seguinte formato:

flag_sesq	nome_sube	posusre	posesq
byte	string[15]	integer	integer

lst_regse	lst_relse	lst_usrse
integer	integer	integer

FIGURA 5.1- FORMATO DE UM REGISTRO DE TAB_SUBE.

- Campo **flag_sesq** - indica se o registro de tab_sube se encontra disponível ou não.

- Campo **nome_sube** - guarda o nome do sub-esquema.

- Campo **posusre** - apontador para o registro de tab_usre, que guarda as informações do usuário que criou o sub-esquema.

- Campo **posesq** - apontador para o registro de tab_esq, que possui as informações do esquema no qual o sub-esquema está vinculado (veja figura 5.2).

- Campo **lst_regse** - apontador para uma lista encadeada de registros, que guarda informações sobre os conjuntos de entidades definidos para o sub-esquema (veja figura 5.2).

- Campo **lst_relse** - apontador para uma lista encadeada de registros, que guarda informações sobre os conjuntos de relacionamentos definidos para o sub-esquema (veja figura 5.2).

- Campo **lst_usrse** - apontador para uma lista encadeada de registros, que guarda informações sobre os usuários do sub-esquema (veja figura 5.2).

Tabela de Sub-Esquemas

4	1	SUB-ESQUEMA1	0	2	3	4	0

Tabela de Usuários (Esquema)

0	1	USR1	...	9
9	1	USR2	...	-1

Tabela de Esquemas

2	1	ESQUEMA1	...

Tabela de Usuários (Sub-Esquema)

0	1	USUARIO0	...	1
1	1	USUARIO1	...	-1

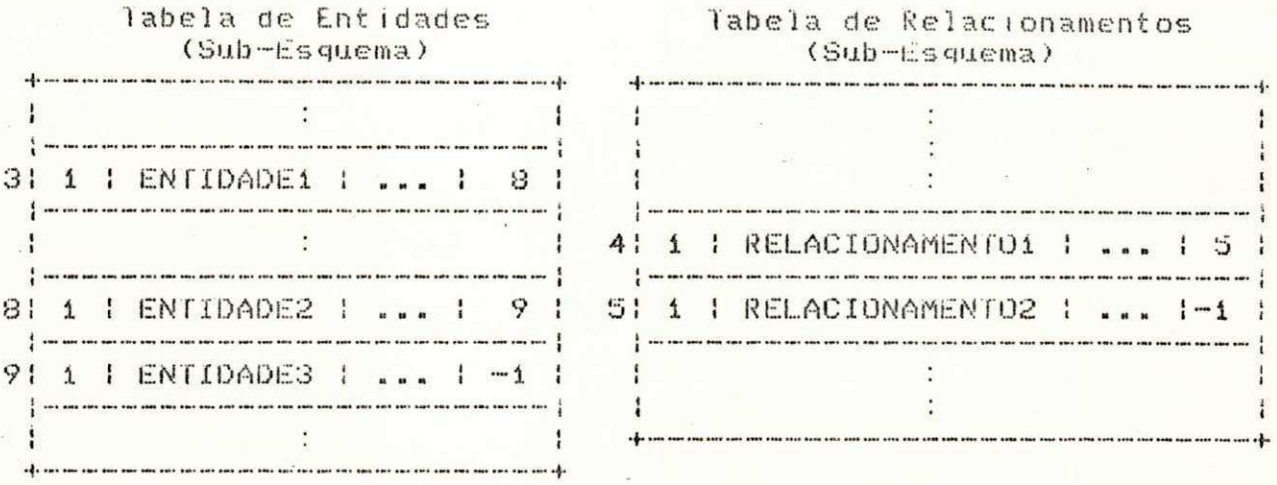


FIGURA 5.2 - EXEMPLO DE INTEGRAÇÃO ENTRE AS ESTRUTURAS INTERNAS DO SUB-ESQUEMA.

5.1.2 - A Tabela de Entidades dos Sub-Esquemas (tab_regse)

É a estrutura de dados que guarda as informações da definição dos conjuntos de entidades de um sub-esquema. Apresenta o seguinte formato:

flag_regse	nom_regse	pos_reg	lstitemse	proxregse
byte	string[15]	integer	integer	integer

lst_expr	lst_átomo	restrito
integer	integer	char

FIGURA 5.3 - FORMATO DE UM REGISTRO DE TAB_REGSE

- Campo **flag_regse**- indica se o registro de tab_regse se encontra disponível ou não.

- Campo **nom_regse** - guarda o nome do conjunto de entidades.

- Campo **pos_rege** - apontador para o registro de tab_rege, que guarda as informações do conjunto de entidades do esquema no

qual o conjunto de entidades do sub-esquema está vinculado (veja figura 5.4).

- Campo **lstitemse** - apontador para uma lista encadeada de registros, que guarda informações sobre os atributos que formam o conjunto de entidades do sub-esquema (veja figura 5.4).

- Campo **proxregse** - apontador para o próximo registro de tab-regse, que guarda informações sobre o outro conjunto de entidades que forma o sub-esquema (veja figura 5.4).

- Campo **lst_expr** - apontador para uma lista encadeada de registros, que guarda informações sobre as expressões que definem a projeção horizontal sobre os atributos do conjunto de entidades correspondente no esquema de dados (veja figura 5.4).

- Campo **lst_átomo** - apontador para uma lista encadeada de registros, que guarda informações sobre os átomos (<atributo> <operador-relacional> <valor>) que compõem as expressões da projeção horizontal (veja figura 5.4).

- Campo **restrito** - indica se o conjunto de entidades do sub-esquema sofreu ou não projeção vertical (veja figura 5.4).

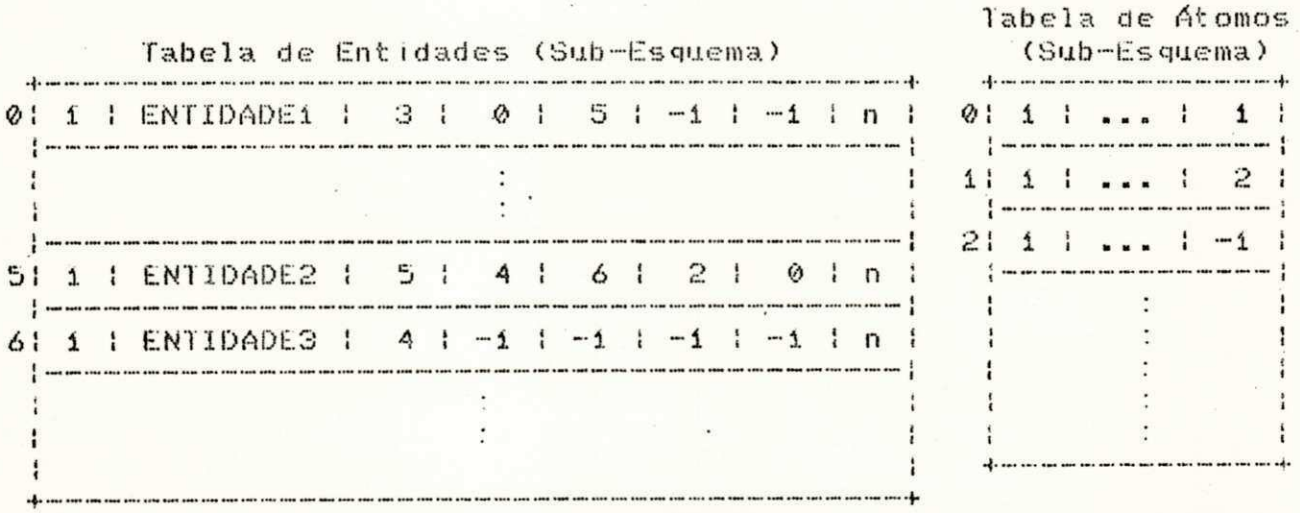


FIGURA 5.4 - EXEMPLO DE TAB_REGSE E SUA INTERAÇÃO COM TAB_ITSE, TAB_EXPR E TAB_ÁTOMOS

5.1.3 - A Tabela de Atributos dos Sub-Esquemas (tab_itse)

É a estrutura de dados que guarda as informações das definições dos atributos que formam os conjuntos de entidades do sub-esquema. Apresenta o seguinte formato:

flagitem	nom_itse	pos_ite	proxitse
byte	string[15]	integer	integer

FIGURA 5.5 - FORMATO DE UM REGISTRO DE TAB_ITSE

- Campo **flagitem** - indica se o registro de tab_itse se encontra disponível ou não.

- Campo **nom_itse** - guarda o nome do atributo.

- Campo **pos_ite** - apontador para o registro de tab_ite, que guarda as informações do atributo do esquema no qual o atributo do sub-esquema está vinculado (veja figura 5.6).

- Campo **proxitse** - apontador para o próximo registro de

tab-itse, que guarda informações sobre outro atributo do conjunto de entidades do sub-esquema (veja figura 5.6).

FIGURA 5.6 - EXEMPLO DE TAB_ITSE

5.1.4 - Tabela de Relacionamentos do Sub-Esquema (tab_relse)

É a estrutura de dados que guarda as informações da definição dos conjuntos de relacionamentos que formam o sub-esquema. Apresenta o seguinte formato:

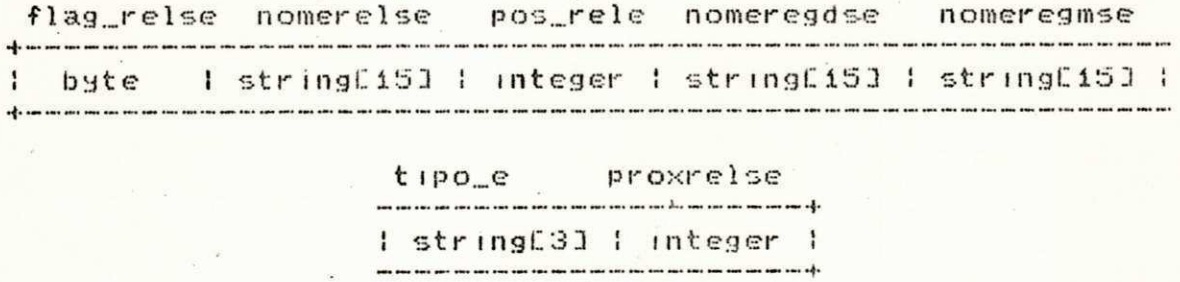


FIGURA 5.7 - FORMATO DE UM REGISTRO DE TAB-RELSE

- Campo **flag_relse**- indica se o registro de tab-relse se encontra disponível ou não.

- Campo **nomerelse** - guarda o nome do conjunto de relaciona-

mentos do sub-esquema.

- Campo **pos_rele** - apontador para o registro de tab-rele, que guarda as informações do conjunto de relacionamentos do esquema no qual o conjunto de relacionamentos do sub-esquema está vinculado (veja figura 5.8).

- Campo **nomeregdse**- guarda o nome do conjunto de entidades ORIGEM do conjunto de relacionamentos do sub-esquema (veja figura 5.8).

- Campo **nomeregmse**- guarda o nome do conjunto de entidades DESTINO do conjunto de relacionamentos do sub-esquema (veja figura 5.8).

- Campo **tipo_e** - guarda o tipo do conjunto de relacionamentos do esquema (veja figura 5.8).

- Campo **proxrelse** - apontador para o próximo registro de tab_relse,que guarda informações sobre outro conjunto de relacionamentos que forma o sub-esquema (veja figura 5.8).

Tabela de Relacionamentos do Sub-Esquema

4	1	RELACIONAMENTO1	3	ENTIDADE1	ENTIDADE2	1:n	5
5	1	RELACIONAMENTO2	4	ENTIDADE2	ENTIDADE1	1:n	-1

FIGURA 5.8 - EXEMPLO DE TAB_REGSE

5.1.5 - Tabela de Usuários do Sub-Esquema (tab_usrse)

É a estrutura de dados que guarda as informações da

definição dos usuários do sub-esquema. Apresenta o seguinte formato:

flagse	nom_usre	senha_se	pos_usre	tipo_usr	prox_usrse
byte	string[15]	string[15]	integer	byte	integer

FIGURA 5.9 - FORMATO DE UM REGISTRO DE TAB- USRSE

- Campo **flag_usrse**- indica se o registro de tab_usrse se encontra disponível ou não.

- Campo **nom_usrse** - guarda o nome do usuário do sub-esquema (veja figura 5.10).

- Campo **senha_se** - guarda a senha do usuário do sub-esquema (veja figura 5.10).

- Campo **pos_usre** - apontador para o registro de tab-usre, que guarda as informações do usuário do esquema no qual o usuário do sub-esquema está vinculado (veja figura 5.10).

- Campo **tipo_usr** - identifica o tipo do usuário do sub-esquema (veja figura 5.10).

- Campo **prox_usrse**- apontador para o próximo registro de tab_usrse, que guarda informações sobre outro usuário do sub-esquema.

Tabela de Usuários do Sub-Esquema

0	1	USUÁRIO1	SENHASE1	0	1	8
		:				
8	1	USUÁRIO2	SENHASE2	0	1	-1
		:				

FIGURA 5.10 - EXEMPLO DE TAB_USRSE

5.1.6 - Tabela de Expressões (tab_expr)
Tabela de Átomos (tab_atomos)

Ambas, conjuntamente, são estruturas de dados que guardam as informações da definição da projeção horizontal.

Como vimos no capítulo 3, a projeção horizontal é feita usando uma expressão booleana. Entretanto, a maneira como o usuário apresenta a expressão não implica que seja na forma do seu armazenamento. Optou-se em guardar a expressão da seguinte forma:

⟨expressão_and1⟩ ! ⟨expressão_and2⟩ !...! ⟨expressão_andN⟩

onde: cada **⟨expressão_andi⟩** faz parte de uma lista encadeada em tab_expr, cujos elos de ligação representam o conectivo ! (**or**) dentro do conjunto de entidades. Cada **⟨expressão_andi⟩**, apresenta a seguinte forma:

⟨átomo1⟩ & ⟨átomo2⟩ &...& ⟨átomoM⟩

onde: cada **⟨átomoj⟩** faz parte de uma lista encadeada em tab_átomos e apresenta a seguinte forma:

⟨atributo_esquema⟩ ⟨operador_relacional⟩ ⟨valor⟩

As razões que nos levaram a estabelecer esta forma de armazenamento da expressão foram:

1 - Eliminar a necessidade de guardar os parênteses e os operadores relacionais das expressões, pois a ordem dos elementos na lista encadeada de tab_expr já estabelece, implicitamente, os operadores e as prioridades de execução das operações.

2 - Facilitar a interpretação da expressão, usando a

leitura sequencial da lista encadeada de tab_expr.

A tabela de expressões apresenta o seguinte formato:

flag_aval	conj_int	prox_exp
byte	set of integer	integer

FIGURA5.11 - FORMATO DE UM REGISTRO DE TAB_EXPR

- Campo **flag_aval** - indica se o registro de tab_expr se encontra disponível ou não.

- Campo **conj_int** - representa a <expressão_and> através de um conjunto de números inteiros (0 a 255), que indicam a posição dos átomos, envolvidos na <exprssão_and>, na tabela de átomos (veja figura 5.13). Cada átomo, em conj_int, está relacionado com outros átomos (caso existam) através do conectivo & (and).

- Campo **prox_exp** - apontador para o próximo registro de tab_expr, que guarda informações sobre outra <expressão_and>. Caso exista, seu significado será o do conectivo | (or) (veja figura 5.13).

A tabela de átomos apresenta o seguinte formato:

flag_at	nom_item	operador	valor	prox_átomo
byte	string[8]	string[2]	string[80]	integer

FIGURA 5.12 - FORMATO DE UM REGISTRO DE TAB_ATOMOS

- Campo **flag_at** - indica se o registro de tab_átomos se encontra disponível ou não.

- Campo **nom_item** - identifica o atributo participante do

⟨átomoi⟩ (veja figura 5.13).

- Campo **operador** - guarda o operador relacional envolvido no ⟨átomoi⟩ (veja figura 5.13).

- Campo **valor** - guarda o valor utilizado no ⟨átomoi⟩ na comparação com o atributo. Este valor, na versão atual do nosso trabalho, deve ser uma constante (veja figura 5.13).

- Campo **prox_átomo**- apontador para o próximo registro de tab_átomos, que guarda informações sobre outro átomo que forma a expressão booleana da projeção horizontal de um conjunto de entidades (veja figura 5.13).

Vejamos, na figura 5.13, como ficaria armazenada a seguinte expressão :

(ATRIBUTO21 ⟩ 200 ! ATRIBUTO22 ⟨⟩ FALSO) & ATRIBUTO21 ⟨ 120;

Tabela de Expressões do Sub-Esquema

2	1	0, 2	3
3	1	1, 2	-1

Tabela de Átomos do Sub-Esquema

0	1	ATRIBUTO21	⟩	200	1
1	1	ATRIBUTO22	⟨⟩	FALSO	2
2	1	ATRIBUTO21	⟨	120	-1

FIGURA 5.13 - REPRESENTAÇÃO INTERNA DA EXPRESSÃO (ATRIBUTO21 ⟩ 200 ! ATRIBUTO22 ⟨⟩ FALSO) & ATRIBUTO21 ⟨ 120; EM TAB_EXPR E TAB_ATOMOS.

5.2 - Implementação da LINDEX

Vamos analisar, agora, o funcionamento básico da LINDEX, através da descrição em forma de algoritmo (em alto nível) dos principais módulos que a compõem.

5.2.1 - O Módulo Principal

É o módulo que implementa o gerenciador de sub-esquemas. Apresenta o seguinte algoritmo:

```
início (módulo principal)
  Obtém identificação do usuário e do esquema.
  Verifica veracidade dos dados
  Se ocorreu erro então encerre sessão
  Senão
    Enquanto não fim de tarefa faça
      Mostra menu principal
      Lê opção
      Caso opção seja
          1 - Cria Sub-Esquema
          2 - Elimina Sub-Esquema
          3 - Elimina/Inclui usuários
          4 - Lista Sub-Esquema
          5 - Modifica Sub-Esquema
          6 - Muda de Esquema
          7 - Fim da tarefa
      fim caso
    fim enquanto
  fim se
fim algoritmo
```

5.2.2 - O Módulo 'Cria Sub-Esquema'

É o módulo que implementa a tarefa de criar um novo sub-esquema. Apresenta o seguinte algoritmo:

```
início (módulo cria sub-esquema)
  Obtém nome do sub-esqema
  Verifica veracidade dos dados
  Se ocorreu erro então retorna menu principal
  Senão
    Cria entidades
    Se ocorreu erro ou desistência então retorna menu principal
    Senão
      Cria relacionamentos
      Se ocorreu erro ou desistência então
         retorna menu principal
      Senão
        Cria usuários
       Se ocorreu erro ou desistência então
         retorna menu principal
       Senão
        Enquanto não fim de tarefa faça
          Lê opção
          Caso opção seja
                1 - Modifica Sub-Esquema
                2 - Lista Sub-Esquema
                3 - Guarda Sub-Esquema
                4 - Fim da tarefa
          fim caso
          Se não gravou então
            Pergunta (quer gravar?)
            Se sim então Grava Sub-Esquema
```

```
            Senão Restabelece Tabelas-LINDEX
          fim se
        fim enquanto
    fim SE'S
  fim Cria Sub-Esquema
```

5.2.2.1 - O Sub-Módulo 'Cria Entidades'

É o sub-módulo que implementa a tarefa de criar novos conjuntos de entidades para o sub-esquema, bem como, os atributos que interessam (projeção vertical) e a expressão booleana que define a projeção horizontal.

Vale salientar que, em qualquer parte da LINDEX, optou-se pela pesquisa sequencial dos elementos de qualquer das tabelas de listas encadeadas existentes. Esta escolha se deu devido a duas razões principais: a primeira, por ser um método de pesquisa muito simples e portanto de fácil implementação e a segunda, por manipularmos com listas encadeadas pequenas, pois, por maior que seja um sub-esquema, ele nunca terá um número muito grande de componentes.

Apresentamos a seguir, o algoritmo deste sub-módulo:

```
início {sub-módulo Cria entidade}
  Enquanto não fim da tarefa faça
    Obtém nome_entidade_sub_esquema, nome_entidade_esquema
    Se nome_entidade_sub_esquema vazio então
      fim da tarefa
    Senão
      Verifica veracidade dos dados
      Se não ocorreu erro então
        Enquanto existir atributos na entidade do esquema faça
          Lista atributo da entidade do esquema
          Obtenhe nome_do_atributo_sub_esquema
          Verifica veracidade dos dados
          Se não erro então
            Encontra próximo atributo
        fim enquanto
        Obtenhe expressão booleana
        Guarda conjunto de Entidades do sub-esquema
      fim se
    fim se
  fim enquanto
fim Cria entidade
```

5.2.2.2 - O Sub-Módulo 'Cria Relacionamento'

Este sub-módulo implementa a tarefa de criação de novos conjuntos de relacionamentos para o sub-esquema. Apresenta o seguinte o algoritmo:

```
início {sub-módulo Cria Relacionamentos}
  Enquanto não fim da tarefa faça
    Obtém nome_relacionamento_sub_esquema
    Se nome_relacionamento_sub_esquema vazio então
      fim da tarefa
    Senão
      Verifica veracidade dos dados
      Se não ocorreu erro então
        Obtenhe nome dos conjuntos de Entidades ORIGEM e DESTINO
        Verifica veracidade dos dados
        Se não ocorreu erro então
          Procura conjunto de relacionamentos equivalente esquema
          Se encontrou mais de um então
            Obtenhe nome do conjunto de relacionamentos do esquema
            Verifica a veracidade dos dados
            Se não ocorreu erro então
              Mostra tipo do conjunto de relacionamentos
              Guarda conjunto de relacionamentos
          Senão
            Mostra tipo do conjunto de relacionamentos
            Guarda conjunto de relacionamentos
    fim SE's
  fim enquanto
fim Cria Relacionamentos
```

5.2.2.3 - **Sub-Módulo 'Cria Usuários'**

Este sub-módulo implementa a tarefa de criar novos usuários para o sub-esquema. Apresenta o seguinte algoritmo:

```
início (sub-módulo Cria Usuários)
  Enquanto não fim da tarefa faça
    Obtém nome e senha do usuário
      Se nome do usuário VAZIO então
        fim da tarefa
      Senão Verifica veracidade dos dados
            Se não ocorreu erro então
             Obtém tipo de usuário
             Guarda usuário
      fim se
  fim enquanto
fim cria usuários
```

5.2.3 - O Módulo 'Elimina Sub-Esquema'

É o módulo que implementa a tarefa de eliminar um sub-esquema existente. Apresenta o seguinte algoritmo:

```
início (módulo Elimina Sub-Esquema)
  Obtém nome do sub-esquema
  Verifica veracidade dos dados
  Se não ocorreu erro então
    Pergunta 'Confirma Eliminação?'
    Se sim então
      Elimina usuários
      Elimina relacionamento
      Elimina entidades/atributos
      Elimina sub-esquema
  fim SE'S
fim Elimina Sub-Esquema
```

5.2.4 - O Módulo "Elimina/Inclui Usuários"

É o módulo que implementa a tarefa de eliminar ou incluir um usuário no sub-esquema. Apresenta os seguintes algoritmos:

```
1)   início (módulo Elimina/Inclui Usuários)
       Obtém nome do sub-esquema
       Verifica veracidade dos dados
       Se não ocorreu erro então
         Lê opção
         Caso opção seja
           1 - Elimina Usuários
           2 - Cria Usuários
         fim caso
       fim se
     fim Elimina/Inclui Usuários

2)   início (sub-módulo Elimina Usuários)
       Enquanto não fim da tarefa faça
         Obtém nome e senha do usuário
         Se nome do usuário VAZIO então
           fim da tarefa
         Senão
           Verifica veracidade dos dados
           Se não ocorreu erro então
             Elimina usuário
         fim SE's
       fim enquanto
     fim Elimina Usuários
```

5.2.5 - O Módulo 'Modifica Sub-Esquema'

Este módulo implementa a tarefa de realizar modificações nos componentes de um sub-esquema. Apresenta o seguinte algoritmo:

```
início {módulo Modifica Sub-Esquema}
    Obtém nome sub-esquema
    Verifica veracidade dos dados
    Se ocorreu erro então encerre sessão
    Senão
      Enquanto não fim da tarefa faça
        Lê opção
        Caso opção seja
          1 - Muda Nome Sub-Esquema
          2 - Modifica Entidades
          3 - Modifica Relacionamentos
          4 - Modifica Usuários
          5 - Fim da Tarefa
        fim caso
      fim enquanto
    fim se
  fim modifica sub-esquema
```

O sub-módulo 'Modifica Entidades' é reponsável pela tarefa de realizar modificações nos conjuntos de entidades e seus atributos. Este sub-módulo dispõe das seguintes tarefas: Mudar o nome dos conjuntos de entidades; Incluir conjuntos de entidades; Eliminar conjuntos de entidades e Modificar os atributos de um conjunto de entidades. Todas estas tarefas possuem comportamento

semelhante as descritas nos módulos anteriores, a exceção dos atos de mudar o nome e eliminar um conjunto de entidades, que provocam mudanças automáticas nos conjuntos de relacionamentos. Por exemplo, o ato de eliminar um conjunto de entidades provoca a eliminação dos conjuntos de relacionamentos que possuam o conjunto de entidades eliminado.

Para estas duas tarefas, teremos os seguintes algoritmos:

```
1) início (Elimina Conjuntos de Entidades)
     Enquanto não fim da tarefa faça
       Obtém nome do conjunto de entidades
       Se nome do conjunto de entidades VAZIO então
         fim da tarefa
       Senão
         Verifica veracidade dos dados
         Se não ocorreu erro então
           Elimina conjunto de entidades
           Para cada conjunto de relacionamentos faça
             Se conjunto de entidades pertence ao
                conjunto de relacionamentos então
               Elimina conjunto de relacionamentos
             fim se
           fim para
       fim SE's
     fim enquanto
   fim Elimina Conjunto de Entidades
```

```
2) início (Muda Nome dos Conjuntos de Entidades)
     Enquanto não fim da tarefa faça
       Obtém nome do conjunto de entidades
       Se nome do conjunto de entidades VAZIO então
         fim da tarefa
       Senão
         Verifica veracidade dos dados
         Se não ocorreu erro então
           Obtenhe nome do conjunto de entidades
           Verifica veracidades dos dados
           Se não ocorreu erro então
             Muda nome do conjunto de entidades
             Para cada conjunto de relacionamentos faça
               Se conjunto de entidades pertence ao
                  conjunto de relacionamentos  então
                 Muda nome do conjunto de entidades
                      no conjunto de relacionamentos
               fim se
             fim para
       fim SE's
     fim enquanto
   fim muda nome dos conjuntos de entidades
```

6 - ASPECTOS DE IMPLEMENTAÇÃO DA LIMADEX

6.1 - Estruturas Internas das Operações da LIMADEX

A realização de uma consulta (ou de qualquer outra operação) na LIMADEX, implica nas seguintes definições:

a) Um caminho da consulta (ou da operação)

b) A lista-alvo da consulta

c) O predicado da consulta (ou da operação)

As informações fornecidas por estas definições são guardadas nas seguintes tabelas:

a) **Tabela de Entidades do Caminho**

b) **Tabela de Atributos das Entidades**

c) **Tabela de Relacionamentos do Caminho**

d) **Tabela de Expressões**

e) **Tabela de Átomos e Tabela de Átomos Separados**

6.1.1 - Tabela de Entidades do Caminho (tab_rgesq)

É a estrutura de dados que guarda as informações dos conjuntos de entidades que formam o caminho da consulta. Apresenta o seguinte formato:

pont_rel	pos_regse	var_reg	tipo_reg	tem_bif	lst_it
integer	integer	string[4]	char	char	integer

pos_prsep	pos_ulsep	lst_expr	lst_átomos
integer	integer	integer	integer

FIGURA 6.1 - FORMATO DE UM REGISTRO DE TAB_RGESQ.

- Campo **pont_rel** - apontador para o próximo conjunto de relacionamentos do caminho, cujas as informações estão guardadas num registro da tabela de relacionamentos do caminho.

- Campo **pos_regse** - apontador para o registro de tab_regse, que guarda as informações do conjunto de entidades do sub-esquema que participa do caminho.

- Campo **var_reg** - identificador único gerado pelo sistema para o conjunto de entidades do caminho. Este identificador é importante, pois, o nosso modelo admite a definição de AUTO-RELACIONAMENTOS, o que pode provocar a existência, no caminho, de uma duplicidade de conjuntos de entidades.

- Campo **tipo_reg** - identifica se o conjunto de entidades é **origem** ou **destino** do próximo conjunto de relacionamentos do caminho.

- Campo **tem_bif** - identifica se o conjunto de entidades do caminho é um ponto de bifurcação para outros caminhos, que compõem o sub-esquema. Um caminho definido pelo usuário nunca será bifurcado, mas quando o sistema incorporar os outros caminhos existentes na definição do sub-esquema, a consulta poderá se tornar uma consulta com caminhos bifurcados.

- Campo **lst_it** - apontador para uma lista encadeada dos atributos do conjunto de entidades do caminho.

- Campo **lst_expr** - apontador para uma lista encadeada de registros, que guarda as informações sobre a expressão do conjunto de entidades, que participa do predicado da consulta.

- Campo **lst_átomos**- apontador para uma lista encadeada de registros, que guarda informações sobre os átomos (<atributo>

(operador_relacional) (valor)) do conjunto de entidades, que compõem a expressão do predicado da consulta.

- Campos **pos_prsep** e **pos_ulsep**- apontadores,respectivamente, para o primeiro e último átomos separados do conjunto de entidades do caminho, que participam de forma especial do predicado da consulta (esta forma será explicada adiante).

6.1.2 - Tabela de Atributos das Entidades do Caminho (tab_itesq)

É a estrutura de dados que guarda as informações dos atributos que formam os conjuntos de entidades do caminho. Apresenta o seguinte formato:

pos_itse	var_item	item_rec	ap_prox_it
integer	string[4]	char	integer

FIGURA 6.2 - FORMATO DE UM REGISTRO DE TAB_ITESQ.

- Campo **pos_itse** - apontador para o registro de tab_itse, que guarda as informações do atributo do conjunto de entidades do sub-esquema, presente no caminho.

- Campo **var_item** - identificador único para o atributo, gerado pelo sistema, cuja finalidade é garantir a individualidade no tratamento do atributo, em quàlquer ocasião, evitando a existência de ambiguidades.

- Campo **item_rec** - indica se o atributo do conjunto de entidades do caminho está ou não participando da Lista_Alvo.

- Campo **ap_prox_it**- apontador para o próximo registro de tab_itesq, que guarda as informações do próximo atributo do

conjunto de entidades do caminho.

6.1.3 - Tabela de Relacionamentos do Caminho (tab_rlesq)

É a estrutura de dados que guarda as informações dos conjuntos de relacionamentos que formam o caminho. Apresenta o seguinte formato:

pont_reg	pos_relsb
integer	integer

FIGURA 6.3 - FORMATO DE UM REGISTRO DE TAB_RLESQ.

- Campo **pont_reg** - apontador para a próxima entidade do relacionamento, que forma o caminho, cujas informações estão guardadas em tab_rgesq.

- Campo **pos_relsb** - apontador para o registro de tab_relse, que guarda as informações do conjunto de relacionamentos do sub-esquema, que participa do caminho.

6.1.4 - Tabela de Expressões (tab_expat)
Tabela de Átomos (tab_átomos)
Tabela de Átomos Separados (tab_expsep)

tab_expat e **tab_átomos** são estruturas de dados, que guardam as informações da definição do predicado da consulta, da mesma forma como **tab_expr** e **tab_átomos** guardam as informações da definição da projeção horizontal na LINDEX, apesar da diferença semântica, que ambos os tipos de expressão trazem embutidas no seu bojo.

A semântica da expressão que define o predicado de uma

consulta pode provocar a existência de um caso especial em que este predicado não poderá ser armazenado da mesma maneira como se armazena uma definição de uma projeção horizontal. Vamos analisar este caso através do seguinte exemplo de consulta baseada no sub-esquema da Figura 2.14: "Mostrar os professores do colégio X que ensinam as disciplinas FÍSICA e MATEMÁTICA". Portanto, temos a seguinte expressão para o conjunto de entidades DISCIPLINA:

NOME_DISC = 'FÍSICA' & NOME_DISC = 'MATEMÁTICA'

A semântica deste predicado exige que, para torná-lo verdadeiro, cada entidade professor do conjunto de entidades PROFESSOR, esteja relacionada com ambas as entidades disciplina (FÍSICA e MATEMÁTICA) do conjunto de entidades DISCIPLINA. Então, para que o predicado da consulta seja testado, neste caso, faz-se necessário que a expressão seja desmembrada em duas, para que ambas sejam analisadas individualmente para cada entidade professor.

Generalizando, temos que todas as vezes que o predicado condicionar a existência de dois ou mais átomos exclusivos numa expressão de uma entidade, o sistema terá que desmembrá-los e testá-los individualmente para cada elemento da entidade.

Portanto, **tab_expsep** é a estrutura de dados que guarda as informações dos átomos que serão testados separadamente da expressão geral. Apresenta o seguinte formato:

flag_atsep	átomo
byte	integer

FIGURA 6.4 - FORMATO DE UM REGISTRO DE TAB_EXPSEP

- Campo **flag_atsep**- indica se o registro de tab_expsep se encontra disponível ou não.

- Campo **átomo** - apontador para o registro de tab_átomos, que guarda as informações do átomo separado, que participa do predicado da consulta.

Vejamos na FIGURA 6.5, como ficaria o predicado da consulta mostrado anteriormente.

TAB_RGESQ

	pos_prsep	pos_ulsep	lst_expr	lst_átomos
3	...	0	1	-1	0

TAB_EXPSEP

0	1	0
1	1	1

TAB_EXPR

0	-1	NULL	-1
N	-1	NULL	-1

TAB_ÁTOMOS

0	1	NOME_DISC	=	'FÍSICA'	1
1	1	NOME_DISC	=	'MATEMÁTICA'	-1

FIGURA 6.5 - ESTRUTURA DE ARMAZENAMENTO DE PREDICADOS DA CONSULTA COM ÁTOMOS SEPARADOS.

A FIGURA 6.6 mostra como as informações de uma consulta são armazenadas nas tabelas descritas anteriormente. Como exemplo,

utilizaremos a consulta formulada no capítulo 3.

TABELA DE ENTIDADES DO CAMINHO

0	0	1	VR0	d	n	0	-1	-1	0	0
1	1	3	VR1	m	n	3	-1	-1	1	1
2	-1	4	VR2		n	6	-1	-1	2	2
					:					

TABELA DE ATRIBUTOS

0	0	VI0	5	1
1	2	VI1	5	2
2	5	VI2	n	-1
3	1	VI3	n	4
4	6	VI4	n	5
5	3	VI5	n	-1
6	4	VI6	n	7
7	7	VI7	n	8
8	8	VI8	n	-1
		:		

TABELA DE RELACIONAMENTOS DO CAMINHO

0	2	3
1	-1	1
	.	
	.	
	.	

TABELA DE ÁTOMOS

0	NUM_PROF	>	68	-1
1	DURACAO	>	2	-1
2	NUM_ESTUD	>	121	3
3	NUM_ESTUD	<	212	-1
		:		

TABELA DE EXPRESSÕES

0	1	0	-1
1	1	1	-1
2	1	2,3	-1
		.	
		.	

FIGURA 6.6 - INTEGRAÇÃO DAS ESTRUTURAS DE DADOS DE UMA CONSULTA.

6.2 - O Processador de Consultas

Uma consulta na LIMADEX é processada conforme mostra a figura 6.7.

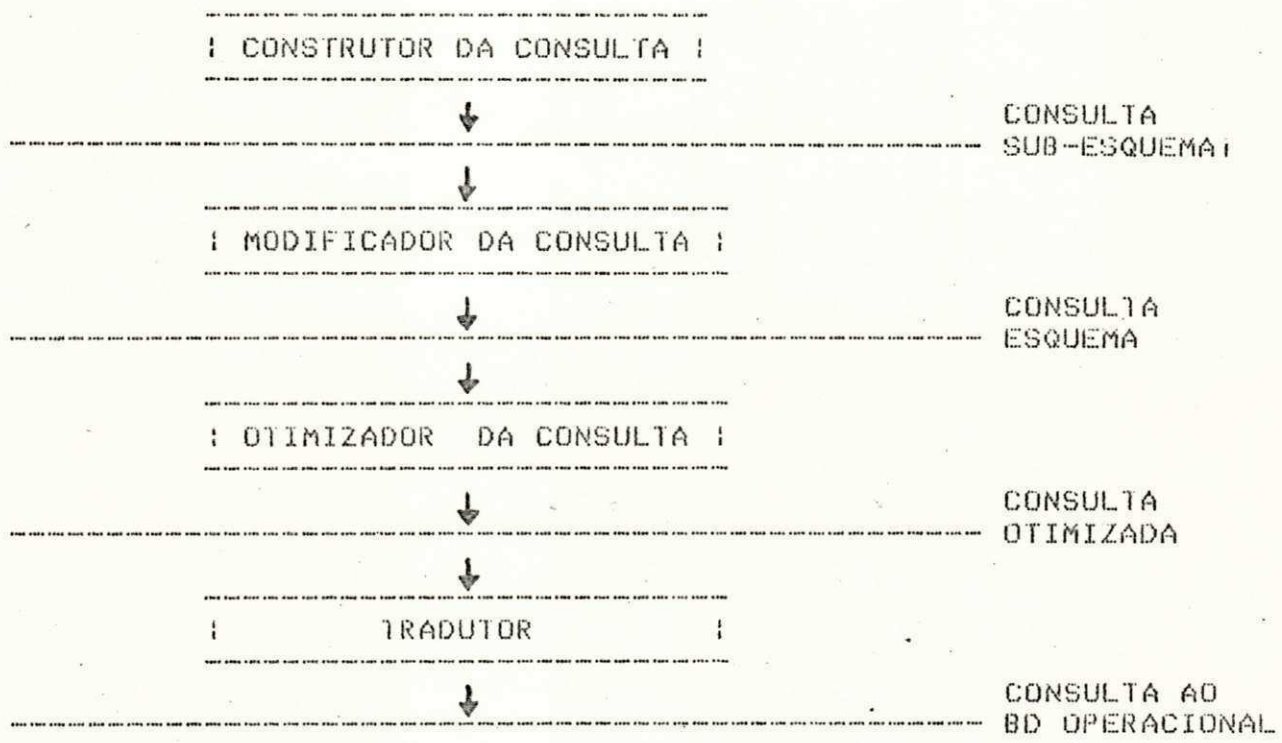


FIGURA 6.7 - O PROCESSADOR DA CONSULTA

Nesta sub-seção, analisaremos o funcionamento básico de cada um dos módulos que compõem o processador de consultas, com exceção do TRADUTOR, que deixaremos para analisá-lo no próximo capítulo.

6.2.1 - O Construtor da Consulta

É o módulo que implementa a construção das tabelas que descrevem a consulta e incorpora os conjuntos de relacionamentos e entidades do sub-esquema, que embora não tenham sido definidos no caminho da consulta, terão que participar da consulta, devido

a existência das projeções horizontais, definidas na construção do sub-esquema. Apresenta o seguinte algoritmo geral:

```
início  (módulo CONSULTA)
  Lista entidades do sub-esquema
  Escolhe entidade
  Se (fim da consulta) Então retorna menu principal
  Senão
    Cadastra entidade
    Enquanto (não fim da consulta)
           e (existir relacionamentos possíveis) faça
      Lista relacionamentos possíveis com a nova entidade
      Se (existir relacionamentos posíveis) Então
        Escolhe relacionamento
        Se (não fim da consulta) Então
          Cadastra relacionamentos
          Cadastra a outra entidade do relacionamento
        fim se
      fim se
    fim enquanto
    Para cada entidade cadastrada faça
      Escolhe atributos da Lista-Alvo
      Lê predicado da consulta
      Se (existir predicado da consulta) Então
        Cadastra predicado da consulta
      fim se
    fim Para cada
```

Lista Opções de Saída

Escolhe Opções de Saída

Cadastra Opções de Saída

Define nome do catálogo da consulta

fim se

fim algoritmo

Com as tabelas montadas, o sistema pesquisará os outros componentes do sub-esquema (não participantes do caminho), que restringem os acessos aos elementos das entidades da consulta, incorporando-os ao caminho definido pelo usuário. Caso esta incorporação provoque uma consulta com caminhos bifurcados, o sistema dividirá a consulta em várias consultas sem caminhos bifurcados, marcando as entidades do caminho original que são pontos de bifurcação. Este procedimento permite que a consulta seja resolvida através da interseção dos vários resultados das consultas desmembradas.

6.2.2 - O Modificador da Consulta

É o módulo que implementa o mapeamento dos componentes do sub-esquema para os componentes do esquema.

Este mapeamento usa as seguintes correspondências:

- Um relacionamento **1:1** num sub-esquema corresponde a um idêntico relacionamento no esquema.

- Um relacionamento **1:n** independente num sub-esquema corresponde ou a um relacionamento **1:n** independente no esquema ou

ao relacionamento 1:n do mesmo sentido no par de relacionamentos 1:n opostos e dependentes no esquema.

- Dois relacionamentos **1:n opostos e dependentes** num sub-esquema correspondem a dois relacionamentos **1:n opostos e dependentes** no Esquema.

- Cada conjunto de entidades e cada atributo num sub-esquema correspondem, respectivamente, a um conjunto de entidades e a um atributo no esquema.

Feito o mapeamento, o sistema modifica a consulta, incluindo as projeções e/ou seleções da definição do sub-esquema.

6.2.3- O Otimizador da Consulta

É o módulo que implementa algumas tarefas, com a finalidade de tornar a consulta mais eficiente.

O nosso otimizador possui dois sub-módulos: o otimizador do caminho e o otimizador de relacionamentos.

a) O Otimizador do Caminho

Se for possível, este sub-módulo exclui do caminho de uma consulta, a partir dos seus extremos, os conjuntos de entidades que não contribuem para a realização da consulta.

Um conjunto de entidades não contribui para a realização de uma consulta quando ocorrer, simultâneamente, as seguintes condições:

1 - O conjunto de entidades não possuir atributos no predicado da consulta, nem na lista-alvo e nem na expressão de definição da projeção horizontal do sub-esquema;

2 - O conjunto de entidades não ser elo de ligação entre os conjuntos de entidades do caminho;

3 - O conjunto de entidades não ser ponto de bifurcação no caminho.

b) **O Otimizador de Relacionamentos**

Este sub-módulo realiza duas tarefas básicas:

1 - Se for possível e necessário, torna o caminho da consulta mais natural, tentando evitar a formação de arquivos intermediários, através da troca de um relacionamento 1:n, num sentido, pelo relacionamento 1:n de sentido oposto, quando estes formarem um par de relacionamentos opostos e dependentes. Um exemplo deste tipo de otimização se encontra descrito no capítulo 3, deste trabalho.

2 - Avalia as condições especiais de seleção de atributos, surgidas no contexto do caminho da consulta. Esta avaliação é importante, porque este tipo de condição de seleção (vista anteriormente) só deve ser criada em conjuntos de entidades destinos, pois, seu significado é: **Encontrar no conjunto de entidades origem, aquelas entidades que possuam relacionamentos, ao mesmo tempo, com as entidades E1, E2, ... , Ej, ... do conjunto de entidades destino, conforme a definição expressa na condição de seleção especial.** Podemos examinar algumas destas

situações nas seguintes consultas ao sub-esquema, definido no capítulo 2:

1 - **Listar os professores que ensinam as disciplinas Física e Matemática.** Neste caso, é possível se obter um resultado não vazio da consulta, pois um professor (entidade origem do relacionamento ensina) poderá se relacionar com mais de uma disciplina (entidade destino do relacionamento ensina) e portanto poderá ensinar as disciplinas Física e Matemática, num mesmo período.

2 - **Listar as disciplinas ensinadas pelos professores João e Maria.** Neste caso, não é possível se obter um resultado diferente de vazio na consulta, pois uma disciplina (entidade destino do relacionamento ensina) só poderá estar relacionada com um professor (entidade origem do relacionamento ensina) e portanto, não poderá ser ensinada pelos professores João e Maria, num mesmo período.

Para se evitar a geração de programas aplicativos de consultas que, pelo motivo descrito acima, resultarão sempre em resultados vazios, este sub-módulo enviará ao usuário uma mensagem **CONSULTA VAZIA**, quando uma condição de seleção especial aparecer numa das seguintes situações:

1 - Num conjunto de entidades que forme sózinho um caminho da consulta.

2 - Num dos conjuntos de entidades de um conjunto de relacionamentos do tipo 1:1.

3 - Num conjunto de entidades origem de um conjunto de

relacionamentos do tipo 1:n independente.

4 - Num conjunto de entidades origem de um conjunto de relacionamentos do tipo 1:n, do par de relacionamentos opostos e dependentes, cujo o usuário do sub-esquema não tenha acesso ao outro conjunto de relacionamentos do par.

7 - ESTUDO DE CASO USANDO O SBD/TS (OPERACIONALIZAÇÃO DO SISTEMA)

Neste trabalho, a operacionalização do nosso sistema foi feita usando o Sistema de Gerenciamento de Banco de Dados (SGBD) SBD/TS [9], descrito, sucintamente, no Apêndice deste trabalho.

As razões que nos levaram a escolher o SBD/TS como BD operacional foram as seguintes:

1- Ser um SGBD construído para ser utilizado em micro-computadores, compatíveis com a linha IBM/PC/XT/AT.

2- Não apresentar, na época do ínicio do trabalho, uma proposta para a definição e manipulação de dados a nível externo.

3- Apresentar, ao nível conceitual, uma interface com a linguagem a linguagem PASCAL.

7.1 - O Mapeador entre a LINDEC e a LDD/SBD-TS

Apesar da implementação deste mapeador não fazer parte do escopo do nosso trabalho, apresentaremos aqui as correspondências que servirão de base para uma futura implementação.

Portanto, para um conjunto de informações fornecidas na definição de um esquema usando a LINDEC, teremos as seguintes correspondências, que devem ser realizadas pelo mapeador:

1- O nome do esquema na tabela de esquemas corresponderá ao nome do BD na LDD/SBD-TS.

2- A definição dos parâmetros (num_de_unidades, página_lógica e descrição_das_unidades) será solicitada pelo mapeador ao ABD,

pois são informações específicas da LDD/SBD-TS.

3- Cada nome do usuário e senha do esquema, na tabela de usuários, corresponderá a um nome do usuário e uma senha na LDD/SBD-TS.

4- Cada nome de entidade do esquema, na tabela de entidades, corresponderá a um nome de registro na LDD/SBD-TS.

5- Cada nome de atributo de uma entidade, na tabela de atributos, corresponderá ao nome de um item de dados do registro equivalente na LDD/SBD-TS. Cada tipo definido para um atributo, terá um tipo equivalente no item de dados correspondente. Cada tamanho, para um atributo, corresponderá ao tamanho do item de dados equivalente na LDD/SBD-TS.

6- Um relacionamento 1:1 no esquema corresponderá a um relacionamento 1:1 na LDD/SBD-TS. Portanto, o nome, o tipo e as entidades origem e destino deste relacionamento, na tabela de relacionamentos, serão, respectivamente, o nome, o tipo e os registros dono e membro na LDD/SBD-TS.

7- Um relacionamento 1:N independente no esquema,corresponderá a um relacionamento 1:N na LDD/SBD-TS. Portanto, o nome, o tipo, o atributo chave de ordenação da entidade destino e as entidades origem e destino deste relacionamento, na tabela de relacionamentos,serão, respectivamente, o nome, o tipo, o item de dados chave de ordenação do registro membro e os registros dono e membro na LDD/SBD-TS.

8- Dois relacionamentos opostos e dependentes no esquema corresponderão a um relacionamento N:M na LDD/SBD-TS. Aqui, o mapeamento se dará da seguinte maneira:

- O primeiro relacionamento da definição do par de relacionamentos opostos e dependentes no esquema, fornecerá o nome do relacionamento, o item de dados chave de ordenação do registro membro e o registro membro na LDD/SBD-TS.

- O outro relacionamento do par de relacionamentos opostos e dependentes no esquema, fornecerá o item de dados chave de ordenação do registro dono e o registro dono na LDD/SBD-TS.

Este mapeamento é possível devido a existência de uma tabela de apontadores para os relacionamentos do esquema (tab-relopr), criada, após a definição dos relacionamentos na LINDEC, pelo mapeador. Esta tabela indicará qual dos relacionamentos será realmente considerado no esquema operacional. Vejamos, através do esquema definido no capítulo 2, como ficaria a tabela criada pelo mapeador na figura 8.2.

TABELA DE RELACIONAMENTOS DO ESQUEMA

											TAB RELOPR	
0	1	VINCULA1	COLEGIOS	PROF	NP	1:n	1	i	0	0
1	1	VINCULA2	COLEGIOS	PROF	NOME_PRF	1:n	2	i	1	1
2	1	ENSINA	PROF	DISC	ND	1:n	3	i	2	2
3	1	CURSA	ESTUD	DISC	ND	1:n	4	d	3	3
4	1	EHCURSADA	DISC	ESTUD	NE	1:n	5	d	4	3
5	1	POSUI	COLEGIOS	ESTUD	NE	1:n	6	i	5	5
6	1	ALOCA	DISC	FUNCION	NF	1:n	7	i	6	6
7	1	TEM	PROF	QUALIFIC	NQ	1:n	8	d	7	7
8	1	PERTENCE	QUALIFIC	PROF	NP	1:n	-1	d	8	7
	:		:		:		:			:

FIGURA 7.1 - EXEMPLO DA TABELA DE MAPEAMENTO ENTRE OS RELACIONAMENTOS DO ESQUEMA E OPERACIONAL.

9- Para cada registro gerado na LDD/SBD-TS, via as entidades do esquema, será definido um relacionamento na LDD/SBD-TS entre o registro interno SISTEMA e o registro gerado, sendo que, o nome deste relacionamento será o nome do registro gerado, o seu tipo será 1:N, a sua chave de ordenação será uma das chaves de ordenação do registro gerado num outro relacionamento e seu registro membro será o registro gerado.

10- Adota-se, para todos os relacionamentos gerados na LDD/SBD-TS, com exceção dos existentes com o registro interno SISTEMA, o modo de inserção manual (MAN). As razões desta adoção, são: deixar o controle de qualquer operação de manutenção do BD sobre a responsabilidade do programa de manutenção gerado na LIMADEX; e a inexistência,no nosso modelo, da definição de modos de inserção nos conjuntos de relacionamentos.

7.2 - O Mapeador entre LIMADEX e a LMD/SBD-TS

Neste trabalho, implementamos somente o módulo consulta da LIMADEX,estando disponível duas sub-operações: Criar uma Consulta e Operar Arquivos Resultados. Por isto, nesta sub-seção, apresentaremos somente os mapeamentos destas sub-operações da LIMADEX para programas aplicativos na LMD/SBD-TS.

7.2.1 - Mapeamento da sub-operação 'Criar uma Consulta'

Como foi visto, uma consulta na LIMADEX será mapeada para a LMD/SBD-TS, que na versão utilizada, é um programa aplicativo na linguagem PASCAL da MICROSOFT [21] com comandos SBD/TS acoplados. Por esta razão, o algoritmo principal desta sub-operação é

composto de sub-módulos, que basicamente gerarão as declarações e procedimentos que compõem um programa básico PASCAL. Seus sub-módulos são: Gerador de declarações CONST, Gerador de declarações TYPE, Gerador de declarações VAR, Gerador de PROCEDURES e o Gerador do Módulo Principal. A seguir descrevemos este algoritmo:

```
início  (Mapeamento 'criar uma consulta')
  Abre arquivo_das_declarações
  Gera título_das_declarações
  Enquanto existir entidades no caminho da consulta faça
    Gera rótulo_entidade
    Gera tipo_entidade
    Gera var_entidade
  fim enquanto
  Gera rótulo_texto
  Se local_da_saida = arquivo Então
    Gera rótulo_resultado
    Enquanto existir entidades no caminho da consulta faça
      Gera tipo_arquivo_resultado
    fim enquanto
  fim se
  Gera arquivo_texto_da_consulta
  Gera corpo_principal_da_consulta
fim algoritmo
```

1 - **Gerador de Declarações CONST**

É o sub-módulo que gera um arquivo, do tipo texto, cujo nome

no diretório é <nome_do_arquivo_consulta>.ROT, e que contem todas as declarações de constantes (CONST) utilizadas no programa aplicativo. As constantes são:

a) Para cada conjunto de entidades do caminho existe uma constante, cujo o valor é nome do conjunto de entidades no sub-esquema. Esta constante será utilizada pelo programa aplicativo na identificação dos dados das entidades que serão mostrados na impressora ou/e no video. Seu formato será:

rótulo<posição_entidade_caminho> = '<nome_ent subesquema>';

Por exemplo, para um conjunto de entidades do sub-esquema chamado de PROFESSOR, que participe da consulta e esteja cadastrado em tab_rgesq na posição 1, teremos a seguinte constante: **rótulo1 = 'PROFESSOR';**

b) Uma constante que identifica o arquivo,feito pelo usuário da LIMADEX, que contem o texto da consulta. Seu formato será :

rótulo_txt = '<unidade_de_armazenamento> :
<nome_do_arquivo_consulta>.TXT'

c) Uma constante que identifica o arquivo, que contem os dados selecionados pela consulta. Esta constante só existe, caso um dos locais de saída do resultado da consulta escolhido tenha sido ARQUIVO. Seu formato será :

nome_arq_resultado = '<unidade_de_armazenamento> :
<nome_do_arquivo_consulta>.DAT';

Vejamos como ficariam as constantes dos itens "b" e "c" se o usuário tivesse elaborado uma consulta cujo nome dado ao arquivo consulta fosse CONS1, que se desejasse armazená-lo na unidade "c" e que o resultado da consulta ficasse num arquivo:

```
ROTULO.TXT = 'C : CONS1.TXT';
NOME_ARQ_RESULTADO = 'C : CONS1.DAT';
```

2 - Gerador de Declarações TYPE

É o sub-módulo que gera um arquivo, do tipo texto, cujo nome no diretório é <nome_do_arquivo_consulta>.IIP, e que contem todas as declarações de tipos (TYPE) de dados utilizados no programa aplicativo. Os tipos gerados são:

a) Para cada conjunto de entidades do caminho existe um tipo de dados RECORD,cujo o nome é dado pela seguinte composição: <variável_registro_da_entidade>tipo, sendo que, cada atributo do conjunto de entidades forma um dos seus campos. Os tipos primitivos definidos para cada atributo são convertidos em tipos primitivos do PASCAL, qualificando, assim, cada campo. Este tipo RECORD identifica, dentro do programa aplicativo, quais os conjuntos de entidades que participam do caminho da consulta, permitindo assim, a criação de variáveis que possam receber ou fornecer dados aos registros existentes no BD fisico. Este tipo RECORD terá o seguinte formato:

```
<variável_registro_da_entidade>tipo = record
        <nome_do_atributo1> [posição_do_atributo1] :
                            <tipo_convertido_do_atributo1>;
              :                          :
              :                          :
        <nome_do_atributoN> [posição_do_atributoN] :
                            <tipo_convertido_do_atributoN>;
    end;
```

Para exemplificar, suponhamos que o conjunto de entidades

PROF (NP,NOME_PRF,SALÁRIO,ENDEREÇO,N_DEPEND) definido no capítulo 2, participe do caminho de uma consulta, estando cadastrada na posição 2 de tab-rgesq. Então, teríamos a geração do seguinte tipo RECORD no programa aplicativo:

```
VR2tipo = record
        NP[00]: byte;
        NOME_PRF[01]: lstring[20];
        SALÁRIO[22]: real;
        ENDERECO[26]: lstring[40];
        N_DEPEND[67]: byte;
end;
```

b) Se um dos locais de saída do resultado for ARQUIVO, existe um tipo RECORD, cujo o nome é **reg_resultado**, sendo que, cada atributo da lista_alvo, forma um dos campos do tipo RECORD e os tipos de cada atributo são convertidos em tipos primitivos do PASCAL, qualificando, assim, cada campo. Este tipo RECORD identifica, dentro do programa aplicativo, qual o formato do arquivo resultado da consulta, permitindo assim, a criação de variáveis que possam receber ou fornecer dados a ele. Este tipo RECORD terá o seguinte formato:

```
reg_resultado = record
   (nome_do_atributo1_na_lista_alvo) :
                                  (tipo_convertido_do_atributo1);
                    :                          :
                    :                          :
   (nome_do_atributoN_na_lista_alvo) :
                                  (tipo_convertido_do_atributoN);
end;
```

3 - Gerador de Declarações VAR

É o sub-módulo que gera um arquivo do tipo texto, cujo nome no diretório é **(nome_do_arquivo_consulta).var**, e que contem todas as declarações de variáveis utilizadas no programa aplicativo. As variáveis geradas são:

a) Uma variável tipo **booleana**, chamada **fez_consulta**, cuja função é indicar se o PDC foi alguma vez satisfeito.

b) Uma variável tipo **char**, chamada **tecla**, cuja função é receber as respostas (s ou n) do usuário,as perguntas feitas pela aplicação.

c) Se um dos locais de saída do resultado da consulta for ARQUIVO, teremos:

- Uma variável do tipo **file of reg_resultado**, cuja função é permitir o acesso ao arquivo que contem o resultado da consulta. Seu formato será:

arq_resultado : file of reg_resultado ;

- Uma variável do tipo **reg_resultado**, cuja função é receber ou fornecer dados ao arquivo, que contem o resultado da consulta. Seu formato será:

dado_resultado : reg_resultado ;

d) Para cada conjunto de entidades do caminho de uma consulta teremos:

- Uma variável tipo **booleana**, cuja função será detectar o encerramento da pesquisa nos elementos do conjunto de entidades. Seu formato será:

tem(variável_registro_da_entidade) : boolean;

- Uma variável tipo **byte**, cuja função é receber o código de retorno de uma operação SBD/TS. Seu valor é importante, pois, através dele se decide as ações a serem tomadas.Seu formato será:

cret(variável_registro_da_entidade) : boolean;

- Uma variável do tipo RECORD definida para cada conjunto de entidades, cuja função é receber ou fornecer dados aos conjuntos de entidades armazenados fisicamente no BD. Seu formato será:

(variável_registro_da_entidade)(nome_da_entidade) :

(variável_registro_da_entidade)tipo;

Para exemplificar, suponhamos que o caminho da consulta possua dois conjuntos de entidades PROF e DISC cadastrados nas posições 0 e 1 de tab_rgesq e que o local de saída escolhido para o resultado da consulta seja ARQUIVO. Então, teremos a seguinte geração de variáveis:

```
Var
   fez_consulta      : boolean;
   tecla             : char;
   VR0PROF      : VR0tipo;
   cretVR0      : byte;
   temVR0       : boolean;
   VR1DISC      : VR1tipo;
   cretVR1      : byte;
   temVR1       : boolean;
   arq_resultado     : file of reg_resultado;
   dado_resultado    : reg_resultado;
```

4 - Gerador de PROCEDURES

O mapeador, ao gerar o programa aplicativo, transforma cada tarefa necessária a execução da consulta em procedimentos. Basicamente, podemos dividir estes procedimentos em três tipos: procedimentos padrões, procedimentos externos e procedimentos de tarefas em entidades.

a) Procedimentos Padrões

São os procedimentos gerados internamente pelo mapeador, cujas tarefas são apoiar a realização da consulta, tais como, a de apresentação da finalidade da consulta, a de realização dos objetivos finais da consulta, a de MERGE-SORT em arquivos e a de operações com arquivos. A seguir, descrevemos de forma sumária e informal estes procedimentos:

- Procedure Apresentação

Este procedimento apresenta ao usuário, que utiliza o programa aplicativo gerado, o texto explicativo da consulta(feito durante sua construção) e o questiona sobre a necessidade ou não de execução do programa. Apresenta uma forma única, independente da consulta, e portanto seu algoritmo só prevê a transcrição de suas declarações para o arquivo texto, que o contem **(<nome_do_arquivo_consulta>.con)**. Seu conteúdo será:

```
procedure apresentacao(var tecla : char);
var arquivo_texto : text;
    texto         : lstring(80);
begin
 assign(arquivo_texto,rotulo_txt);
 reset(arquivo_texto);
 GOTOXY(1,3);
 while (not eof(arquivo_texto)) do
  begin
   readln(arquivo_texto,texto);
   writeln(OUTPUT,texto);
  end;
 close(arquivo_texto);
 GOTOXY(3,21);
 write(OUTPUT,'Quer realizar esta consulta(s/n)?');
 tecla := ' ';
 while (not(tecla in ['s','S','n','N'])) do
  begin
   GOTOXY(37,21);
   write(OUTPUT,' ');
   GOTOXY(37,21);
   read(INPUT,tecla);
  end;
end;
```

- Procedure Ação Final

Este procedimento produz a saída da consulta de acordo com a

definição da lista_alvo e dos locais de saída dos resultados, elaborados durante a construção da consulta. Este procedimento, também, é gerado no arquivo texto <nome_do_arquivo_consulta>.con. Apresenta o seguinte algoritmo de geração:

```
início {Gerar Ação Final}
  Gera cabeçalho_do_procedimento
  Gera comandos_iniciais_de_edição
  Enquanto existir entidades no caminho da consulta faça
    Enquanto existir atributos nas entidades faça
      Se o atributo está na Lista-Alvo Então
        Se local_saída = video Então
          Gera saída_video
        fim se
        Se local_saída = impressora Então
          Gera saída_impressora
        fim se
        Se local_saída = arquivo Então
          Gera saída_arquivo
        fim se
      fim se
    fim enquanto
  fim enquanto
  Gera comandos_finais_de_edição
fim algoritmo
```

Vejamos como ficaria este procedimento no caso da consulta descrita no capítulo 3:

```
procedure Ação Final
begin
  if (not(fez_consulta)) then fez_consulta:= true;
  LIMPATELA;
  GOTOXY(3,2);
  writeln(OUTPUT, rotulo0);
  writeln(OUTPUT,VR0PROF.NUM_PROF);
  dado_resultado.NUM_PROF := VR0PROF.NUM_PROF;
  writeln(OUTPUT, VR0PROF.NOME_PRF);
  dado_resultado.NOME_PRF := VR0PROF.NOME_PRF;
  write(arq_resultado, dado_resultado);
  write(OUTPUT, 'aperte uma tecla e (ENTER)');
  read(tecla);
end;
```

- **procedure MERGE_SORT**

Este procedimento tem a função de ordenar arquivos que não pertecem a estrutura física do DB,mas são manipulados no programa aplicativo gerado. Estes arquivos podem ser **arquivos temporários**, no caso da **sub-operação Consulta** ou **arquivos resultados de consulta** no caso da **sub-operação Operar Arquivos-Resultados**. A descrição deste procedimento é por demais conhecida e se encontra em detalhes em [19,20].

- **Procedimentos de Operações com Arquivos**

Estes procedimentos, também manipulam arquivos que não pertencem a estrutura física do BD, mas que podem participar do

programa aplicativo gerado. Existem quatro destes procedimentos: União, Diferença, Interseção e Cópia-Arq. Os três primeiros, recebem dois arquivos do mesmo tipo e geram um terceiro, contendo os registros resultantes das operações de união, diferença e interseção, respectivamente. O último, cópia um arquivo em outro.

b) **Procedimentos Externos**

São aqueles procedimentos cujos conteúdos não são gerados internamente pelo mapeador. Eles foram elaborados externamente na linguagem ASSEMBLY com a finalidade de fornecer alguns comandos necessários ao programa aplicativo, ausentes na linguagem PASCAL da Microsoft [21]. Estes procedimentos são declarados como procedure EXTERNAL, cabendo esta tarefa ao mapeador. Cabe, também, ao mapeador gerar as chamadas necessárias a estes procedimentos, dentro do programa aplicativo.

Na versão atual do sistema, dois procedimentos externos foram implementados: **LIMPATELA** e **GOTOXY(coluna,linha)**.

c) **Procedimentos de Tarefas em Entidades**

É o sub-módulo do mapeador que gera um procedimento, para cada conjunto de entidades do caminho, cuja função é realizar as tarefas (encontrar itens de dados, recuperar seus valores e testá-los com as condições estabelecidas pelo PDC e/ou pela definição do sub-esquema) necessárias para se chegar ao resultado da consulta. Cada procedimento é gerado em um arquivo tipo texto, cujo nome é dado pela seguinte composição:

(nome_do_arquivo_consulta).E(posição_da_entidade_em_TAB_RGESQ)

Por exemplo, uma consulta chamada **consult1**, que possui um caminho com dois conjuntos de entidades cadastrados nas posições 0 e 1 de tab_rgesq, fará com que o mapeador produza dois arquivos textos: **consult1.E0; consult1.E1.**

Estes procedimentos, gerados pelo mapeador, a partir dos conjuntos de entidades do caminho,são as peças chaves na resolução da consulta,pois,um programa aplicativo nada mais é do que um encadeamento de chamadas entre estes procedimentos. A figura 8.3, deixa bem claro como este encadeamento é realizado.

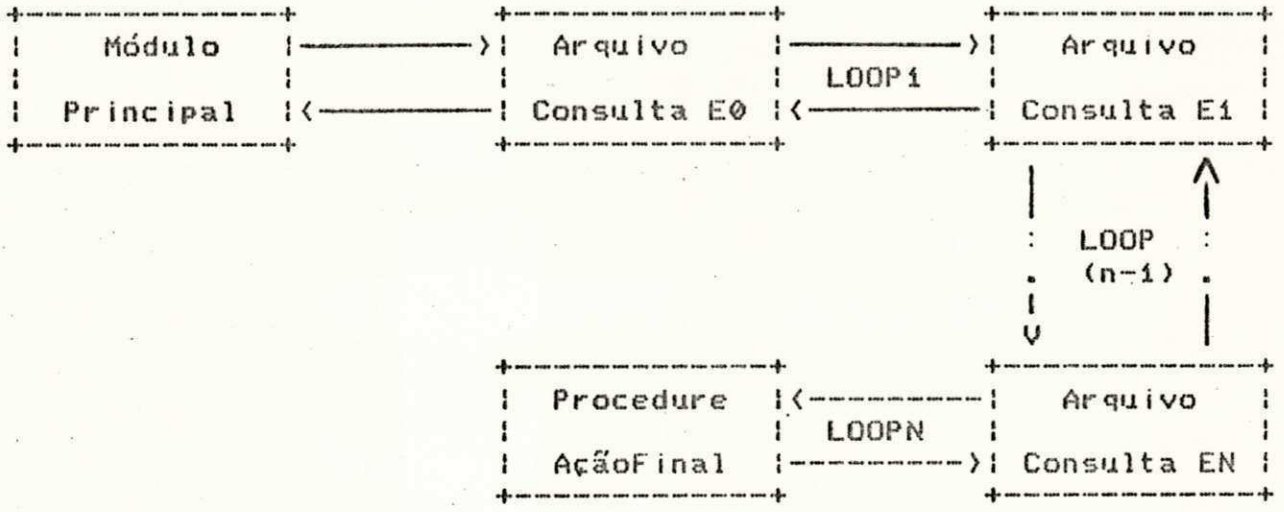


FIGURA 7.2 - ESQUEMA DO FUNCIONAMENTO BÁSICO DO PROGRAMA APLICATIVO

Os corpos destes procedimentos dependem,basicamente, de duas situações possíveis nos conjuntos de entidades:

a) O conjunto de entidades possuir ou uma condição de consulta normal ou uma condição de consulta especial.

b) O conjunto de entidades ser ou não um ponto de bifurcação surgido no caminho da consulta.

Para cada combinação destas duas situações, as tarefas a serem realizadas terão um rumo diferente, e portanto o mapeador gerará corpos diferentes para os procedimentos entidades.

Vejamos, a seguir, os algoritmos de geração destes procedimentos:

```
início (Gera procedimentos entidades)
 Enquanto existir entidades no caminho da consulta faça
   Se a entidade não for ponto de bifurcação Então
     Gera Procedimento_sem_Bifurcação
   Senão Gera Procedimento_com_Bifurcação
   fim se
 fim enquanto
fim algoritmo

início (Gera procedimentos para os conjuntos de entidades que não
        são pontos de bifurcação do caminho da consulta)
 Se a entidade possui condição de consulta normal Então
   Gera Procedimento_Normal
 Senão Gera Procedimento_Especial
 fim se
fim algoritmo
```

```
início  {Gera procedimentos  para os conjuntos de  entidades  com
         condição de consulta normal e sem bifurcação}

 Gera título_do_procediemento

 Se primeira entidade do caminho Então

   Gera comando_SBD_ACHAR_PRIMEIRO

   Gera teste_da_variável_CRETVR

 fim se

 Gera atribuição_para_variável_TEMVR

 Se não for a segunda entidade do caminho Então

  Gera comando_SBD_FAZER

 fim se

 Gera comando_REPEAT

 Gera comando_SBD_RECUPERAR

 Gera condição_SUBESQUEMA

 Gera condição_consulta_normal

 Gera próxima_chamada_de_procedimento

 Se primeira entidade do caminho Então

   Gera comando_SBD_ACHAR_PRÓXIMO

   Gera comando_SBD_FAZER

 Senão Gera comando_SBD_ACHAR_PRÓXIMO

 fim se

 Gera teste_variável_CRETVR

 Gera atribuição_para_TEMVR

 Gera comando_UNTIL

 Gera fim_do_procediemnto

fim algoritmo
```

```
início  (Gera procedimentos para os conjunto de entidades com
         condição consulta especial e sem bifurcação)
 Se primeira entidade do caminho da consulta Então
  Faz a segunda entidade do caminho ser entidade corrente
 Senão
  Se segunda entidade do caminho da consulta Então
   Faz a primeira entidade do caminho ser entidade corrente
  fim SE's
 Gera título_do_procediemnto
 Gera função_condição_especial
 Gera teste_da_função_condição
 Se primeira entidade do caminho Então
  Gera próxima_chamada_de_procedimento
 Senão
    Gera atribuição_para_TEMVR
    Gera comando_RESET
    Gera comando_REPEAT
    Gera comando_READ
    Gera comando_SBD_ACHAR_PRIMEIRO
    Gera comando_SBD_ACHAR_BASEADO_EM_ITEM
    Gera teste_variável_CRETVR
    Gera próxima_chamada_de_procedimento
    Gera teste_fim_de_arquivo
    Gera comando_UNTIL
fim se
 Gera comando_CLOSE
 Gera fim_de_procedimento
fim algoritmo
```

```
início  (Gera procedimentos para os conjuntos de entidades pontos
         de bifurcação no caminho da consulta)

 Se a entidade não possui condição especial Então

   Gera Procedimento_Normal_Bifurcado

 Senão Gera Procedimento_Especial_Bifurcado

fim algoritmo


início   (Gera procedimentos para os conjuntos de  entidades  com
          condição de consulta normal e ponto de bifurcação)

 Gera titulo_do_procedimento

 Gera declarções_VAR

 Gera comandos_iniciais

 Gera comandos_finais

fim algoritmo


início  (Gera  procedimentos para os conjuntos de entidades  com
         condição especial e ponto de bifurcação)

 Se primeira entidade do caminho da consulta Então

  Faz a segunda entidade do caminho entidade corrente

 Senão Se segunda entidade do caminho Então

       Faz primeira entidade do caminho entidade corrente

  fim SE's

 Gera título_do_procedimento

 Gera declarações_VAR

 Gera função_condição_especial

 Gera procedimento_processador_de_bifurcação

 Gera comandos_iniciais

 Gera teste_da_função_condição

 Gera comandos_finais

fim algoritmo
```

```
início (Gera declarações VAR para o procedimento de um
        conjunto de entidades bifurcado )

 Gera título_VAR

 Gera variável_EXISTE

 Gera variável_VRNAUX

fim algoritmo


início  (Gera comandos iniciais)

 Gera comandos_ASSIGN_REWRITE

 Gera chamada_procedimento_processador_da_consulta

 Gera comando_RESET

 Gera atribuição_para_1EMVR

 Gera atribuição_para_EXISTE

fim algoritmo


início (Gera os comandos finais de um procedimento de um conjunto
        de entidades ponto de bifurcação)

 Gera comando_REPEAT

 Gera teste_variável_EXISTE

 Gera corpo_do_teste_da_variável_EXISTE

 Gera comando_SBD_RECUPERAR

 Gera teste_dos_dados_coletados_processo_bifurcado_e_os_dados_
      existentes_no_conjunto_de_entidades

 Gera corpo_de_teste_da_comparação_entre_dados

 Gera teste_de_fim_arquivo

 Gera atribuição_para_1EMVR

 Gera comando_UNTIL

 Gera fim_do_procedimento

fim algoritmo
```

Vejamos como ficariam os procedimentos entidades para a consulta descrita no capítulo 3.

1) Arquivo texto CONSULT1.E0

```
procedure proccolégios
begin
 apm('colégios',cretVR0)
 If (cretVR0 <> 0) then
   writeln(OUTPUT,'Não existe ',rotulo0,' no BD')
 else
  begin
   temVR0 := true;
   fdcmc('vincula1','colégios',cretVR0)
   repeat
    rddc('vincula1',ads VR0colégios,cretVR0)
    If (cretVR0 = 0) then
    If (VR0colégios.NOME_COL = 'X') then
       procprof1;
    apxm('colégios',cretVR0)
    fdcmc('vincula1','colégios',cretVR0);
    If (cretVR0 <> 0) then temVR0 := false;
   until (not(temVR0);
  end;
end;
```

2) Arquivo texto CONSULT1.E1

```
procedure procprof1;
begin
 temVR1 := true;
 repeat
  rdmc('vincula1',ads VR1prof,cretVR1)
  If (cretVR1 = 0) then
  If (VR1prof.NP > 68) then
     procdisc2;
  apxm('vincula1',cretVR1)
  If (cretVR1 <> 0) and (cretVR1 <> 8) then temVR1 := false;
 until (not(temVR0);
end;
```

3) Arquivo texto CONSULT1.E2

```
procedure procdisc2;
begin
 temVR2 := true;
 fdcmc('ensina','vincula1',cretVR2);
 repeat
  rdmc('ensina',ads VR2disc,cretVR2)
  If (cretVR2 = 0) then
  If (VR2disc.duração > 2) then
     procestud3;
  apxm('ensina',cretVR2)
  If (cretVR2 <> 0) and (cretVR2 <> 8) then temVR2 := false;
 until (not(temVR2);
end;
```

4) Arquivo texto CONSULT1.E3

```
procedure procestud3;
begin
 temVR3 := true;
 fdcmc('ehcursada','ensina',cretVR3);
 repeat
  rdmc('ehcursada',ads VR3estud,cretVR3)
  If (cretVR3 = 0) then
  If (VR3estud.NE > 121) and
     (VR3estud.NE <= 212)  then
     AçãoFinal;
  apxm('ehcursada',cretVR3)
  If (cretVR3 <> 0) and (cretVR3 <> 8) then temVR3 := false;
 until (not(temVR3);
end;
```

5 - Gerador do Módulo Principal

O módulo principal do programa aplicativo gerado, realiza as seguintes tarefas principais:

a) Chama a procedure apresentação.

b) Testa a resposta dada pelo usuário, na procedure apresentação, sobre a realização ou não da consulta.

c) Abre e fecha o arquivo resultado, se necessário.

d) Abre e fecha o BD operacional solicitado.

Para gerar este módulo, teremos o seguinte algoritimo:

```
início  (Gera módulo principal do programa aplicativo)
  Gera título_do_programa_aplicativo
  Gera comandos_INCLUDES
  Gera variável_FEZ_CONSULTA
  Gera chamada_procedimento_LIMPA_TELA
  Gera chamada_procedimento_APRESENTAÇÃO
  Gera teste_com_variável_TECLA
  Se local_resultado = ARQUIVO Então
     Gera comando_ASSIGN
     Gera comando_REWRITE
  fim se
  Gera comando_SBD_ABREBD
  Gera chamada_para_procedimento_entidade_que_acessa_o_BD
  Se local_resultado = ARQUIVO Então
     Gera comando_CLOSE
  fim se
  Gera teste_com_variável_FEZ_CONSULTA
  Gera corpo_do_teste_FEZ_CONSULTA
  Gera fim_do_módulo_principal
fim algoritmo
```

Este módulo principal é gerado, num arquivo tipo texto, definido pelo usuário (**<nome_do_arquivo_consulta>.con**). Este arquivo, também, contem o comando PASCAL INCLUDE, cuja função é ligar todos os arquivos textos gerados pelo mapeador a este módulo, bem como, o arquivo texto fornecido pelo SBD/TS, que

contem as definições, em forma de procedimentos externos, de todos os comandos SBD/TS.

Vejamos como fica o módulo principal gerado para a consulta descrita no capítulo 3.

```
program consulta1(input,output);
(*LINESIZE  :  132  *)
(*$INCLUDE:  'c:LMDSBD.PAS'      *)
(*$INCLUDE:  'c:consulta1.rot'   *)
(*$INCLUDE:  'c:consulta1.tip'   *)
(*$INCLUDE:  'c:consulta1.var'   *)

procedure GOTOXY(nc,nl  :  byte); external
procedure LIMPATELA; external

procedure apresentação(var tecla  :  char);
begin
            :
            :
end;

procedure AcaoFinal;
begin
            :
            :
end;

(*$INCLUDE:  'b:consulta1.e0'     *)
(*$INCLUDE:  'b:consulta1.e1'     *)
(*$INCLUDE:  'b:consulta1.e2'     *)
(*$INCLUDE:  'b:consulta1.e3'     *)

begin
 fez_consulta := false;
 LIMPATELA;
 apresentação;
 If (tecla = 's') or (tecla = 'S') then
  begin
   assign(arq_resultado,nome_arq_resultado);
   rewrite(arq_resultado);
   abrebd('colégios',0,2,'luciano','senha1',1);
   proccolégios0;
   close(arq_resultado);
   fechabd('colégios');
   If (not(fez_consulta)) then
      writeln(OUTPUT,'Consulta Vazia');
  end;
end;
```

7.2.2 - Mapeamento da sub-operação "Operar Arquivos Resultados"

Como foi visto anteriormento, esta sub-operação possui a função de criar novos arquivos que serão resultados de operações de INTERSEÇÃO, UNIÃO e DIFERENÇA entre arquivos resultados. Portanto, o programa aplicativo gerado para esta operação, basicamente, se constitui de chamadas aos procedimentos de operar arquivos, gerados pelo mapeador, já visto na sub-seção 7.2.1.. Portanto, apresentaremos aqui, somente, o algoritmo de geração do programa aplicativo, desta operação e o algoritmo de geração do seu módulo principal.

```
início  (Gera a sub-operação Operar Arquivos Resultados)
  Abre arquivos_textos
  Gera título_do_procedimento
  (No arquivo texto (nome_do_arquivo_consulta).con)
  Gera declarações_CONST
  (No arquivo texto (nome_do_arquivo_consulta).rot)
  Gera declarações_TYPE
  (No arquivo texto (nome_do_arquivo_consulta).tip)
  Gera declarações_VAR
  (No arquivo texto (nome_do_arquivo_consulta).var)
  Gera texto_da_suboperação
  (No arquivo texto (nome_do_arquivo_consulta).txt)
  Gera corpo_principal
  (No arquivo texto (nome_do_arquivo_consulta).con)
fim algoritmo
```

```
início  (Gera corpo principal do programa aplicativo)
  Gera comandos_INCLUDES
  Gera chamada_do_procediamento_APRESENTAÇÃO
  Gera teste_com_variável_TECLA
  Enquanto existir arquivos_resultados na operação faça
    Gera comando_BEGIN
    Gera chamada_para_procedimento_MERGE_SORT
    Encontra próximo_arquivo_resultado
  fim enquanto
  Gera operações_com_arquivos_resultados
  Para cada operação_UNIÃO_DIFERENÇA_INTERSEÇÃO existente faça
    Gera comando_ASSIGN
    Gera chamada_procedimento_operação
    Encontra outra_operação
  fim para cada
  Gera saída_da_operação
  Gera fim_do_procedimento
fim algoritmo
```

Para finalizar este capítulo, apresentamos um resumo dos arquivos gerados pelo mapeador ou pela execução do programa aplicativo e alguns dados fisicos das implementações realizadas:

1 - **<nome_do_arquivo_consulta>.con** -> é o arquivo que contem as procedures APRESENTAÇÃO e ACAOFINAL, os INCLUDES dos outros arquivos necessários a execução do programa aplicativo e o módulo principal deste programa.

2 - **<nome_do_arquivo_consulta>.rot** -> é o arquivo que contem

as declarações CONST, necessárias à realização do programa aplicativo.

3 - **<nome_do_arquivo_consulta>.tip** -> é o arquivo que contem as declarações TYPE, necessárias à realização do programa aplicativo.

4 - **<nome_do_arquivo_consulta>.var** -> é o arquivo que contem as declarações VAR, necessárias à realização do programa aplicativo.

5 - **<nome-do-arquivo-consulta>.E<posição-caminho>** -> é o arquivo que contem o procedimento que realiza as tarefas necessárias, num conjunto de entidades, a resolução de uma consulta.

6 - **<nome-do-arquivo-colsulta>.dat** -> é o arquivo que contem os dados gerados numa consulta, caso um dos locais de saída do resultado seja ARQUIVO.

7 - **<nome-do-arquivo-consulta>.cat** -> é o arquivo que contem a descrição de arquivo-resultado gerado por uma consulta.

8 - Dados Fisicos:

LINDEX

Memória Total do Fonte	: 147.977 Bytes
Memória Total do Objeto	: 83.534 Bytes
Memória Total dos Arquivos do Catálogo	: 1.319 Bytes
Tempo de Compilação (TURBO-PASCAL 3.0)	: 70 Segundos
Tempo Médio de Resposta	: 3 Segundos

LIMADEX

Memória Total do Fonte	: 241.147 Bytes
Memória Total do Objeto	: 146.171 Bytes
Memória Total dos Arquivos do Catálogo	: 53.963 Bytes
Tempo de Compilação (TURBO-PASCAL 3.0)	: 140 Segundos
Tempo Médio de Resposta	: 3 Segundos

8 - Conclusão e Futuros Trabalhos

A realização deste trabalho traz no seu conjunto algumas contribuições que nos parecem importantes :

a) Apresenta uma proposta para modelagem de dados de pequena complexidade, mas abrangente e flexível, principalmente, por permitir a construção de sub-esquemas , baseada em seleções e projeções de conjuntos de entidades e conjunto de relacionamentos.

b) Apresenta um conjunto de linguagens, de fácil uso por usuários não programadores, com a finalidade de definir e manter os dois níveis no modelo de dados.

c) Implementa a Linguagem de Definição de Dados de um Esquema Externo (LINDEX) e a sub-operação consulta da Linguagem de Manipulação de Dados de um Esquema Externo (LIMADEX).

d) Implementa um otimizador de consultas no nível conceitual que, apesar de realizar poucas tarefas, possibilita um aumento na eficiência do processador de consultas.

e) Permite aos usuários do SBD/TS ou de outros GBD's, um ganho em produtividade, pois o tempo gasto pelo processador de consultas, em todas as suas etapas, será bem menor que o tempo gasto por um programador para escrever e testar um programa equivalente. Além disto,os usuários do SBD/TS ou de outros GBD's, que não implementam o conceito de nível externo, poderão utilizar um GBD mais flexível com a incorporação deste conceito.

A concretização deste trabalho nos levou a superar alguns obstáculos, cujas soluções poderão ser utilizadas em outros

trabalhos semelhantes:

a) A escolha de uma proposta de modelo de dados que fosse, ao mesmo tempo, uma ferramenta eficaz na modelagem de dados e tivesse uma implementação que permitisse sua utilização de uma maneira eficiente. Foi por esta razão que optamos por uma proposta de modelo de dados que, se comparada a outras propostas existentes atualmente, não apresenta um grau de semântica elevado. Aliás, achamos que um dos pontos de gargalo no desenvolvimento da Ciência da Computação atual, é a contradição entre ferramentas poderosas de fácil uso e a estrutura sequencial (Von Newman) dos atuais computadores. Entretanto, avaliamos que o modelo proposto atendeu às nossas ansiedades e objetivos.

b) A escolha de uma estrutura de dados que guardasse as informações fornecidas pelas linguagens utilizadas pelo modelo de dados e que permitisse a utilização eficiente destas informações.

c) A escolha de formas que tornassem as linguagens utilizadas pelo modelo de fácil uso por usuários não programadores. Optamos, neste caso, por MENUS e TELAS PROGRAMADAS que permitissem aos usuários escrever o menos possível no fornecimento das informações necessárias exigidas pelo sistema.

d) A escolha da forma de armazenamento interno das expressões definidas nas linguagens utilizadas pelo sistema. Optamos em transformá-las, primeiramente, para forma pós-fixada e então deixá-las em sequências AND/OR.

Sugerimos, ainda, alguns trabalhos que são possíveis de serem realizados, tomando como referência este trabalho:

a) Implementar, a partir das definições existentes neste trabalho, as operações da LIMADEX não implementadas ainda.

b) Implementar a linguagem LINDEC a partir das definições e dos algoritmos existentes neste trabalho.

c) Estudar a questão da adaptabilidade do nosso suporte a outros GBD's, cujos modelos de dados sejam bastante diferentes do MER, o que não é o caso do SBD/TS. Como sugestão, podemos citar a grande família dos GBD's Relacionais. Para ela, não será tão difícil projetar-se o tradutor respectivo: afinal, já existe na literatura a adaptação do MER para o Modelo Relacional [3].

d) Estudar a viabilidade de usar técnicas de Inteligência Artificial na construção de um Processador de Consultas mais poderoso.

e) Estudar alternativas que não prejudiquem a eficiência do sistema, no intuito de superar as limitações semânticas do modelo, tais como, a impossibilidade de definição de atributos para os conjuntos de relacionamentos e a inexistência dos conceitos: Agregação e Generalização [3].

9 - BIBLIOGRAFIA

[1] - C.J.Date, Introdução a Sistemas de Bancos Dados,Ed. Campus, 4a. Edição, 1986.

[2] - J. Ullman, Principles of Database Systems, Computer Science Press, 2nd. Edicion, 1982.

[3] - Valdemar W. Setzer, Banco de Dados, Ed. Edgard Bhucher, 1a. Edição, 1986.

[4] - D. C. Tsichritizis e A. Klugs (Eds), The ANSI/X3/SPARC DBMS Framework:Report of the Study Group on Data Base Management Systems, Information Systems, 3,3,1978.

[5] - M. M. Astrahan et al., System R, A Relational Database Management System, IEEE Computer Society: Computer, 12,5,1979.

[6] - R. W. Taylor e T. L. Frank, CODASYL Database Management Systems, Tutorial: Centralized and Distributed Data Base Systems, IEEE Catalog n. EHO-154-5, 1979, pags. 23-59.

[7] - D. C. Tsichritizis e F. M. Lochowsky, Hierarchical Data Base Management: A survey, ACM Computer Survey, 8,1,1976.

[8] - P. P. Chen, The Entity-Relationship Model - Toward a Unified View of Data, ACM Transation on Database Systems, 1,1,1976, pags. 9-36.

[9] - SBD/TS - Manual do Usuário, TECNOSOFT - Tecnologia de Software, Rio de Janeiro, 1985.

[10] - Daniel A. Menascé, ARCO-IRIS: Um Novo Conceito para Acesso a Banco de Dados, Anais XIV SEMISH, 1987, pags. 153-163.

[11] - Relational Software, Inc. ORACLE Introduction - Version 1.3, 1979.

[12] - M. R. Stonebraker et al., The Design and Implementation of INGRES, ACM Transation on Data Base Systems, 1,3,1976, pags. 189-222.

[13] - ADABAS - Antônio L. Furtado & Celso S. dos Santos, Organização de Banco de Dados, Ed. Campos, 1979.

[14] - C. W. Bachman, Data Struture Diagrams, Data Base,1,2,1969, pags. 4 - 10.

[15] - U. Bussolati et al., View Conceptual Design, em: Methodology and Tools for Data Base Design, E.Ceri (Ed), North-Holland,1985,pags.28-43.

[16] - D. Kapp e J. F. Leben, IMS Programming Techiniques: A Guide to Using DI/I,1979.

[17] - C. L. G. Medeiros e U. Schiel, Projeto do Esquema Conceitual para Sistemas de Informação. Uma abordagem usando a metodologia THM, Anais XII SEMISH, 1985.

[18] - C. J. Date, An Introduction to Database Systems, Addison-Wesley Publishing Company, vol II, 1983.

[19] - E. Horowitz e S.Shani, Fundamentals and Data Structures, Computer Science Press, 1982.

[20] - N. Wirth, Algorithms + Data Structure = Programs, Englewood Cliffs, N.J.,Prentice-Hall, 1976.

[21] - Microsoft Pascal Compiler for the MS-DOS Operating System, 1984.

[22] - A. V. Aho, J. F. Ullman, Principels of Compiler Design, Addison-Wesley Company, 1979.

[23] - DMSII - User Language Interface - Programing Guide,UNISYS, 1985.

A P Ê N D I C E

1 - O SGBD SBD/TS

O SGBD SBD/TS [9] foi desenvolvido pela empresa TECNOSOFT, sendo projetado para ser utilizado em micro-computadores compatíveis com a linha IBM-PC/XT/AT. Adota como modelo de dados uma generalização do modelo CODASYL [6], apresentando os seguintes componentes:

- **Item de Dados** - menor unidade de informação manipulada pelo sistema. Apresenta um nome, com até 8 caracteres, podendo ser de vários tipos (inteiro, real, string, etc). Um valor atribuído a um item de dados é chamado de uma **Ocorrência do Item de Dados.**

- **Registro** - conjunto de itens de dados. Apresenta um nome, com até 8 caracteres. Um conjunto de ocorrências de itens de dados, que compõem um registro, é chamado de uma **Ocorrência de um Registro.** O seu símbolo adotado é um retângulo nomeado.

- **Relacionamento** - associação entre dois registros, sendo que, um é chamado de dono e o outro de membro do relacionamento.Apresenta um nome, com até 8 caracteres. O conjunto formado por todas as as ocorrências do tipo de registro que é dono e por todas as ocorrências do tipo de registro que é membro é chamado de **Ocorrência de um Relacionamento.** O seu símbolo adotado é uma seta no sentido dono/membro. Apresenta os seguintes tipos:

a)**Relacionamento 1:1** - indica que um registro dono pode se

relacionar com um registro membro.

b)**Relacionamento 1:N** - indica que um registro dono pode se relacionar com vários registros membros.

c)**Relacionamento N:1** - indica que vários registros donos podem se relacionar com um registro membro.

d)**Relacionamento N:M** - indica que vários registros donos podem se relacionar com vários registros membros.

- **Indicadores de Corrência** - apontadores de registros, que indicam um determinado estado de corrência do BD. Os indicadores de corrência são:

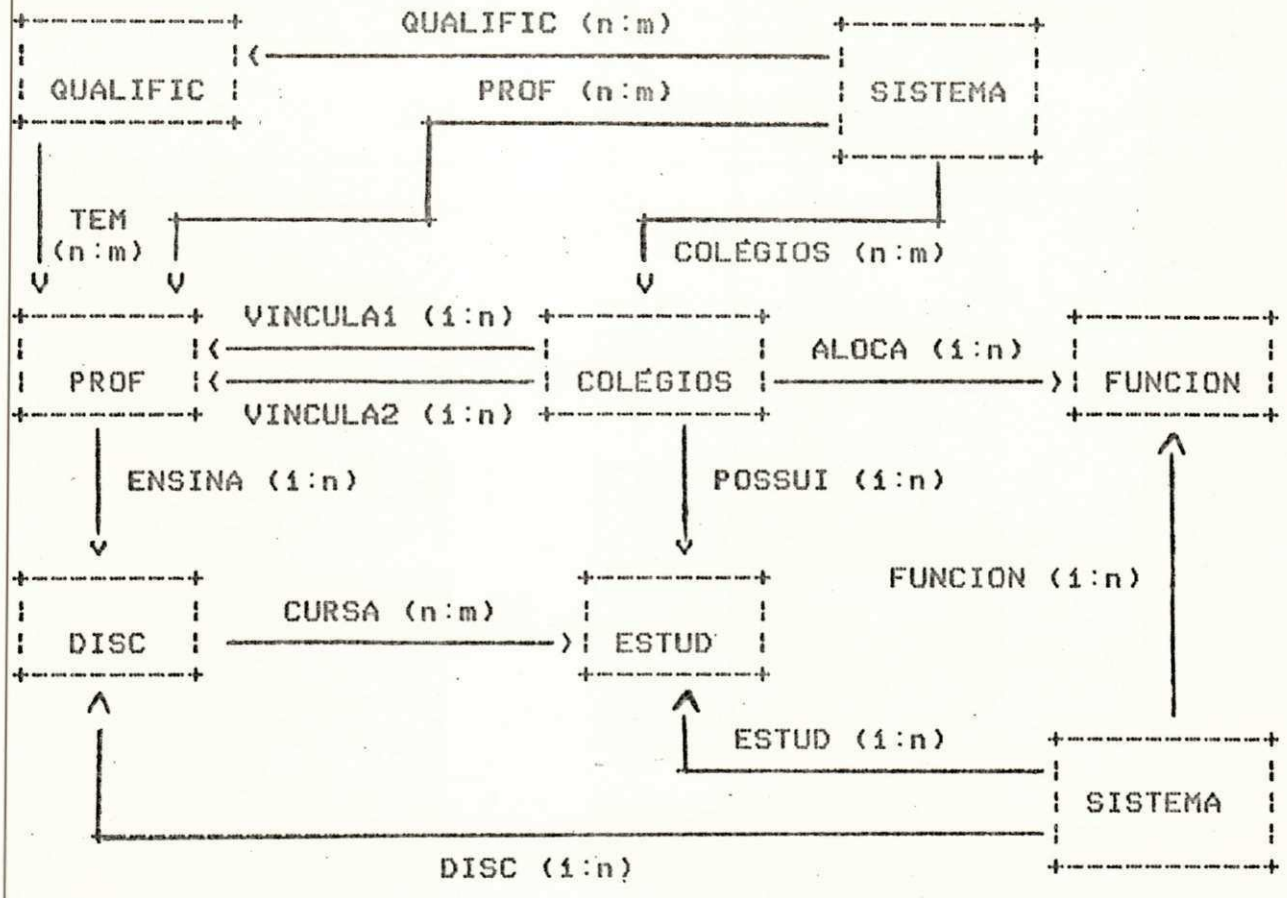
a)**Membro Corrente** - apontador para o último registro membro de um determinado relacionamento em que foi feito um acesso.

b)**Dono Corrente** - apontador para o último registro dono de um determinado relacionamento em que foi feito um acesso.

c)**Corrente Aplicação** - apontador para o último registro em que foi feito um acesso.

d)**Registro Corrente** - apontador para o último registro, de cada tipo de registro, em que foi feito um acesso.

Portanto, usando os elementos básicos do modelo adotado pelo SBD/TS, o esquema conceitual, no MERA, descrito no capítulo 2, ficaria conforme mostra a figura a seguir.



ESQUEMA CONCEITUAL "Colégios" REPRESENTADO NO MODELO ADOTADO PELO SBD/TS

O SBD/TS possui, na versão que utilizamos neste trabalho, duas listagens: uma para a definição e outra para a manipulação dos dados de um esquema, conforme descrevemos a seguir:

a) A Linguagem de Definição de Dados (LDD)

É a linguagem utilizada, pelo SBD/TS, na definição dos dados do DB. Ela é composta dos seguintes módulos sequênciais (adotaremos letras maíusculas para as palavras reservadas do SBD/TS; letras minúsculas entre '<>' para as declarações do usuário; e as declarações e palavras reservadas entre '[]' são