# ESPECIFICAÇÃO DA GERAÇÃO DE CÓDIGO E MÉTODO DE INTERPRETAÇÃO PARA INTERFACE MICROLOBAN

Vera Lúcia Vasconcelos Cavalcanti D'Albuquerque 1

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

Aprovada por:

Prof. Estevam de Simone
(Presidente)

Profa. Sueli Mendes dos Santos

Prof. Michael Stanton

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 1984

D'ALBUQUERQUE, VERA LÚCIA VASCONCELOS CAVALVANTI

Especificação da Geração de Código e Método de Interpretação para Interface MICROLOBAN (Rio de Janeiro), 1984.

VIII,173.p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1984).

Tese - Universidade Federal do Rio de Janeiro, COPPE.

1. Banco de Dados. I. COPPE/UFRJ II. Título (série).

Aos meus pais, Aureliano e Lenira, e as minhas irmãs Cristina, Silvia e Aurélia.

## AGRADECIMENTOS

Inicialmente gostaria de agradecer ao professor Estevam De Simone pela competente orientação, além do incentivo, dedicação, paciência e amizade sempre demonstrada.

À professora Lidia Micaela Segre pela grande amizade, paciência, incentivo e apoio emocional.

Aos colegas do projeto MINIBAN/COPPE, Beatriz Zakimi Miyasato, Antônio Carlos dos Santos, Antônio Claudio Gomez de Sousa e Jorge Silva Dantas.

Aos amigos do Departamento de Suporte da GEDES/BANERJ pelo incentivo e apoio demonstrado na fase final do trabalho.

Ao grande amigo Marcos Neme pelo apoio e incentivo demonstrado nesta fase de minha vida.

Aos amigos e professores da COPPE/SISTEMAS pela amizade e apoio durante o decorrer do Mestrado, em particular a Denise Schwartz Cupolillo.

Ao professor Jano Moreira de Souza pela orientação inicial do trabalho e pelo apoio dedicado.

Às amigas do Pensionato Santa Rosa de Lima pela ajuda emocional e consideração mostrada.

À CAPES, CNPq e FINEP pelo apoio financeiro.

À Suely Klajman pela amizade e trabalho datilográfico, e a Noema Menta pelos desenhos.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requeriments for the degree of Master of Science (M.Sc.)

SPECIFICATION OF THE CODE GENERATION
AND OF THE METHOD FOR MICROLOBAN INTERFACE

Vera Lúcia Vasconcelos Cavalcanti D'Albuquerque

Março de 1984

Chairman: Estevam De Simone

Department: Engenharia de Sistemas e Computação

This work is a proposal for an intermadiate code specification method of the class of data base acess language with functions to define and to manipulate the information in conceptual level and functions to define the structures and the operations on data base (phisical level).

As example, the developed methodology is applied on MICROLOBAN, a subset of LOBAN (Data Base Operation Language), obtaining as a result the intermediate language LIBAN.

Resumo da Tese Apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ESPECIFICAÇÃO DA GERAÇÃO DE CÓDIGO E MÉTODO DE INTERPRETAÇÃO PARA INTERFACE MICROLOBAN

Vera Lúcia Vasconcelos Cavalcanti D'Albuquerque

Março de 1984

Orientador: Estevam De Simone

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta uma proposta de especificação de um método criado para gerar código intermediário da classe de linguagens de acesso à banco de dados que possuem funções que descrevem e manipulam a informação e seu relacionamento à nível conceitual, e funções que definem as estruturas e as operações sobre a base de dados (nível físico).

Também é exposto um exemplo da aplicação do método proposto usando um subconjunto da Linguagem de Operação de Banco de Dados (LOBAN) denominado MICROLOBAN, resultando na linguagem intermediária LIBAN.

# ÍNDICE

Páginas

Páginas

# CAPÍTULO I

# INTRODUÇÃO

## 1.1. Histórico do Projeto MINIBAN

Devido a um convênio especial datado de junho de 1972, entre o CNPq e o GMD da República Federal da Alemanha sobre Cooperação Científico-Tecnológica, foi gerado o projeto denominado MINIBAN - Projeto de Sistema de Banco de Dados para Minicomputador Nacional, iniciado em Abril de 1977. Além dos órgãos CNPq e GMD, este projeto contava com a participação da DIGIBRÁS e da Universidade Federal do Rio Grande do Sul (UFRGS).

Inicialmente tinha-se como objetivos conhecer a capacidade, disponibilidade e confiabilidade dos sistemas portadores nacionais (hardware e software básico de microcomputadores), estudar os sistemas de banco de dados já utilizados no país para computadores de pequeno porte, e como última fase desta etapa, elaborar as alternativas de desenvolvimento de um software adequado, considerando o exposto anteriormente. CUNHA (13)

Sua segunda etapa previa a definição das estruturas de informação e suas funções disponíveis na interface entre Usuário e Banco de Dados (início de 1978). Dessa etapa resultou a especificação inicial da Linguagem de Operação de Banco de Dados (LOBAN) - CASTILHO (07).

Finalmente o projeto teve por meta a determinação do que seria a especificação e implementação de um subconjunto da linguagem LOBAN, como protótipo, que a princípio poderia ser preparado e executado em uma ou várias universidades do país. A primeira implementação denominada Sistema L (HEUSER

(18)), está sendo desenvolvida pela Universidade Federal do Rio Grande do Sul.

Em outubro de 1978, a COPPE/UFRJ iniciou sua participação neste projeto (MINIBAN), dando origem ao projeto MINIBAN/COPPE. Projeto este que teve como meta inicial o desenvolvimento de uma especificação detalhada de LOBAN, tendo como característica principal a separação nítida entre etapa de definição da interface usuário/banco de dados e sua realização ou implementação em um sistema portador (SANTOS (27), PINTO (23)).

Devido a extensão e a extrema complexidade da interface assim definida e ao desejo de adaptar LOBAN a um computador de pequeno porte, a equipe do Projeto MINIBAN/COPPE optou como segunda etapa do projeto (denominado MICROBAN) pelo desenvolvimento de um protótipo capaz de suportar um subconjunto LOBAN chamado MICROLOBAN. Desta forma foi escolhido o COBRA-300 (COBRA (09)) para ser o sistema portador, por ser uma máquina nacional e pela necessidade de desenvolvimento de software para sistemas deste porte. (SOUZA (30), DANTAS (14), CRIVOROT (12)).

## 1.2. Proposta, Objetivos e Organização da Tese

O projeto MICROBAN/COPPE teve seu início propriamente dito em fins de 1981. Seu objetivo inicial era a especificação de suas partes componentes, desta forma foi definida a estrutura geral do projeto, onde se pretendia fornecer uma visão geral dos seus componentes e de suas interfaces. A representação gráfica desta estrutura é especificada na figura 1.1.

Um ponto importante a ser frisado dentro desta es-

trutura, é a distinção feita entre os aspectos de especificação e de implementação de seus componentes. A estrutura geral apresentada tenta mostrar a diferença entre estes aspectos para que dúvidas posteriores sejam sanadas.

O presente trabalho está mais ligado a parte de especificação do Interpretador que corresponde a definição da Máquina virtual Intermediária MICROLOBAN. Composta de duas partes bem distintas, esta especificação engloba a comunicação com especificação do analisador/tradutor e com a do Executor.

A primeira parte do trabalho consistiu da especificação da interface entre o Analisador/Tradutor e o Interpretador. Como resultado obtivemos a especificação das árvores binárias cujos conteúdos dos nós são instruções da linguagem intermediária LIBAN, sendo algumas dessas instruções apresentadas no Anexo III.

Como segunda etapa foi definido o conjunto de algoritmos correspondentes as instruções da máquina Virtual intermediária MICROLOBAN, e do conjunto de dados que correspondem aos registradores da referida Máquina. A descrição informal desses algoritmos se encontram no Anexo IV.

A especificação e implementação do Analisador/Tradutor foram realizados através do trabalho de tese de CRIVOROT (12), tendo como saída as árvores LIBAN especificadas neste trabalho. Por outro lado, os algoritmos desenvolvidos utilizam de maneira bastante indireta, as primitivas de gerenciamento e armazenamento interno especificadas no trabalho de tese de DANTAS (14).

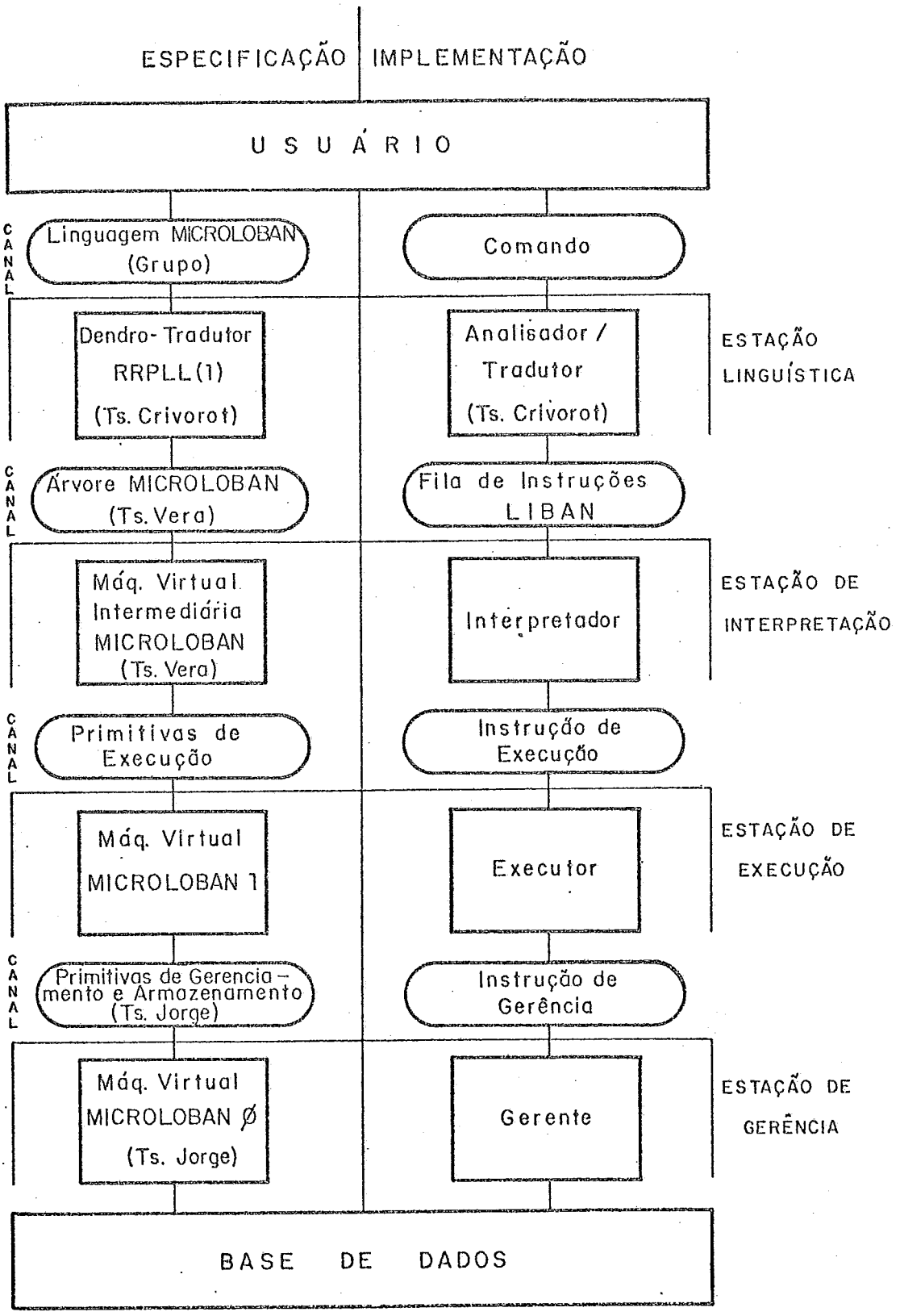



Figura 1.1 - Estrutura Geral do Projeto

Para descrição deste trabalho de tese foi encontrada uma certa dificuldade devido ao universo abrangente a ser considerado. Dentre os conhecimentos necessários para o desenvolvimento desta descrição, podemos citar os referentes a linguagem fonte MICROLOBAN e a linguagem intermediária LIBAN,e por fim o conhecimento relativamente profundo da semântica dos comandos fonte necessários na definição detalhada dos algoritmos desenvolvidos. (Anexo IV)

Como podemos observar, os Anexos III e IV particularmente correspondem ao "grosso" do resultado do trabalho de pesquisa realizado. Por outro lado, é fornecida uma visão geral dos detalhes de desenvolvimento da especificação proposta através dos Capítulos da referida tese.

O presente trabalho encontra-se organizado em capítulos, cada um com uma seqüência bem definida porém com um sequenciamento lógico que visa atingir os objetivos mostrados anteriormente.

O Capítulo II apresenta a descrição um tanto resumida do projeto MICROBAN. Serão fornecidos neste capítulo alguns pontos importantes no que diz respeito a interface MICROLOBAN, juntamente com suas estruturas de informação e as operações realizadas sobre elas. Finalizando será fornecida uma visão global de funcionamento da arquitetura geral do sistema.

O Capítulo III consta da descrição do método de especificação de um Interpretador para linguagens de Acesso a Banco de Dados. Além disto será detalhada a aplicação deste método utilizando a interface MICROLOBAN, e descrita a linguagem intermediária LIBAN gerada como resultado desta aplicação.

Finalmente o Capítulo IV apresenta as conclusões chegadas através do desenvolvimento do trabalho.

Além dos capítulos acima citados, vários anexos também são apresentados, contendo o necessário para a compreensão do método desenvolvido.

# CAPÍTULO II

## Projeto MICROBAN

### 2.1. Introdução a Interface MICROLOBAN

O projeto Microban tem como objetivo principal a definição e implementação de um subconjunto da linguagem de Operação de Banco de Dados (LOBAN) (SANTOS (27)). Este subconjunto, denominado Microloban, foi especificado sem a preocupação de como suas estruturas seriam realizadas fisicamente, possibilitando desta forma a escolha de uma solução qualquer para a implementação das mesmas no sistema portador.

Suas estruturas de informação possibilitam uma visão global da informação, além de que as operações sobre estas solucionam a grande parte dos problemas de tratamento de informação na área de Banco de Dados. Na definição das estruturas e suas operações procurou-se preservar as consideradas básicas em LOBAN, de forma que futuras extensões possam ser adicionadas sem contudo alterar o núcleo básico do protótipo.

Sendo uma interface considerada poderosa, inicialmente sua definição e implementação terá como finalidade principal servir de ferramenta de ensino na área de banco de dados, funcionando como um sistema de referência na comparação com diversas abordagens.

O principal critério adotado na definição da interface MICROLOBAN foi reduzir a complexidade de LOBAN a um nível implementável pelo sistema portador utilizado na sua realização, sem modificar sua filosofia básica.

A separação entre os aspectos funcionais e os aspec-

tos de realização leva também à independência da interface quanto aos dispositivos de entrada e saída utilizados. Desta forma existe um único grupo de funções de definição e manipulação, ou seja, não importa quais sejam os dispositivos de entrada e saída pois o usuário MICROLOBAN terá a sua disposição sempre a mesma interface.

Por outro lado, procurou-se dar ao usuário uma ferramenta para definir e implantar sistemas de informação em geral e não apenas um sistema específico, além de ter a possibilidade de tratar a informação sob qualquer das três principais abordagens sobre banco de dados: hierarquica, de redes ou relacional.

MICROLOBAN é uma interface muito concisa e com funções relativamente poderosas, possuindo seu conjunto de instruções na língua portuguesa para que sua linguagem permitisse a autodocumentação do sistema definido. A definição da linguagem, isto é, sua meta-linguagem, é também em português por motivos semelhantes. Isto fez com que suas instruções fossem mais extensas que a maioria das linguagens de programação, tornando-as contudo mais compreensiveis.

O Banco de Dados será constituido das seguintes unidades funcionais (RICHTER (25)) : uma Base de Dados, um canal auxiliar e um sistema de Gerência da Base de Dados. Sua representação gráfica é mostrada na figura 2.1, a seguir.

A Base de Dados (BD) é a unidade funcional que armazena informações de forma permanente e segundo um enfoque determinado. O usuário pode incluir, retirar, modificar ou obter informações da Base de Dados.

O Canal Auxiliar (CA) é a unidade funcional que arma

zena as informações temporariamente durante a execução de um serviço, sendo uma espécie de base de dados particular a cada usuário.

O Sistema de Gerência de Base de Dados (SGBD) é a unidade funcional que administra tanto a base de dados quanto o canal auxiliar, executando as instruções fornecidas pelo usuário. É dotado de funções que vão desde a interpretação de instruções passando pela leitura e armazenamento tanto de entrada como das definições a serem incluidas na própria Base de Dados, até a emissão de diagnósticos sobre as instruções submetidas.

O diálogo entre as unidades funcionais Usuário e o Banco de Dados é realizado por meio da interface MICROLOBAN, através da linguagem de mesmo nome.

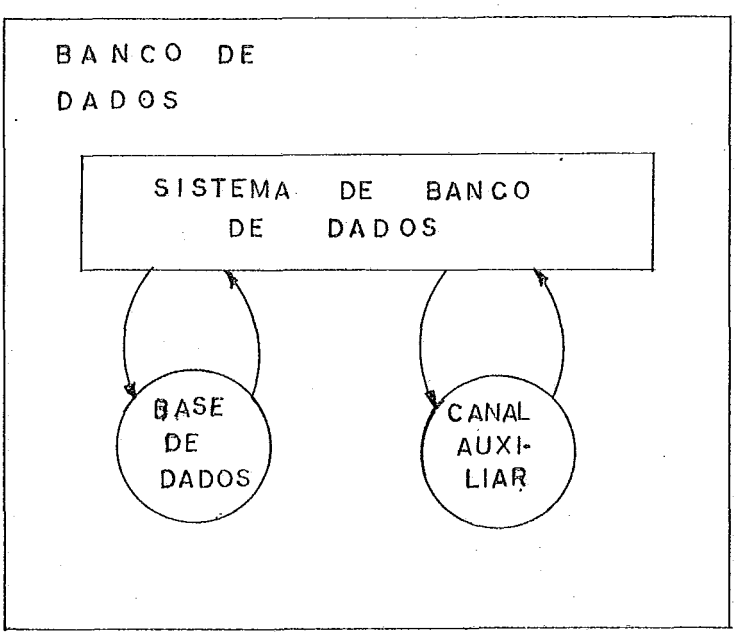



Figura 2.1. - Representação Gráfica do Banco de Dados.

A seguir serão relacionadas as principais características funcionais e estruturais da interface MICROLOBAN, realizada pela sua linguagem. São elas:

- Autocontida

Possui funções que descrevem a informação e seu relacionamento (esquema), funções para descrever as estruturas da base de dados (DDL), funções para manipulação do esquema e funções para utilização da base de dados (DML).

- Operações globais

As instruções da linguagem de comunicação entre usuário e banco de Dados (MICROLOBAN) correspondem a operações complexas, cuja complexidade o usuário não precisa conhecer.

- Processamento em lotes

Grande quantidade de lote de dados são processados de forma sequencial como nos sistemas de informação clássicos.

- Pré-definição de procedimentos

Texto fonte pode ser submetido com antecedência sendo armazenado na base de dados e posteriormente referenciado como procedimento previamente definido.

- Pré-definição da coerência dos dados

Consistência da informação e a sua persistência (modificações permitidas) podem ser pré-definidas e guardadas na Base de Dados.

- Definição de mecanismos de segurança

Mecanismos de segurança para os usuários e regulamentação de acesso conforme esses mecanismos podem ser definidos por meio das expressões de especificação de acesso e das instruções para sua manipulação. As especificações de acesso também são pré-definidas e armazenadas na Base de Dados.

- Reconstrução da Base de Dados

Disponibilidade de medidas que permitem reconstruir a Base de Dados e reiniciar serviços em caso de erro ou falha de processamento.

- Processamento interativo

Forma conversacional pode ser utilizada para processamento de pequeno volume de dados e que exija uma execução passo a passo.

Algumas das características principais de LOBAN foram desconsideradas na especificação do subconjunto devido o porte do sistema portador escolhido, COBRA-300, e também por serem estas características extensões que poderão ser realizadas após o núcleo básico ter sido implementado. Das características estruturais e funcionais que não estão contidas no subconjunto, poderemos citar dentre outras a pré-definição de formatos de relatórios de saída feita através de expressões da própria linguagem e armazenadas na base de dados, e a pré-definição de estratégias na utilização de recursos que influenciam nos custos relativos e na qualidade do serviço requisitado pelo usuário.

A interface não prevê acesso múltiplo concomitante ao mesmo "acervo setorial", isto é, só permite que sejam feitas consultas e/ou modificações na base de dados por apenas um usuário MICROLOBAN em um determinado intervalo de tempo.

Além das saídas requisitadas pelo usuário, o sistema fornecerá uma ata de execução que é um relatório padrão sobre a execução de uma instrução de trabalho, sendo considerada a maior unidade de comunicação da interface. Uma ata de execu

ção será composta de várias mensagens operacionais inseridas automaticamente no decorrer de um determinado evento.

## 2.2. Linguagem MICROLOBAN

MICROLOBAN é uma linguagem autocontida, com instruções escritas em português razoavelmente extensas possibilitando dessa forma uma autodocumentação.

A linguagem engloba diversas funções entre as quais estão as que descrevem a informação e seu relacionamento , as estruturas de informação utilizadas na Base de Dados (DDL), as funções de manipulação e uso dos dados contidos na base de dados (DML), e funções que normalmente são realizadas por utilitários na maioria dos Sistemas de Gerência de Banco de Dados (SGBD), tais como: reconstrução de acervo, modificação das definições de dados, etc ...

Qualquer serviço solicitado ao sistema deverá ser submetido através de uma instrução de trabalho, que é constituida de uma instrução de início (<executar>), uma lista de instruções autônomas e uma instrução de fim (<encerrar>). Podemos distinguir dois tipos de instrução de trabalho: aquela interativa, onde os comandos são submetidos pelo usuário um de cada vez, sendo executados imediatamente; e a instrução de trabalho não interativa, onde o sistema só inicia a execução das instruções fornecidas após ter recebido a instrução <encerrar>. A sintaxe da

Linguagem MICROLOBAN se encontra especificada no Anexo I e II, para conhecimento da especificação completa das instruções. No Anexo I é fornecida a sintaxe utilizando a notação usada na especificação LOBAN.

Por meio da instrução <executar> o usuário determina algumas características de execução da instrução de trabalho além de indicar o início de uma sessão MICROLOBAN e informar sua identificação. Dentre outras características desta instrução de início, podemos citar a indicação do tipo de processamento, ou seja, "batch" ou interativa, requerimento de que apenas a compilação (análise léxica e sintática) seja realizada, atribuição de um nome a instrução de trabalho, etc...

Os dados a serem processados serão fornecidos ou junto com as instruções como anexos, ou separados em arquivos externos. Como resultado do processamento dos dados, são fornecidos o resultado externo (ex: relatório), e o interno que é guardado como uma nova base de dados caso tenham sido feitas alterações sobre a mesma. Serão também fornecidas mensagens operacionais padronizadas informando as ocorrências durante o processamento de cada instrução submetida ao sistema.

A lista de instruções autônomas é formada por grupos distintos de comandos que executam funções, tais como: gerência, manipulação, alocação de recursos, definição de "snap-shots", reconstrução, controle de fluxo, saída de dados e definição de transação. A seguir serão definidos os conceitos básicos necessários a compreensão das estruturas de informação e das operações realizadas na Linguagem MICROLOBAN.

## 2.3. Conceitos Básicos

A definição de alguns conceitos se fazem necessários para uma compreensão relativamente satisfatória das estruturas de informação e das operações sobre estas, consideradas na interface MICROLOBAN.

Entretanto pedimos ao leitor mais profundamente interessado em conhecer os fundamentos teóricos e detalhes característicos da interface MICROLOBAN, referir-se diretamente ao manual completo da linguagem LOBAN , de onde MICROLOBAN se originou (DURCHHOLZ (16), DURCHHORZ (17), RICHTER (26), SANTOS (27)).

Iniciaremos por ressaltar a distinção entre a informação e sua representação. A informação está a nível de abstração, ao passo que a representação está a nível de meio físico, isto é, como a informação é armazenada internamente na Base de Dados. Para o usuário só interessam as informações e o enfoque, vista ou organização, que ele tem da informação.

A cada estrutura de informação referenciável pelo usuário chamamos de construção. Uma construção pode ser um agregado de outras construções. Os componentes imediatos de um agregado podem ter nomes ou não, qualificando dessa forma tipos de construção básicos.

Chamamos de coleção a uma construção em que os componentes imediatos são construções sem nomes. Sua representação gráfica está na figura 2.2.

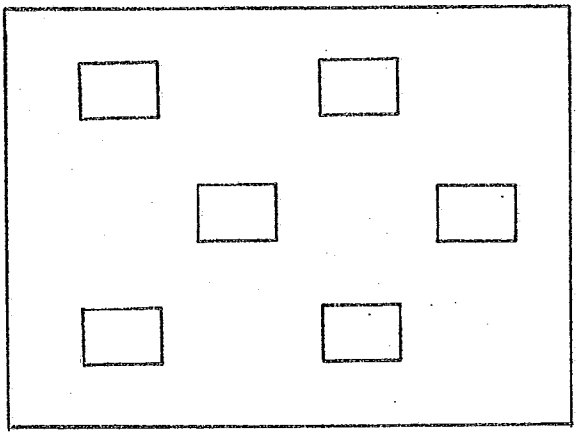


Figura 2.2. - Representação Gráfica de uma Coleção.

Outro tipo de agregado é a Nominação cujos componentes imediatos são construções sob nomes. Uma nominação é definida por pares (nome, componente imediato), portanto a um determinado nome está associado um componente imediato. Uma observação a ser feita é que a um mesmo componente imediato não pode estar associado vários nomes. A representação gráfica para este tipo de construção se encontra na figura 2.3

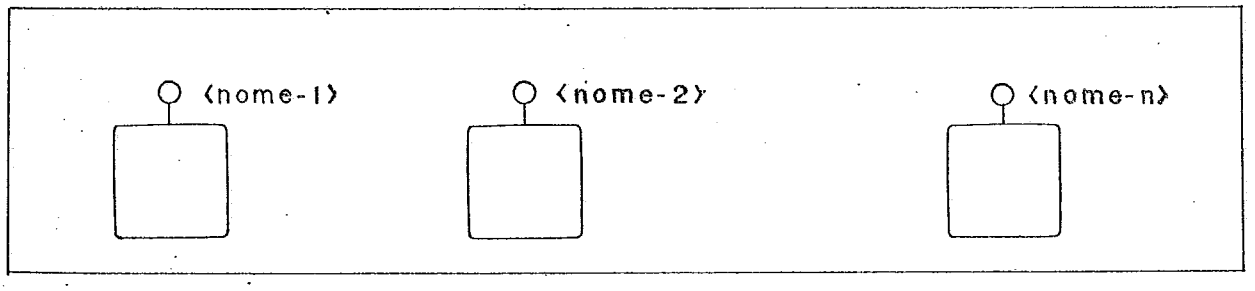



Figura 2.3. - Representação Gráfica de uma Nominação

Um caso especial de uma coleção, denominado de coletivo, caracteriza-se pelo fato de seus componentes imediatos serem nominações que possuem nomes comuns. Vale ressaltar que os nomes dos componentes imediatos de cada nominação devem pertencer ao mesmo conjunto de nomes. A figura 2.4 representa graficamente um exemplo de coletivo.

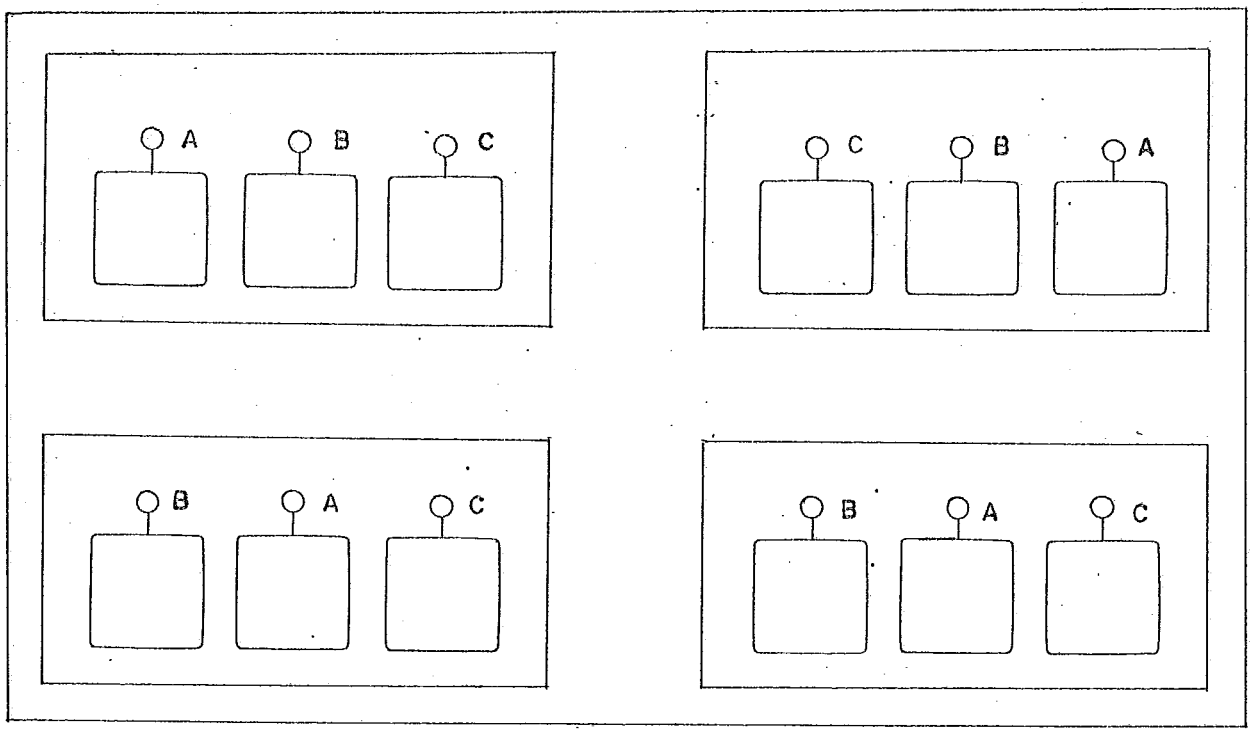



Figura 2.4. - Representação Gráfica de um Coletivo.

Finalmente temos a construção denominada numeração. Uma numeração é uma nominação cujos nomes sob os quais se encontram os componentes imediatos são a série de números naturais a partir de 1 (um) e com sucessão monótona crescente de razão igual a 1. Um exemplo de numeração é representado graficamente na figura 2.5.

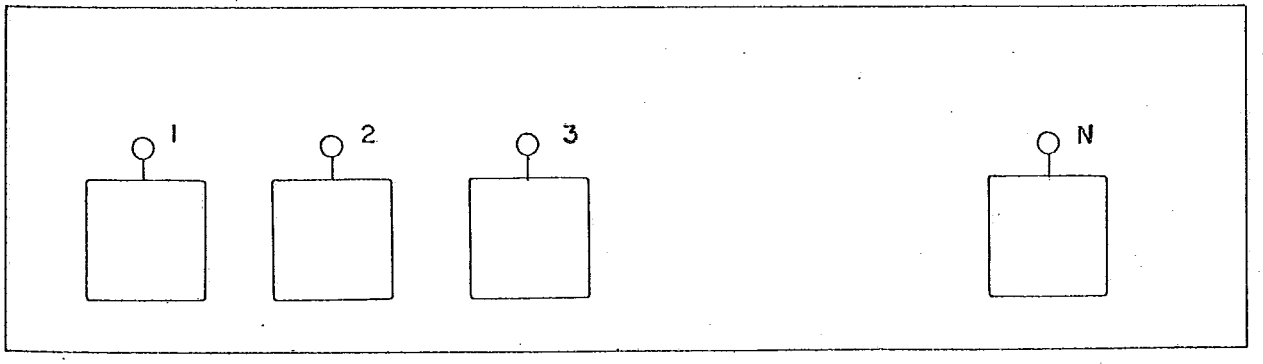



Figura 2.5. - Representação Gráfica de uma Numeração.

Agora partiremos para definir o que seja pretipo de uma construção. Um pretipo, tipo de contrução pré-estabeleci-

do, é um conjunto de construções não necessariamente finito nem componente do acervo (Base de Dados). Seus elementos são possuidores dos mesmos predicados comuns ao pretipo. Portanto,a condição básica de pertinência de um elemento ao pretipo é satisfazer aos predicados do pretipo. Esses predicados estão embutidos no banco de dados (pré-definidos), fazendo parte da interface MICROLOBAN como uma das convenções que permitem a comunicação Usuário/Banco de Dados.

Podemos definir novos tipos de construções a partir dos tipos primitivos da interface (pretipo).Um predicado comum as construções de um mesmo tipo é a sua pertinência ao mesmo pretipo. Os tipos de construções são definidos pelo usuário e suas regras dependem do pretipo da construção.

Uma construção pode ser ocorrência de um determinado tipo de construção, ou seja, elemento de um conjunto de construções definido. Uma diferença básica é que o nome de uma construção não deve ser confundido com a designação do tipo ao qual a construção, sob nome considerado, pertence. Dessa forma um nome serve para selecionar um componente imediato de uma nominação, e uma designação de tipo é utilizado para se fazer referência a um tipo de construção.

Um dos conceitos mais importantes usado em MICROLOBAN é o de ponto. Um ponto identifica a posição relativa de uma construção dentro de um contexto de referência. Assim sendo o conceito de ponto permite localizar uma construção em um contexto determinado apesar dela aparecer em outros pontos. Este conceito é definido como sendo uma seqüência de pares (nome, construção sob este nome) desde o nível de referência (contor-

no do contexto), até o nível da construção considerada neste contexto. Qualquer acesso a componentes da base de dados é realizado através do endereço de ponto (Figura 2.6). (Maiores detalhes SANTOS (27), capítulo 10).

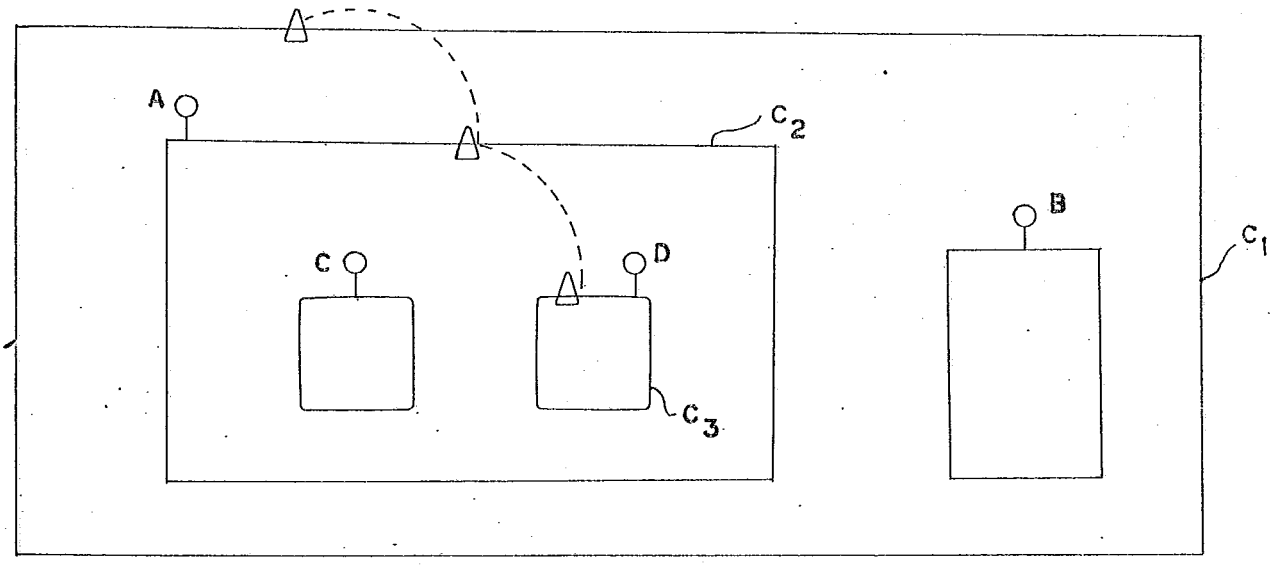



Figura 2.6. - Representação Gráfica de um Endereço de Ponto.

Outro conceito de relativa importância é o de campo. Um campo é uma área demarcada em volume do meio físico podendo ser subdividido em subcampos (também considerados campos) segundo uma máscara sobreposta a ele. Para localizar um campo dentro de um campo de referência também se utiliza uma seqüência de expressões booleanas denominada de endereço de campo. (Maiores detalhes em SANTOS (27), Capítulo 11).

## 2.4. Pretipos suportados pela Interface MICROLOBAN

MICROLOBAN possui alguns dos pretipos de LOBAN considerados básicos na composição do conjunto das estruturas de informação que compõem a Base de Dados e o Canal Auxiliar. A seguir serão descritos sumariamente os pretipos utilizados no subconjunto. Para uma visão geral e mais detalhada sobre pretipos de construções de LOBAN ver SANTOS (27) capítulo 4.

Chamamos de pretipos atômicos ao conjunto de construções indecomponíveis. Desses pretipos somente serão utilizados os átomos numéricos real e inteiro, com designação REAL e INT respectivamente.

Um conjunto de pretipos agregados básicos faz parte de MICROLOBAN, possuindo como elementos os seguintes pretipos:

- Numeração de caracteres

É uma nominação cujos componentes imediatos são caracteres sob nomes numéricos. Sua designação padrão é NUMCAR.

- Data

É uma nominação cujos nomes dos seus componentes imediatos são DIA, MÊS e ANO respectivamente. O acesso aos componentes do pretipo Data não será permitido. A designação deste pretipo é DATA.

- Hora

É uma nominação cujos nomes de seus componentes imediatos são HORA, MIN e SEG respectivamente. Também não será permitido endereçamento a seus componentes imediatos e sua designação padrão é HORA.

- Tupla

É uma nominação de nomes escolhidos pelo usuário sobre átomos ou tuplas. Sua designação padrão é TUP. Uma restrição feita com relação a este pretipo foi o fato de só ser permitido um nível de embutimento, ou seja, uma tupla, componente imediato de uma nominação, só poderá ser composta de átomos. A representação gráfica se encontra na figura 2.7.

- Ligação

É uma nominação de nomes Padrões L e T sobre uma tupla e uma "tabela relacional". Sua designação padrão é LIG. A representação gráfica se encontra na figura 2.8.

- Coleção de itens

É uma coleção cujos componentes imediatos são construções do pretipo itens pertencentes ao mesmo domínio. Um pretipo item é formado pelo conjunto de construções definido pelos pretipos numeração de caracteres, inteiro, real, data e hora. A designação padrão para coleção de itens é COLITENS.

- Tabela relacional

É um coletivo de tuplas de mesmo tipo. Cada ocorrência de uma tupla corresponde a uma linha da tabela (componente imediato da tabela relacional). Assim os atributos são seqüências de nomes comuns a todos os elementos de uma coluna, aplicando-se a cada linha (DATE(15)). A designição padrão (para este tipo de construção é TAREL. Sua representação gráfica se encontra na figura 2.9.

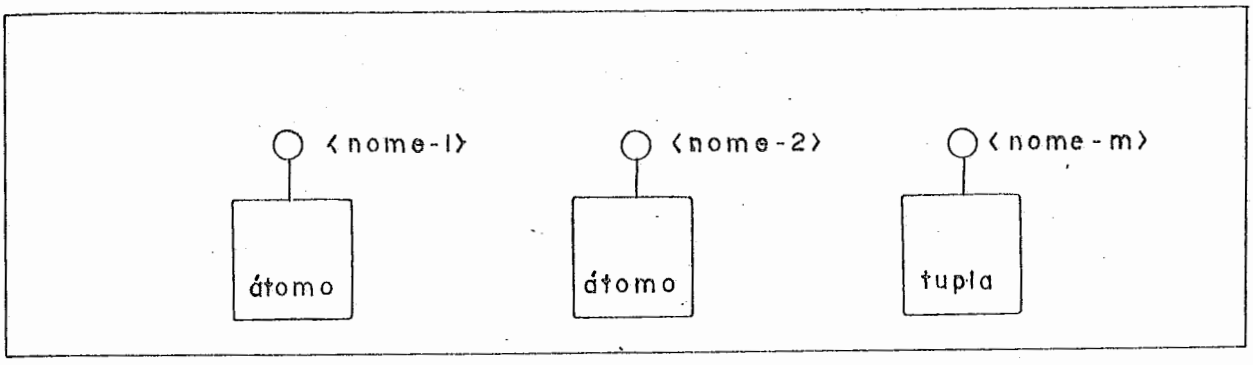



Figura 2.7. - Representação Gráfica de uma Tupla.

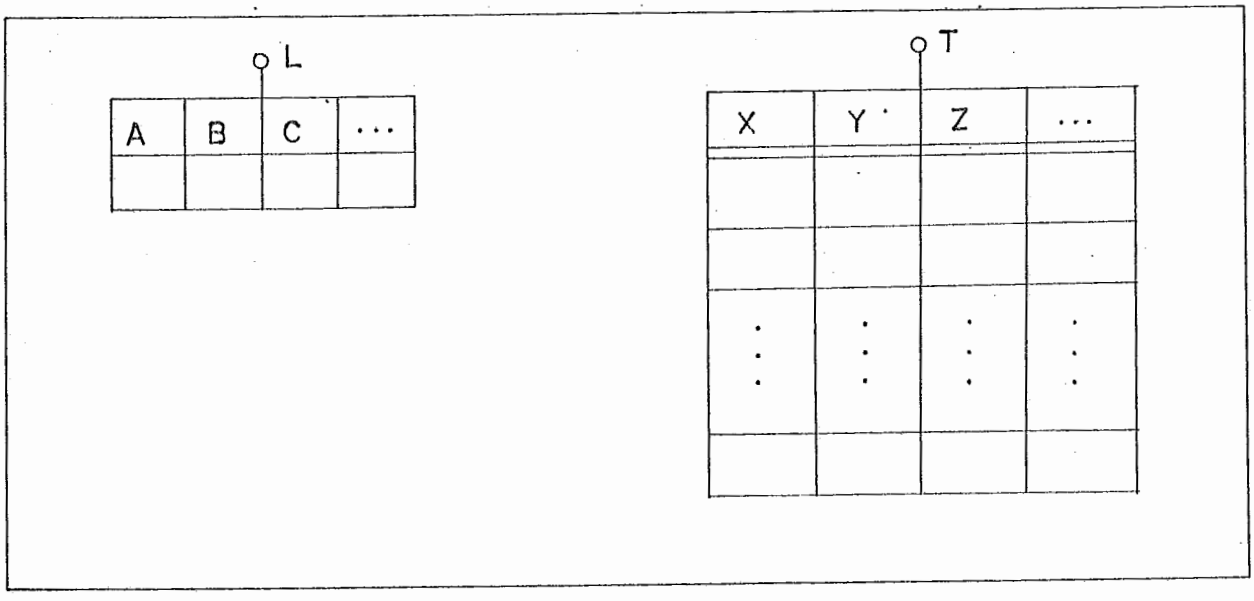



Figura 2.8. - Representação Gráfica de uma Ligação.

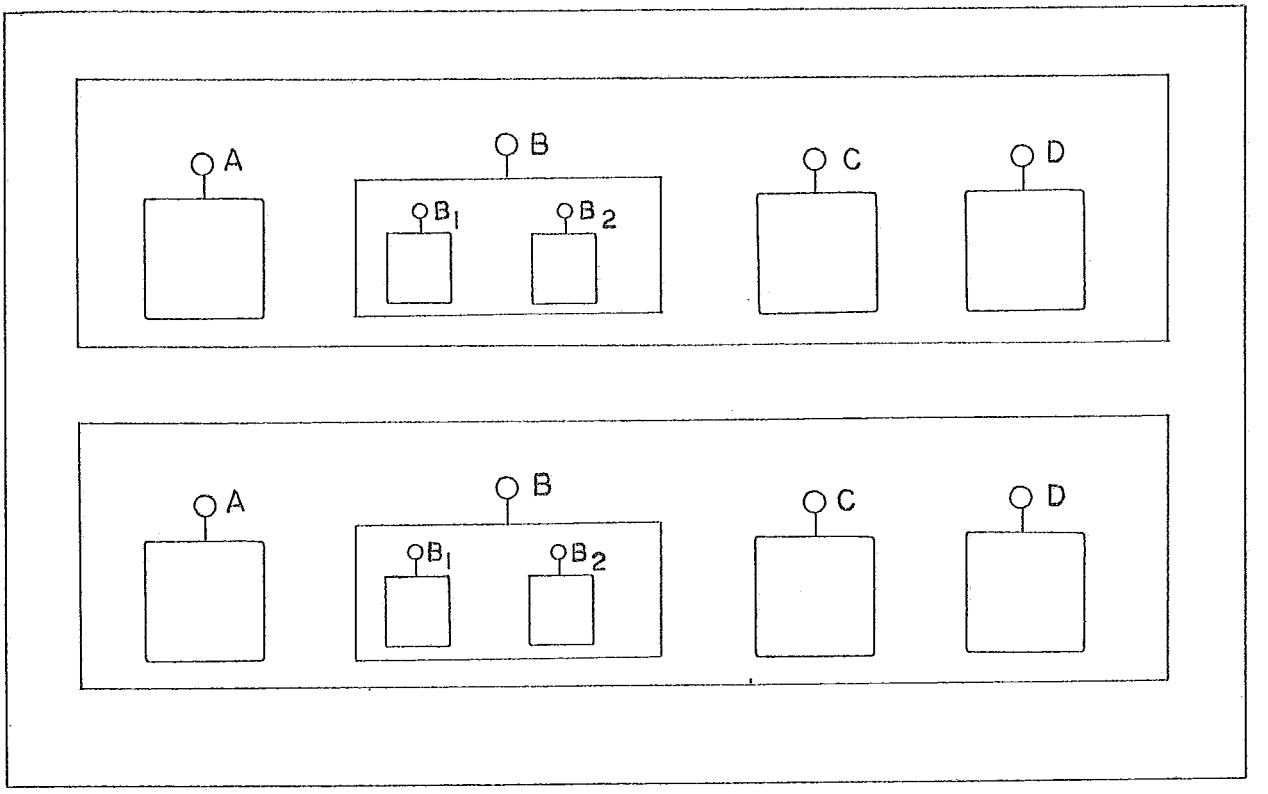



Figura 2.9. - Representação Gráfica de uma Tabela Relacional.

- Tabela ligacional

É uma coleção homegênea de ligações do mesmo tipo. Sua designação padrão é TALIG. Da mesma forma que uma tabela relacional, a ligacional também caracteriza-se por apresentar um conjunto de um ou vários atributos da tupla ligante cujo valor identifica univocamente cada ligação da tabela ligacional. A figura 2.10 contém a representação gráfica de uma tabela ligacional.

Folha

Como LOBAN, MICROLOBAN caracteriza-se por apresentar as descrições dos dados presentes ao acervo (esquema conceitual, DATE(15)), armazenadas como construções endereçáveis e manipuláveis pelo usuário desde que tenha autorização para isso. Além dos verbetes (expressões) que descrevem a consistência, existem os verbetes de acesso que definem autorizações de acesso

aos dados presentes no acervo, os verbetes de usuários que identificam os usuários para o sistema, e por fim os verbetes de fonte que pré- definem procedimentos através de texto fonte.

A representação gráfica para este pretipo se encontra na figura 2.11. A construção folha possui quatro componentes imediatos pré-definidos, correspondendo para cada um deles uma tabela ligacional contendo descrições sobre a consistência (CV-COERÊNCIA), sobre autorização de usuários (CV-ACESSO), sobre texto fonte (CV-FONTE), e sobre os direitos de acesso de um usuário autorizado (CV-ACESSO). A designação padrão para folha é FOLHA. Restrições foram feitas a composição da FOLHA de LOBAN, pois em MICROLOBAN não temos as descrições referentes a pré-definição de máscaras de usuários a serem usadas em entrada e saída de dados, bem como as descrições sobre pré-definição de utilização de recursos.

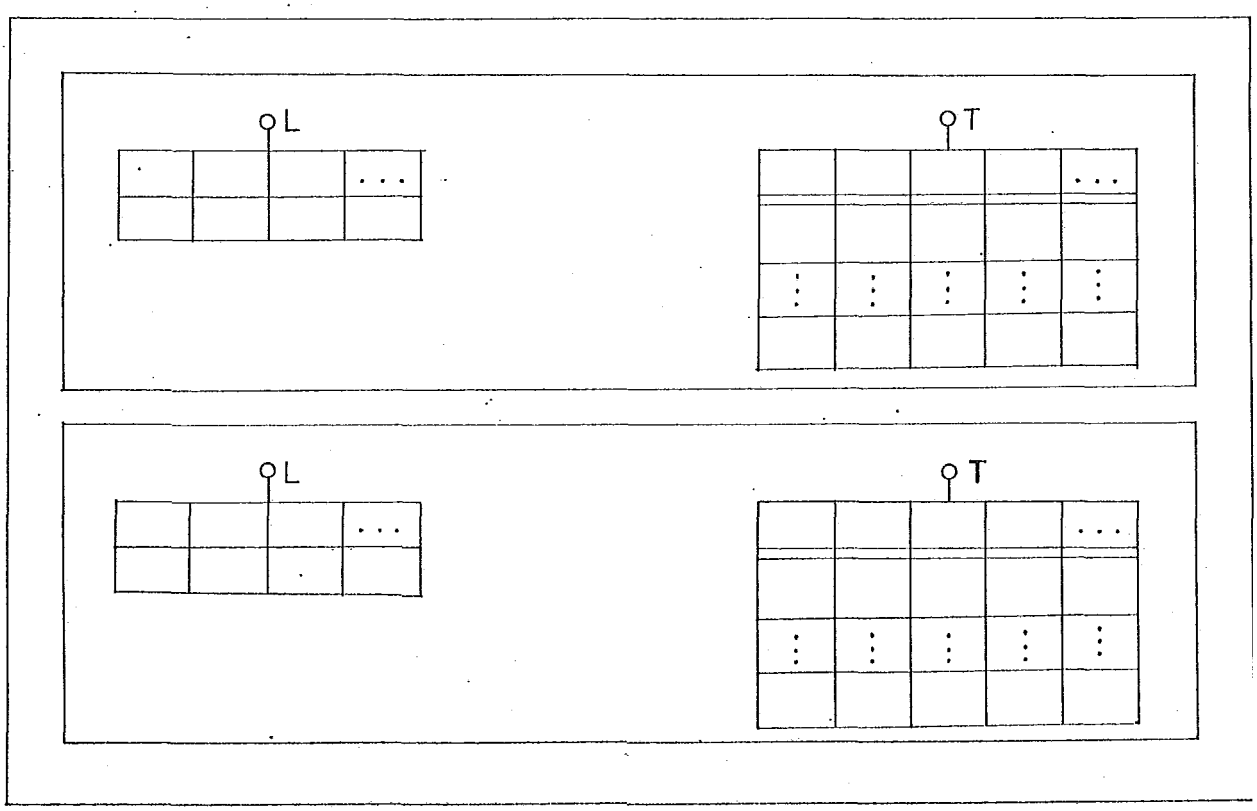



Figura 2.10. - Representação Gráfica de uma Tabela Ligacional.

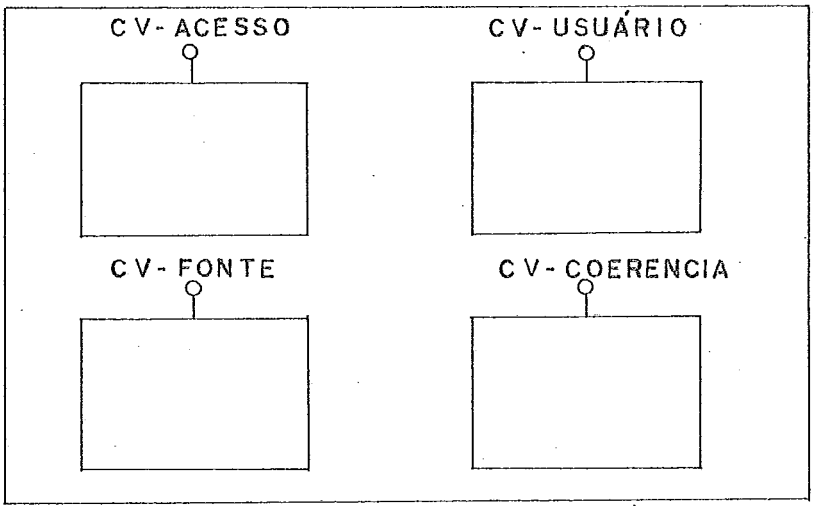



Figura 2.11. - Representação Gráfica de uma Folha.

- Ficha

A ficha em MICROLOBAN é uma tupla de tipo e nome padrão que traz informações mantidas pelo sistema sobre as construções do tipo arquivo (descrita posteriormente), e sobre o "acervo setorial" (Base de Dados). Essas informações constituem da data de criação e da data da última utilização da referida construção, podendo as mesmas serem somente consultadas pelo usuário. Sua representação gráfica se encontra na figura 2.12. Sua designação padrão é FICHA

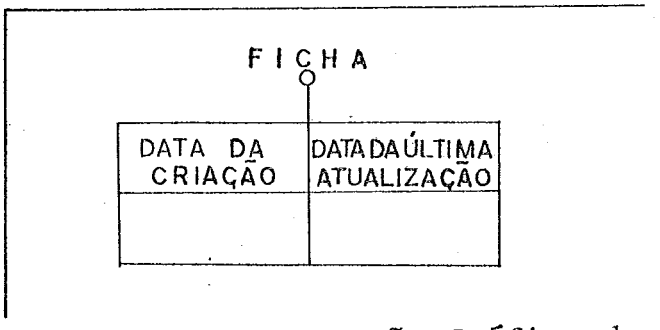



Figura 2.12. - Representação Gráfica de uma Ficha.

- Arquivo

É uma nominação composta por uma construção sob nome FICHA e outra sob nome TR ou TL, identificando uma tabela relacional ou ligacional respectivamente. Quando a tabéla é relacional, dizemos que o arquivo é do tipo relacional tendo designação padrão AREL. Um arquivo ligacional por sua vez é composto de uma tabela ligacional, possuindo designação padrão ALIG. As representações gráficas para arquivo relacional e ligacional são apresentadas na figura 2.13.

- Acervo de Trabalho

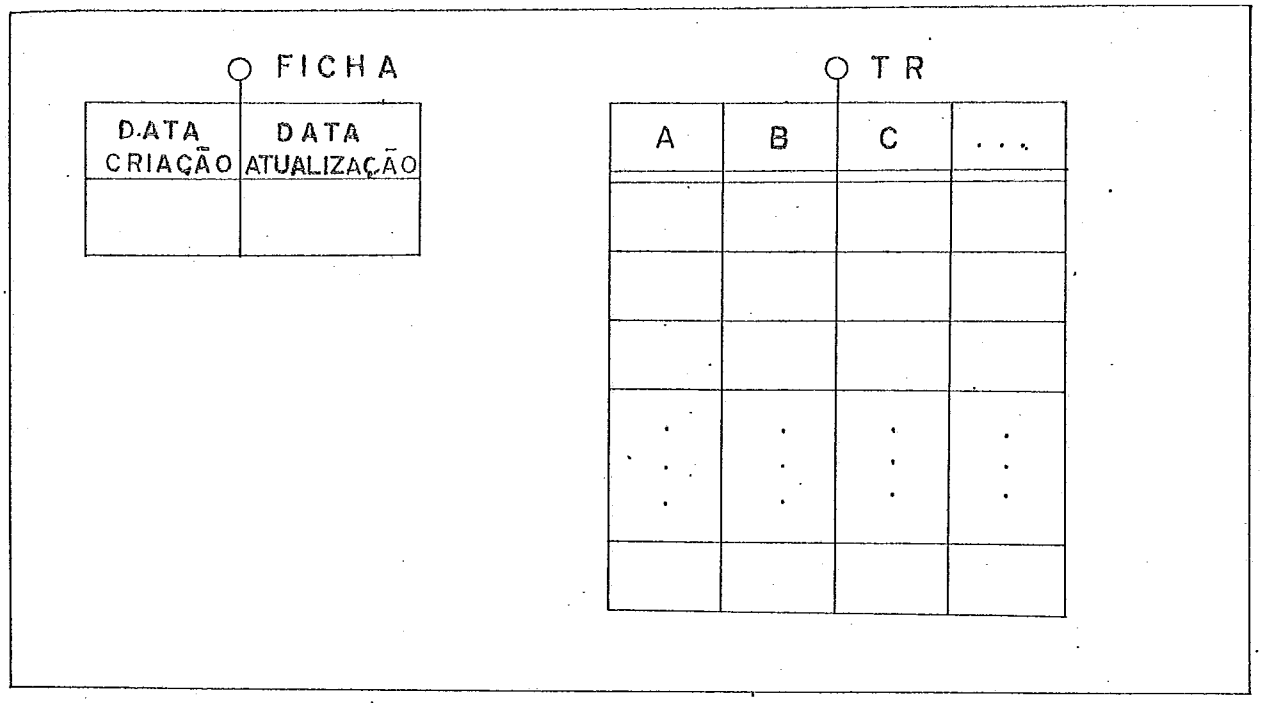
O Acervo de Trabalho é uma nominação cujos componentes imediatos são: uma construção de nome e pretipo FICHA (informações sobre o acervo de trabalho), uma construção de pretipo e nome FOLHA (esquema conceitual, autorização, acesso, texto fonte), e uma ou várias construções de pretipo arquivo Relacional ou Ligacional. A designação padrão para Acervo de Trabalho é ACTRAB. A figura 2.14 esquematiza a composição de um acervo de trabalho.

- Acervo Setorial

Um Acervo Setorial é uma Nominação cujos componentes imediatos são:

- O Acervo de Trabalho (ACTRAB); e

- Uma cadeia de reconstrução (CAD-REC) a qual informa as alterações feitas no ACTRAB utilizada na reconstrução do mesmo. Esta cadeia só é acessada pelo SGBD para registrar as alterações e executar o procedimento de reconstrução do ACTRAB. A designação padrão para o acervo setorial é ACSET.




Arquivo Relacional

Arquivo Ligacional

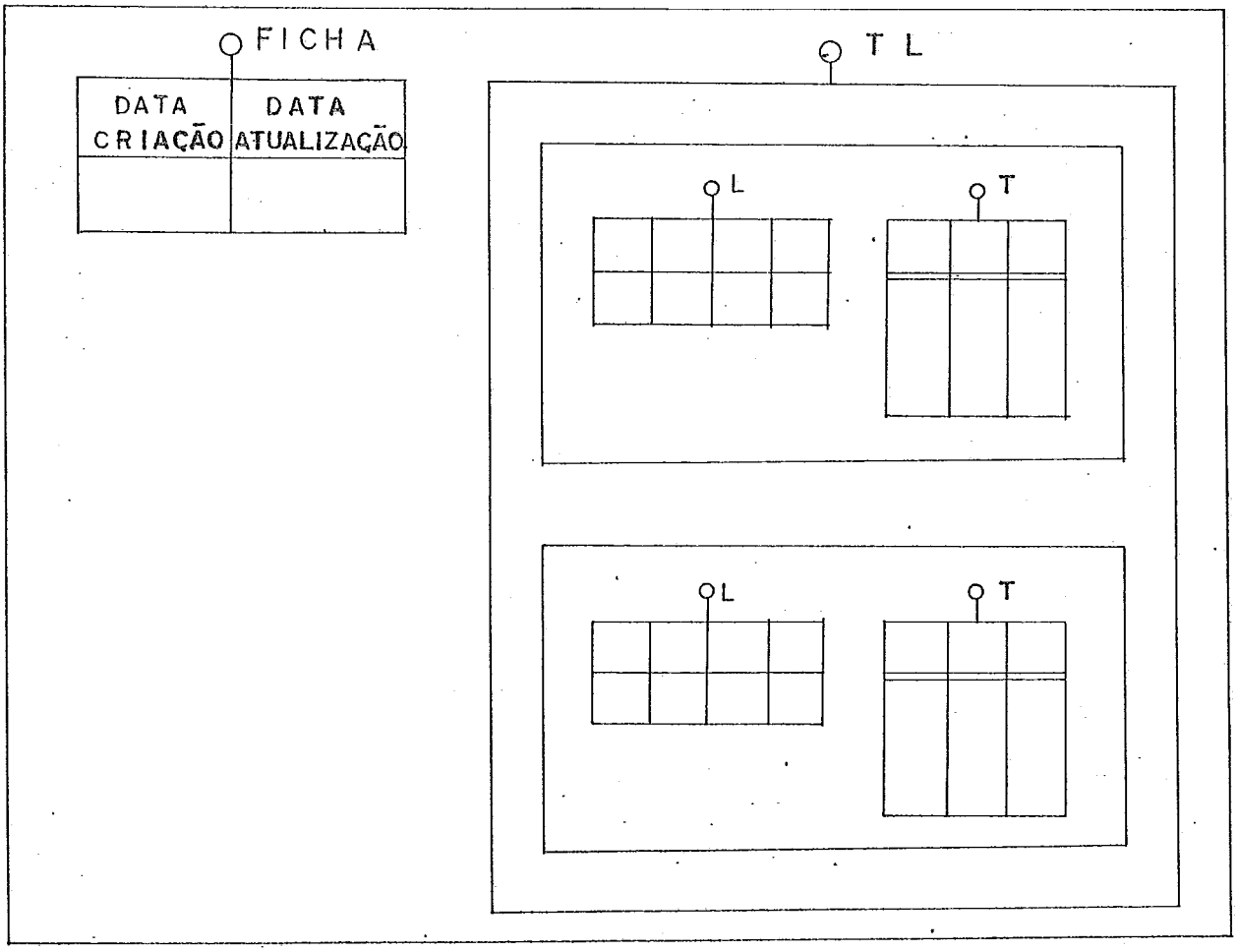



Figura 2.13. - Representação Gráfica de um Arquivo.

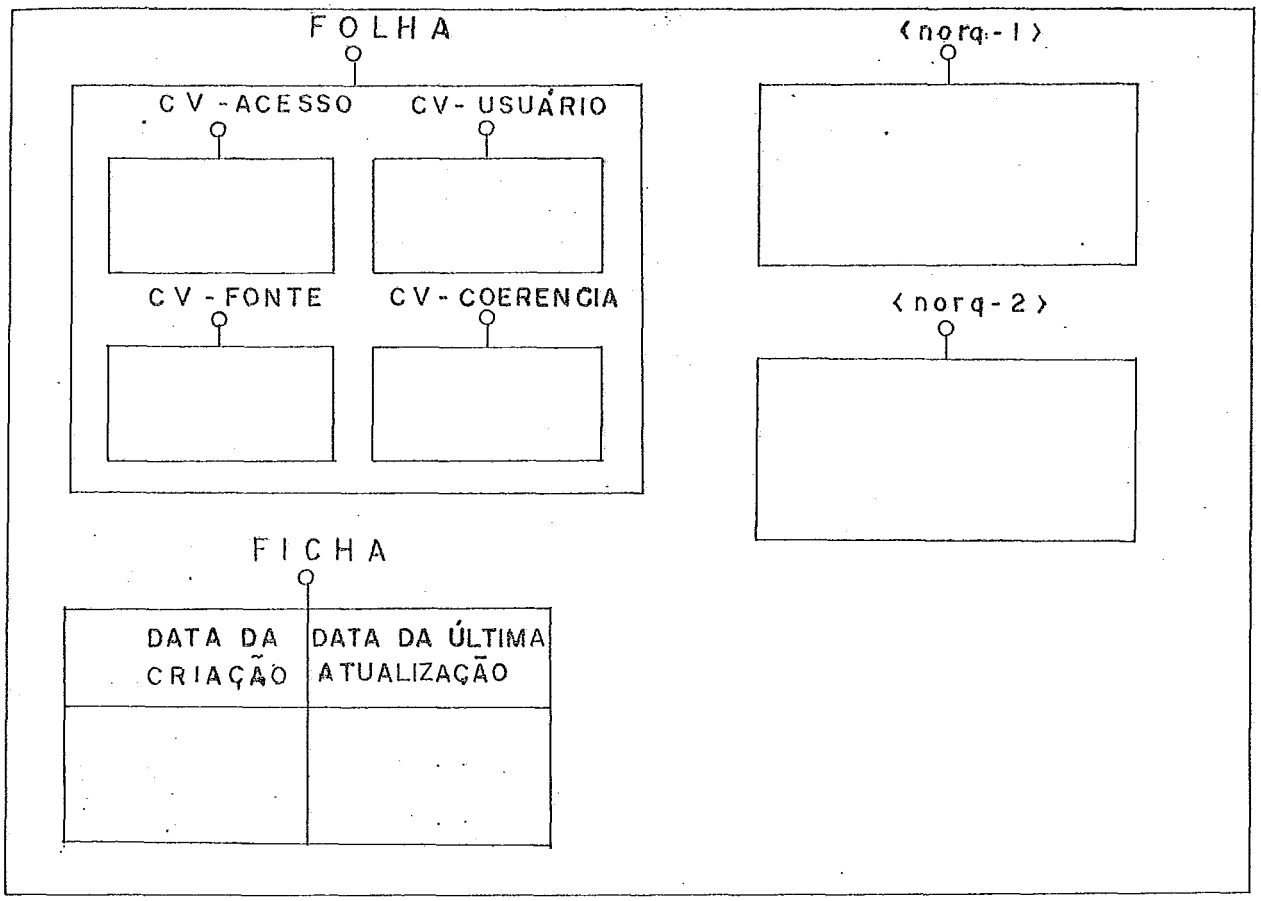



Figura 2.14. - Representação Gráfica de um Acervo de Trabalho (ACTRAB).

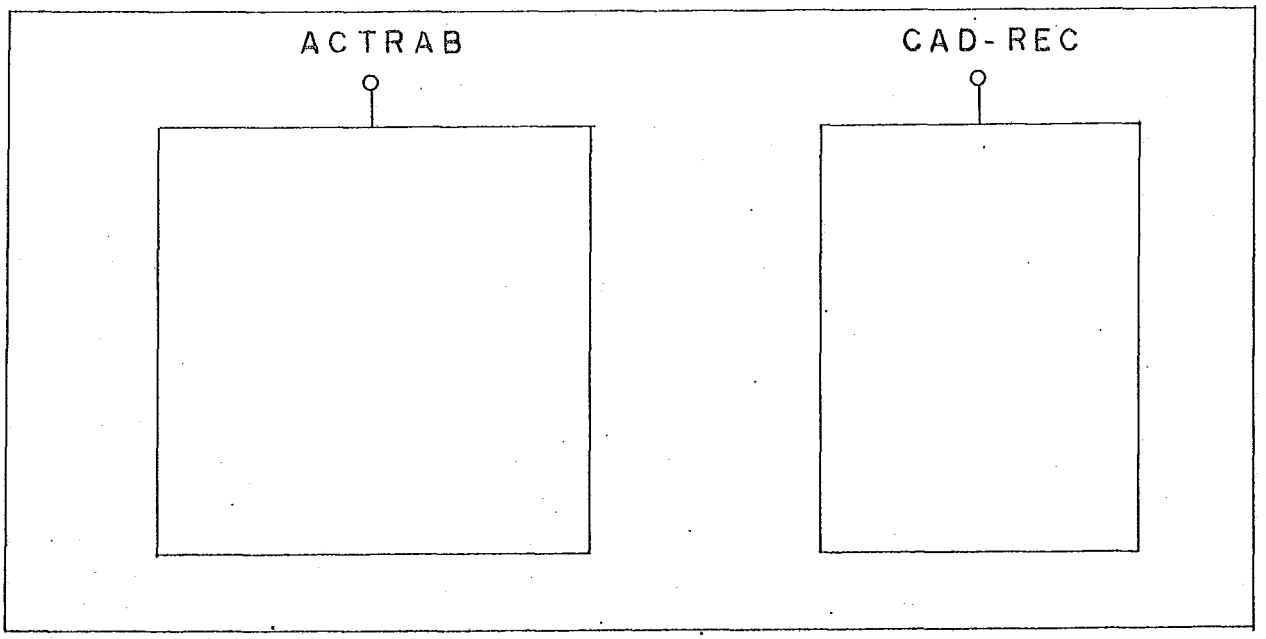



Figura 2.15. - Representação Gráfica de um Acervo Setorial (ACSET).

- Acervo Total

O acervo total MICROLOBAN é o que normalmente é referenciado como Base de Dados. Ele é composto de acervos setoriais conhecidos do sistema por seus nomes. Um acervo setorial corresponde a uma aplicação ou sistema de informação. Em MICROLOBAN além de não haver comunicação entre os acervos setoriais, apenas um destes pode estar em uso a cada momento ainda que somente para leitura. A designação padrão para acervo total é ACTOT.

Devido as restrições semânticas feitas em MICROLOBAN somente alguns desses pretipos podem ser armazenados e recuperados no Canal Auxiliar, sendo eles: itens tuplas, ligações, tabelas relacionais, tabelas ligacionais e coleção de itens.

A representação interna para os pretipos MICROLOBAN são definidos e discutidos detalhadamente em DANTAS (14).

Existem construções suportadas por LOBAN que não foram preservadas na definição do subconjunto, são elas: sigla, numérico natural, numérico decimal, poligação, hiperligação, relação de ordem e registro. ((SANTOS (27)).

## 2.5. Instruções MICROLOBAN

Após terem sido fornecidas as estruturas de dados possíveis de serem criadas no acervo setorial, serão definidos a seguir as instruções que executam as funções citadas na seção 2.2., expondo algumas das restrições feitas em LOBAN para a definição da linguagem MICROLOBAN.

### 2.5.1. Instruções de Gerência

As instruções que executam as funções de gerência são as que criam e abolem acervos setoriais e arquivos (<criar acset>, <abolir acset>, <criar arquivo>, <abolir arquivo>).

Para poder fornecer um comando de gerência o usuário deve ter direito de acesso para gerenciar os pontos a serem endereçados, além dos mesmos estarem protegidos para alteração.

#### Instrução <criar acset>

Como já foi dito anteriormente, o acervo total, conteúdo da base de dados, é composta de um ou mais acervos setoriais com nomes diferentes, sendo assim uma nominação sobre acervos setoriais. Através da instrução <criar acset> , o usuário pode requisitar a criação de um acervo setorial inse-

rindo um par (nome, acervo setorial) no acervo total.

Existem duas formas de criaçao de um acervo setorial. Uma é criar um acervo pela primeira vez, e neste caso o sistema automaticamente insere na folha o verbete de acesso "embutido" com autorização total sobre o acervo, possibilitando o acesso a tal construção. A outra maneira é a criação a partir de outro acervo setorial já existente, sendo incluidos na folha os verbetes de acesso além dos outros verbetes que constam do acervo de reconstrução.

Instrução <abolir acset>

Esta instrução é utilizada para retirar um par (nome, acervo setorial) do acervo total, ou seja, excluir uma construção de pretipo ACSET componente imediato do ACTOT.

Instrução <criar arquivo>

O usuário tem a possibilidade de criar um arquivo permanente cujo nome, fornecido pelo usuário, corresponderá a identificação do tipo de arquivo que deverá ter a sua especificação incluida anteriormente na folha. As restrições aplicadas a criação de arquivos dizem respeito a impossibilidade de se criar arquivos temporários, a obrigatoriedade do nome do arquivo sempre identificar também o tipo, e a inexistência de relação de ordem associada a tabela relacional ou tabela ligacional que compõe o arquivo.

Quando se cria um arquivo o componente imediato TR/TL aparece com a construção vazia, enquanto a FICHA por ser componente obrigatório pré-definido pelo sistema terá a informação da data de criação do arquivo automaticamente incluida.

Instrução <abolir arquivo>

Tem a finalidade de excluir a ocorrência de um arquivo, identificado pelo nome de arquivo fornecido pelo usuário, do acervo de trabalho.

2.5.2. Instruções de Manipulação (Operação)

São as instruções que transformam (alteram) o conteúdo do acervo (base de dados) ou da construção auxiliar. Desses conjuntos de instruções não são consideradas as que recuperam os acervos. Essas instruções executam funções de inclusão, exclusão e substituição de construções.

A alteração do conteúdo da folha também será realizada utilizando-se as instruções de operação.

Instrução <incluir construção>

Dependendo do tipo da construção fornecida podemos incluir uma tupla ou conjunto de tuplas em uma tabela relacional, ou uma ligação ou conjunto de ligações em uma tabela ligacional no acervo setorial.

Quando aplicada à construção auxiliar, ela permite incluir nominações, itens, coleção de itens e tabelas, desde que a essas construções sejam associados nomes e que as mesmas e seus componentes possam ser identificados e endereçados posteriormente.

Observações importantes devem ser feitas quanto ao fato de que a construção endereçada e a construção fornecida devem possuir tipos compatíveis, além de que o ponto endereçado deve ser único.

Instrução <excluir construção>

Exclui uma construção no ponto endereçado. Se a construção endereçada pertencer ao acervo seu pretipo terá que ser tupla (TUP) ou ligação (LIG), caso contrário a exclusão não será realizada.

Quando a construção endereçada faz parte da construção auxiliar, ela poderá ser excluída independente de seu pretipo.

Instrução <substituir construção>

A instrução <substituir construção> aplica-se tanto a construções sob nomes (componentes imediatos de nominações), quanto as construções sem nome (componentes imediatos de uma coleção), dependendo do contexto considerado.

Caso o contexto de referência mais externo seja o acervo de trabalho, só será permitido a substituição do componente imediato de uma nominação (tupla, ligação, ligante, tabela relacional e tabela ligacional). Deve ser observado ainda, que as construções em questão devem ser de tipos compativeis.

Se tivermos como contexto de referência a construção auxiliar, será permitido a substituição de construções componentes imediatos de uma nominação (tupla, ligação, ligante, tabela relacional, tabela ligacional), ou de uma coleção (coleção de itens). Da mesma forma que no caso do acervo de trabalho, as construções serão obrigadas a possuirem tipos compativeis.

### 2.5.3. Instruções de Controle de Fluxo

Permitem a execução repetitiva (comando iterativo) ou seletiva (comando condicional) de um bloco de instruções autonomas.

Restrições foram feitas quanto as instruções que compõem o bloco de instruções utilizado nos comandos <fazer para> e <fazer se>. Só será permitido o uso das instruções de manipulação, de marcação, de controle de fluxo e de definição de transação.

#### Instrução Iterativa <fazer para>

Para especificar a execução repetitiva de uma série de instruções, o usuário usará a instrução <fazer para> que executará o bloco de instruções autônomas para cada ponto de um conjunto de pontos endereçados e/ou para cada campo de um conjunto de campos endereçados. Outra função dessa instrução é atualizar o Ponto Corrente (PC) ou Campo Corrente (CC) do conjunto controlador. Será permitido qualquer nível de embutimento de instruções repetitivas.

O comando iterativo permite também a especificação de uma ordenação temporária sobre o conjunto de pontos a serem processados. (Maiores detalhes ver SANTOS (27), Capítulo 12)

#### Instrução Seletiva <fazer se>

A instrução <fazer se> permite definir a execução de um determinado bloco de instruções autônomas dependendo se o resultado da avaliação do critério fornecido for verdadeiro ou não. Da mesma forma como no comando iterativo, não há restrições quanto ao nível de embutimento de comandos seletivos.

Esta instrução tem semântica semelhante ao comando condicional das linguagens de programação.

### 2.5.4. Instruções de Alocação de Recursos

As instruções que alocam recursos para execução de uma seção MICROLOBAN são as que informam o acervo setorial a ser utilizado (<criar acset> ou <abrir acset>) e os volumes de entrada, saída e interação (<abrir volume>), e a que estabelece a proteção necessária para acessar a base de dados (<estabelecer proteção>).

A instrução de <criar acset> já foi devidamente explicada na seção 2.5.1 quando foram descritas as instruções de gerência.

#### Instrução <abrir acset>

Tem a função de informar e tornar disponível o acervo de setorial utilizado na sessão MICROLOBAN. É uma instrução obrigatória caso não seja fornecido nenhum comando para criar um determinado acervo setorial.

Quando informamos o acervo desejado, devemos também fornecer as senhas de acessos que deverão identificar os tipos de acessos permitidos ao usuário, cujas descrições se encontrarão na folha do ACSET. Caso as senhas de acesso não sejam fornecidas, será dado autorização total ao usuário de acordo com os direitos de acesso pré-estabelecidos.

Instrução <abrir volume>

Esta instrução tem como finalidade principal fornecer quais os volumes a serem usados durante uma execução, além de informar o tipo de uso, isto é, entrada, saída ou interação.

Não é uma instrução obrigatória, desde que não seja efetuado entrada/saída de dados.

Instrução <estabelecer proteção>

Mesmo sendo um sistema que a principio só permite acesso único à base de dados, MICROLOBAN prevê futuras extensões onde acesso múltiplo ao mesmo acervo setorial será autorizado.

Para manter a característica de que qualquer operação só será realizada pelo usuário caso seja precedida ou esteja sob a garantia de uma proteção, MICROLOBAN possui uma instrução que estabelece a proteção sobre componentes imediatos do acervo setorial.. Além de informar os pontos a serem protegidos, também deverá ser indicado o tipo de proteção a ser realizado (Ler ou Alterar).

O Sistema realizará o mesmo tipo de proteção para os pontos que mantém conexão com os pontos especificados.

Como se pode observar a instrução <estabelecer proteção> é única e obrigatória.

## 2.5.5. Instruções de Liberação de Recursos

São as instruções que liberarão os recursos alocados para uma execução MICROLOBAN. Caso alguns desses comandos

não sejam fornecidos a instrução de fim <encerrar> terá a função de liberar recursos, caso o Sistema não tenha recebido ordem para tal.

Instrução <abandonar proteção>

Como a instrução <estabelecer proteção> determina um universo da base de dados que o usuário pode acessar para ler ou alterar, a instrução <abandonar proteção> informa ao sistema que as áreas protegidas anteriormente não serão mais acessadas dentro de uma mesma sessão.

Instrução <fechar volume>

A instrução <fechar volume> encerra a ligação do SBD (Sistema de Banco de Dados) com um determinado volume cujo tipo (Entrada, Saída e Interação) e identificação foram fornecidos no início de uma execução. (Alocação de recursos)

Instrução <fechar acset>

Essa instrução libera o acervo setorial aberto ou criado por um comando <abrir acset> ou <criar acset> respectivamente, acabando com todas as proteções ativas para esse acset.

## 2.5.6. Instruções de Navegação Explícita

Navegar significa marcar e varrer um conjunto de pontos. Em MICROLOBAN só usaremos a navegação denominada explícita onde seu início e término é identificado por uma instrução de início e uma instrução de fim de marcação respectivamente. Esses comandos de marcação definem "snapshots" e não vis-

tas como é o caso de LOBAN.

Instrução <fixar marca>

A definição de um "snapshots" é determinada por uma instrução <fixar marca>. Esse comando fornecerá um identificador (nome) da marca e os pontos a serem considerados na marcação.

O pretipo das construções endereçadas é restrita a tupla (TUP), ligação (LIG) e item (DATA, HORA, INT e REAL).

Instrução <tirar marca>

A instrução <tirar marca> possui duas funções bem definidas. Uma é retirar a(s) marca(s) fornecida(s) independente dos pontos sobre os quais elas foram estabelecidas. Outra função é retirar uma ou várias marcas de um determinado ponto.

## 2.5.7. Instruções de Reconstrução

Reconstrução é um processo de criação de um acervo setorial a partir de um outro acervo setorial antigo ("duplicata") ou atual.

Em MICROLOBAN teremos dois níveis de reconstrução: reconstrução de sessão (tipo regressiva); e reconstrução de comandos dentro de uma sessão (tanto progressiva quanto regressiva). Para os dois tipos de reconstrução existirá um histórico contendo as informações dos estados passados da base de dados denominado de "Cadeia de Reconstrução".

Um conhecedor da interface LOBAN, observará a não existência do tipo de reconstrução de comandos dentro de uma ses

são. Esta se fez necessária para que o usuário comum pudesse desfazer parte das alterações executadas por ele dentro de uma execução.

Além das instruções de reconstrução de sessão (<criar acset>) e de reconstrução de comandos dentro de uma sessão (<reconstruir comandos>), temos as instruções que desprezam a cadeia de reconstrução (<abandonar acrec>) e a que reinicia a gravação da mesma cadeia (<arquivar actrab>).

Instrução <criar acset>

Além de criar um novo acervo setorial, o comando <criar acset> tem a função de reconstruir a base de dados até um estado (número de versão) desejado. Este tipo de reconstrução é chamado de regressiva.

Para este tipo de função será necessário informar o nome do acervo a ser reconstruido e a versão desejada do mesmo.

Instrução <reconstruir comandos>

Os tipos de reconstrução realizados pela instrução <reconstruir comandos> são chamadas regressiva e progressiva. (Maiores detalhes ver SANTOS (27), Capítulo 21; e DANTAS (14))

Para execução da reconstrução somente será necessário a especificação do tipo da reconstrução (EXECUTANDO → progressiva, DESFAZENDO → regressiva), e de quantos comandos a reconstruir.

Instrução <abandonar acrec>

A instrução <abandonar acrec> determina a paralização da geração da cadeia de reconstrução (Fita log) até que seja fornecido uma instrução <arquivar actrab>. A cadeia de reconstrução gerada até o instante da execução deste comando (<abandonar acrec>) será desconsiderada.

Instrução <arquivar actrab>

Esta instrução, quando fornecida após o comando <abandonar acrec>, tem a função de reinicializar a gravação da cadeia de reconstrução. Caso contrário esta instrução determinará um novo estado a partir do qual serão consideradas as alterações, isto é, início de gravação de uma nova Cadeia de Reconstrução.

## 2.5.8. Instrução de Saída (<representar>)

No que diz respeito a saída de dados o subconjunto restringiu bastante o universo considerado preferindo concentrar-se nas funções básicas de um SGBD.

Desta forma a saída de dados se resume em uma única máscara de saída e várias regras de representação colocadas a disposição do usuário pelo sistema.

## 2.5.9. Instrução de Definição de Transação (<bloco de transação>)

A definição de uma transação engloba um conjunto de

comandos cuja execução será considerada como uma unidade de processamento. Restrição quanto ao número de percursos foram feitas na definição do subconjunto, só sendo permitidas transações de único percurso.

O usuário pode fornecer tratamentos a serem realizados quando da ocorrência de falhas de execução ou deixar o tratamento de erro a cargo do Sistema.

As instruções que podem ser especificadas na lista de instruções a serem consideradas na transação, são: Instruções de Manipulação, Instruções de Marcação e Instruções de Controle de Fluxo. Como se deve observar não será permitida a definição de transações encaixadas (embutimento de definição de transação).

### 2.5.10. Expressões Utilizadas pela Linguagem MICROLOBAN

Existem expressões que são utilizadas tanto pelas instruções autônomas quanto pelas instruções intermediárias (descritas posteriormente). São elas: <endereço de campo>, <endereço de ponto>, <obter valor booleano>, <obter construção>, <termo de conjunto de pontos>, <termo de controle>, <verbete>. A seguir serão descritas tais expressões de maneira detalhada, principalmente quanto aos seus aspectos semânticos.

2.5.10.1. Expressão <endereço de campo>

O conceito de campo inclui o conceito de "ponto de um volume", ou seja, não existe o campo separado de seu ponto físico. Um campo pode ser subdividido em subcampo segundo uma máscara sobreposta a ele. Em MICROLOBAN temos uma única máscara pré-definida que será sobreposta a um determinado campo, não sendo permitido assim definição de máscaras através de expressões de divisão favorecidas pelo usuário. (Ver SANTOS (27), Capítulos 8 e 23).

O usuário, através das expressões denominadas endereço de campo, pode especificar um acesso a uma inscrição e ao seu campo seja para efeitos de "ler" ou "escrever". A idéia básica é a partir de um campo já designado buscar um campo imediatamente embaixo do campo de referência. (Maiores detalhes ver SANTOS (27), Capítulo 11).

Devido as restrições feitas para o subconjunto (uma única máscara), o endereço de campo será praticamente pré-definido constando de um identificador de campo (<id campo>).

O <id campo> define o campo de referência utilizado na avaliação do endereço de campo. Temos assim dois campos de referência a serem considerados. Um deles determina que o <end campo> será avaliado com relação ao Campo Corrente, permitindo endereçar o campo ultimamente demarcado no conjunto controlador de campos (Ver instruções de Controle de Fluxo). Se existirem dois conjuntos controladores de campos para uma mesma instrução <fazer para> , o CC não estará definido. No caso de laços encaixados o conjunto controlador considerado é sempre aquele ligado ao bloco de instruções da instrução iterativa ao qual per-

tence a instrução contendo o endereço de campo CC.

Podemos resolver o problema da ambiguidade no caso de <fazer para> encaixados bem como no caso de dois conjuntos controladores de campos numa mesma instrução interativa, bastando para isso estabelecer uma marca sobre o conjunto controlador desejado qualificando o CC.

Outro identificador de campo válido é o que determina que o endereço de campo será avaliado com relação ao volume aberto com nome <id volume> na alocação de recursos para execução.

2.5.10.2.Expressão <endereço de ponto>

O conceito de ponto permite amarrar uma construção ao seu contexto (construção abrangente). O endereçamento de ponto segue também um esquema "de cima para baixo". A idéia básica consiste de a partir de um ponto de referência buscar um ponto imediatamente embaixo do ponto designado (chamado ponto de distância 1 sob o ponto de referência).

Chamamos de "critério para determinar um ponto de componente imediato" ao critério aplicado para encontrar um ponto imediatamente embaixo do ponto de referência. Assim sendo, o endereço de ponto é composto de uma seqüência de tais critérios. A avaliação de endereço obedece ao seguinte processo:

- Determinar o ponto de referência a ser considerado na avaliação (construção de referência).

- Em cada nível, marcar os componentes imediatos como "pontos candidatos".

- Varrer o conjunto de pontos candidatos tornando cada um dos pontos "ponto examinando", e avaliando-o segundo o critério fornecido. Se o resultado da avaliação for "verdadeiro" o ponto examinando passa a ser "ponto escolhido", caso contrário o ponto será rejeitado e não mais considerado.

- Os dois últimos passos se repetirão para todos os níveis aos quais estarão associados critérios.

Como resultado desta avaliação teremos um conjunto de pontos endereçados podendo este conjunto ser vazio, indeterminado, unitário ou conter vários pontos endereçáveis.

A expressão <end ponto> contém um identificador de ponto(<id ponto>) que define o "ponto de referência" a partir do qual será avaliado o endereço de ponto, podendo o mesmo ser o acervo de trabalho (ACTRAB), a Construção Auxiliar (AUX), o Ponto Corrente (PC), o Ponto Examinando (PX) ou uma marca estabelecida sobre um conjunto de pontos através da instrução <fixar marca> (: <lit marca>).

Para eliminar ambiguidade no caso de endereços de ponto encaixados, podemos marcar os pontos candidatos a serem examinados.

Nenhuma restrição é feita quanto ao nível de embutimento do endereço de ponto. Maiores detalhes sobre o funcionamento e sintaxe ver SANTOS (27), Capítulo 10.

2.5.10.3. Expressão <obter valor booleano>

Esta expressão caracteriza-se pelo fato de seu resultado só poder assumir dois estados, ou seja, verdadeiro ou falso. Dessa forma as operações possíveis de serem realizadas são aquelas que usam como operandos valores booleanos. Dentre estas operações temos a conjunção (E), a disjunção (OU), a disjunção exclusiva (OUEX) e a implicação (IMPL) dos valores booleanos obtidos.

Além dos operadores booleanos, temos os qualificadores universais PARA TODO e EXIST, os operadores que avaliarão a posição relativa das construções em relação a uma ordem à revelia ligada ao pretipo das construções obtidas (= , <, >, <=, >=, <>), e o operador de pertinência (ELEM). O uso de parênteses é permitido para fins de determinação de prioridade de execução dos operadores booleanos.

2.5.10.4. Expressão <termo de contrôle>

A expressão <termo de contrôle> gera um conjunto controlador de pontos ou um conjunto controlador de campos utilizados na execução de uma instrução iterativa <fazer para>.

Quando se tratando de conjunto de pontos, podemos fornecer a especificação de uma ordenação temporária sobre tal conjunto, indicando para isto os atributos a serem ordenados e o tipo de ordenação (ASC ou DESC) a ser feita associada a cada atributo.

Independente do tipo do conjunto controlador gerado como resultado da avaliação de um <termo de controle>, o

usuário pode associar um nome de marca a tal conjunto para referência posterior dentro de um bloco de instruções autônomas que compõe a instrução <fazer para>.

2.5.10.5. Expressão <obter construção>

A expressão <obter construção> é uma instrução intermediária cuja execução resulta na obtenção de uma construção denominada de "construção obtida" ou "construção intermediária". Esta instrução permite pedir um "transporte" de informação entre a base de dados, o canal auxiliar e os periféricos, fazendo com que a informação primeiro entre em uma "zona intermediária" onde serão feitas operações do tipo "avaliação da expressão". O resultado desta avaliação estará disponível para alterações na base de dados ou canal auxiliar, ou para a saída. A zona intermediária (ZI) não serve para armazenar temporariamente as construções obtidas, fazendo-se necessário a subsequente transferência das mesmas para um determinado canal.

As instruções intermediárias disponíveis em MICROLOBAN são divididas em grupos que caracterizam-se por possuirem funções distintas, tais como:

- Entrada de dados

MICROLOBAN é uma linguagem que não possui uma instrução autônoma que execute a função de entrada de dados, para isto ela utiliza uma instrução intermediária que especifica a construção a ser obtida pela interpretação da instrução fornecida (INTR).

A entrada de dados será realizada utilizando-se regras de interpretação a qual deverá identificar o tipo da construção a ser obtida na Z.I.

- Operações aritméticas

Operações aritméticas, como a adição, a subtração, a multiplicação, a divisão e a potencialização, podem ser realizadas sobre construções de pretipo inteiro, real, data e hora. Para os dois últimos pretipos relacionados só serão permitidas as operações de soma e subtração.

Microloban permite ainda a especificação de prioridades fornecidas através de parênteses.

- Operações da Álgebra Relacional

Operadores da Álgebra Relacional são disponíveis no subconjunto, executando as operações de união (UNI), de interseção (INTER), de diferença (DIFE), de seleção (C), de projeção (ESTREIT) e de junção (JUNT). A seguir serão descritos os operadores levando em consideração algumas restrições semânticas aplicadas sobre eles.

UNI, INTER, DIFE:

As relações (coleções) operandos, utilizadas na união,interseção e diferença, devem ser união-compatíveis,isto é, devem ter o mesmo grau u e os atributos das duas relações serem construidos a partir do mesmo domínio.

Os elementos da coleção obtida serão do mesmo pretipo dos elementos das coleções operandos. Esses operadores tem sentido equivalente ao da união, diferença e interseção na teoria dos conjuntos. Prioridade de execução também pode ser

estabelecida através do uso de parênteses.

C :

Este operador obtém uma duplicata da construção no ponto endereçado que deve ser único. O critério de seleção será fornecido através da expressão de endereçamento de ponto.

ESTREIT:

Sua função é equivalente a da projeção. Seus operandos são a tabela relacional, obtida pela avaliação de uma expressão <obter construção>,e a lista de atributos a serem despresados na tarel considerada.

A tabela relacional resultante desta operação poderia ter a sua cardinalidade reduzida caso a lista de atributos contenha a chave primária da relação operando.

JUNT:

Especifica uma tabela relacional a ser obtida a partir de duas tabelas relacionais ("equi-join").

A tupla da tabela obtida, resultado da operação de junção, será composta dos componentes imediatos das tuplas das tabelas operandos quando o critério de junção for satisfeito. Além disso, o usuário pode especificar se os atributos da segunda tabela são para serem excluidos da tabela obtida. Caso o critério de junção não seja satisfeito para as tuplas operandos, a tabela resultante será a construção vazia.

-. Operação do Cálculo Relacional

São operações que têm como resultado valores Booleanos Verdadeiro ou Falso. Os operadores do Cálculo Relacional são: a negação (NAO), a conjunção (E), a disjunção (OU), a disjunção exclusiva (OUEX), a implicação (IMPL), os qualificadores universais (PARA TODO e EXIST), a pertinência (ELEM), e os operadores de comparação (=,< , >, <=, >=,<>). Todos os operadores relacionados são utilizados na obtenção de um valor booleano (<obter valor booleano>).

NAO, E, OU, OUEX, IMPL:

Esses operadores usam como operandos valores booleanos. O operador NÃO é o único que é aplicado a um único valor booleano (operação unária).

PARA TODO (∀)

Especifica o valor booleano VERDADEIRO se para todos os pontos endereçados o critério fornecido, em <obter valor booleano> for VERDADEIRO. Caso contrário será especificado o valor booleano FALSO.

Esta operação implica em uma varredura sobre os pontos endereçados ficando definido o Ponto Corrente (PC).

EXIST (∃):

O resultado deste operador será o valor booleano VERDADEIRO se para algum dos pontos endereçados o critério fornecido através do <obter valor booleano>, torna-se VERDADEIRO. Caso contrário, nenhum dos pontos endereçados tornar VERDADEIRO o <obter valor booleano>, o resultado será o valor booleano

FALSO. A instrução intermediária EXIST implica também numa varredura sobre os pontos endereçados pelo qual fica definido o PC (Ponto Corrente).

Na ausência da opção <obter valor booleano> o valor booleano resultante da instrução EXIST será sempre VERDADEIRO se existir pelo menos um ponto endereçado. Neste caso a expressão <end ponto> conterá o critério a ser aplicado sobre os pontos (<obter valor booleano>).

ELEM:

Essa instrução intermediária terá como resultado o valor booleano VERDADEIRO, caso a construção obtida (operador C) for elemento do conjunto de construções definido pela expressão <termo de conjunto de construções>. Os tipos das construções obtidas têm que ser compatíveis para que a operação de pertinência seja executada satisfatoriamente.

- Operadores de comparação (=,<,>,<=,>=,<>):

O resultado, obtido pela aplicação dos operadores de comparação (=,>,<,<=,>=,<>), será o valor booleano VERDADEIRO caso as construções obtidas se encontrarem na posição relativa indicada. Podem ser fornecido uma ordem explicitamente ou será considerado uma ordem implícita dependendo do pretipo das construções obtidas.

O operador de comparação de igualdade será também utilizado na comparação de nomes de construções.

- Operações sobre Tabelas

Existem instruções intermediárias que geram como resultado uma tabela ligacional ou relacional. Esses operadores executam operações tais como: transformação de uma tabela relacional em ligacional (AGRUP), transformação de uma tabela ligacional em relacional (DESAGRUP), obtenção de uma tabela ligacional a partir de duas tabelas relacionais (LIGA), renomeação dos atributos de uma tabela relacional (RENOM) e acréscimo de atributos em uma tabela relacional (ALARG).

AGRUP:

Especifica uma tabela ligacional a ser obtida pelo agrupamento de uma tabela relacional segundo os atributos especificados. O primeiro agrupamento realizar-se-á em cima da tabela relacional obtida mediante a avaliação <obter tarel> formando subconjuntos de tuplas. Esses subconjuntos estão definidos pelo concordância dos valores nas colunas indicadas. A partir de cada subconjunto será formada uma ligação cujo ligante contém os valores comuns sob os atributos originais, e cuja tabela ligada é obtida pelo estreitamento do subconjunto dos atributos do agrupamento.

DESAGRUP:

Tem função contrária a instrução intermediária AGRUP, ou seja, tem como resultado uma tabela relacional obtida pelo desagrupamento de todas as ligações da tabela ligacional obtida mediante a instrução <obter talig>.

O processo de desagrupar consiste de:

- Formar um subconjunto de cada ligação, juntando o con-

teúdo de cada tupla ligado com o conteúdo de seu ligante, e

- Unir os subconjuntos assim formados obtendo a tabela relacional especificada.

A cardinalidade da tabela relacional resultante será igual a soma das cardinalidades de todas as tabelas de ligados.

LIGA:

Especifica uma tabela ligacional a ser obtida a partir de duas tabelas relacionais.

Para cada tupla da primeira tabela relacional obtida, será construída uma ligação cuja tabela ligada será formada a partir das tuplas da outra tabela relacional obtida.

Uma tupla da segunda tabela entrará em todas as ligações que tem os atributos especificados do ligante concordando com os valores dos seus respectivos atributos. O usuário pode explicitar o desejo de excluir os atributos especificados referente a tabela relacional que comporá a tabela de ligados.

A tabela de ligados pode ser a construção vazia desde que não seja encontrada nenhuma tupla concordante na segunda tabela.

A cardinalidade da tabela ligacional obtida será igual a cardinalidade da primeira tabela.

RENOM:

É um operador que renomeia os nomes dos atributos de uma tabela relacional. A estrutura original da tabela relacional obtida deverá ser mantida após o processo de renomeação.

Uma observação a ser feita é que caso a tabela pos-

sua atributos compostos, a renomeação não poderá ser feita a nível de nomes de componentes desse atributo.

ALARG:

Especifica uma tabela relacional a ser obtida pelo acréscimo de uma ou mais colunas (atributo) a tabela obtida pela expressão operando <obter tarel>. Os atributos a serem acrescidos terão o nome fornecido e seu tipo será deduzido da construção obtida através da avaliação da expressão operando <obter construção>. Cada nova coluna será inicializada pelo Sistema com a construção vazia.

- Outras operações intermediárias

A linguagem MICROLOBAN fornece outras instruções que têm funções específicas, tais como: obtenção de um literal, obtenção do nome de uma construção (N), obtenção de uma coleção (COLEC), obtenção da contagem dos pontos endereçados (CONT), obtenção de nome da marca (M), obtenção da cardinalidade de uma tabela (CARD), execução de procedimentos padrões de cálculos disponíveis na instalação (CALC), obtenção da construção vazia (VAZ), da hora do sistema (HORA-CORR) e da data do sistema (DATA-CORR), e definição da consistência, da autorização, do direito de acesso e de procedimento fonte (VERBETE).

Literal:

Especifica a obtenção de uma construção pela interpretação de uma inscrição encontrada no próprio texto fonte MICROLOBAN.

Um literal é uma seqüência de caracteres que serão

interpretados segundo o contexto. Vários tipos de literais podem ser fornecidos, entre eles temos os literais numéricos, inteiros e real, literal data, literal hora e o literal que representa uma numeração de caracteres.

N:

Este operador tem como resultado uma construção a ser obtida pela duplicata do nome no ponto endereçado.

O ponto endereçado deve ser único e deve acessar construções que são componentes imediatas de uma nominação (possuidoras de nomes).

COLEC:

A instrução intermediária COLEC obtém uma coleção de duplicatas das construções nos pontos endereçados. Vale ressaltar que os componentes de uma coleção devem ser construções que pertençam ao mesmo pretipo.

CONT:

Especifica em número natural obtido pela contagem dos pontos endereçados. Caso não seja endereçado nenhum ponto, a construção obtida será o número zero.

M:

Este operador obtém na Z.I. uma coleção de marcas fixadas pelo usuário no ponto endereçado. Caso o ponto endereçado não seja único é obtido o conjunto de marcas nos pontos endereçados.

CARD:

O resultado obtido pela aplicação deste operador é um número natural significando a cardinalidade da coleção obtida. A construção obtida com o operando <obter coleção> deve ser uma coleção.

CALC:

Especifica um número obtido pelo cálculo designado executado em cima das construções encontradas nos pontos endereçados. Microloban permite a execução dos seguintes cálculos pré-definidos: TOTAL, MEDIA, MAX, MIN e DESVIO. Vale a pena observar que as construções endereçadas devem ter tipos compatíveis com o cálculo a ser realizado.

VAZ:

Especifica como construção a ser obtida a construção vazia. Esta instrução intermediária terá aplicação somente quando deseja-se obter este tipo de construção como componente imediato da construção auxiliar (AUX).

HORA-CORR

A construção a ser obtida na Z.I. pela aplicação dessa instrução, é uma ocorrência do pretipo HORAS com os valores de horas, minutos e segundos correspondentes à hora da execução da instrução HORA-CORR (valores fornecidos pelo sistema).

DATA-CORR:

Especifica como construção a ser obtida uma ocorrência do pretipo DATA com os valores do dia, mês e ano correspondentes à data corrente na hora da execução dessa instrução (valores fornecidos pelo sistema).

VERBETE:

Através dessa instrução o usuário pode fornecer expressões que definem o conteúdo da base de dados (acervo) em termos de coerência, de autorização, de direitos de acesso e de texto fonte. Essas expressões, denominadas verbetes, serão descritas posteriormente.

2.5.10.6. Expressão <termo de conjunto de construções>

A avaliação de uma expressão <termo de conjuntos de construções> tem como resultado um conjunto de construção de mesmo tipo gerado na Z.I.

O conjunto de construções pode ser dos tipos relacionados abaixo:

- Conjunto de todas as ocorrências do tipo designado (<d tipo>).

- Conjunto de todas as ocorrências do pretipo designado (<d pretipo>).

- Coleção obtida através da avaliação da expressão <obter coleção> .

- Conjunto de construções por enumeração podendo ser de literais ou de verbetes (mesmo tipo).

2.5.10.7.Expressão <verbete>

Através dessa expressão o usuário MICROLOBAN fornece a definição da coerência da base de dados, da autorização de usuários, da regulamentação de acesso conforme a autorização ou de procedimentos fontes, a serem armazenados no acervo (base de dados).

Um verbete de usuário é utilizado para fornecimento da identificação e senha de um usuário além da regulamentação de acesso para tal autorização .O uso de um acervo setorial é restrito aos usuários cuja autorização está descrita na folha.

Um verbete de acesso descreve os tipos de acessos possíveis a um determinado acervo, ou seja, a especificação de tal verbete informa os pontos cujos acessos serão autorizados a um determinado usuário.

A definição de procedimentos fontes pode ser feita através de um verbete fonte, sendo necessário associar um nome a um determinado texto além da descrição do mesmo.

A coerência de um acervo setorial é descrita através dos verbetes de coerência. Esssa descrição é composta de todas as definições de tipos de dados permissíveis na base de dados desde o nível mais abrangente (acervo de trabalho) até o nível elementar (item).

## 2.6. Arquitetura Geral do Sistema MICROLOBAN

Após a especificação do subconjunto em termos de estruturas de informação e operações a serem consideradas sobre estas, foi iniciado o trabalho de definição de uma Arquitetura

de Multinível para o sistema MICROLOBAN (HUTT (19), MYLOPOULOS (22)).

De uma forma bem abrangente esta arquitetura é composta de três níveis bem distintos. O nível mais interno diz respeito a realização das estruturas de informação e as primitivas de acesso a essas estruturas, bem como o gerenciamento de memória secundária e primária. O nível intermediário tem como função principal a interpretação da forma intermediária dos comandos DML e DDL, isto faz com que tenhamos um alto grau de independência do nível físico, ou seja, qualquer mudança no nível interno provavelmente não afetará os demais níveis. Finalmente temos o nível superior que está intimamente ligado com a linguagem MICROLOBAN proposta no início desse capítulo.

Um detalhamento da arquitetura proposta foi necessário para definição dos módulos componentes de forma a identificar as interfaces internas utilizadas a cada nível, bem como a especificação das funções de cada um destes módulos. Durante esta etapa o subconjunto foi validado em termos de sua construção sintática e de suas características semânticas resultando em algumas alterações (sintáticas e semânticas) na especificação da linguagem. Devido a este fato existem certas diferenças sintáticas entre comandos da linguagem LOBAN e os correspondentes em MICROLOBAN.

Como resultado da fase de detalhamento da arquitetura proposta chegou-se a um funciograma geral onde seus componentes, estações e canais, retratam os componentes do sistema e a interligação entre eles. A figura 2.16 ilustra a Arquitetura Geral detalhada do Sistema MICROLOBAN.

A comunicação entre o Usuário e o Sistema de Banco de Dados é estabelecida através dos Canais Primários de Entra-

da/Saída. Por meio do Canal Primário de Entrada o sistema recebe os comandos fonte fornecidos pelo usuário. Como retorno o usuário recebe as mensagens operacionais resultantes da execução de tais comandos pelo sistema através do Canal Primário de Saída.

A estação ANALISADOR é composta internamente do analisador Léxico , do analisador Sintático e do analisador Semântico, sendo também a responsável pelo recebimento de todos os comandos fornecidos pelo usuário MICROLOBAN utilizando para isto o Canal Primário de Entrada.

O Analisador Léxico é um tradutor cuja entrada é uma seqüência de símbolos representando o programa fonte e cuja saída é uma seqüência de entidades sintáticas primitivas. Sua função principal é grupar seqüências de caracteres terminais nas entidades referenciadas anteriormente. Ele utiliza a tecnologia usual de automatos finitos para localização e tabelas de espelhamento ("hashing") para reconhecimento de palavras reservadas.

O analisador Sintático utiliza o método RRP LL(1) (SIMONE (28)) com esquema de recuperação de erros por eliminação de frase. Em particular quando o modo de operação é interativo, o sistema elimina o comando corrente obrigando sua resubmissão a partir do ponto de erro; quando em modo lote, o recuperador trabalha por eliminação de frases ("panic mode"). A escolha deste método de análise deveu-se principalmente as suas excelentes características de ecônomia de espaço, restrição fundamental da implementação. Maiores detalhes quanto ao processo de análise léxica ver CRIVOROT (12).

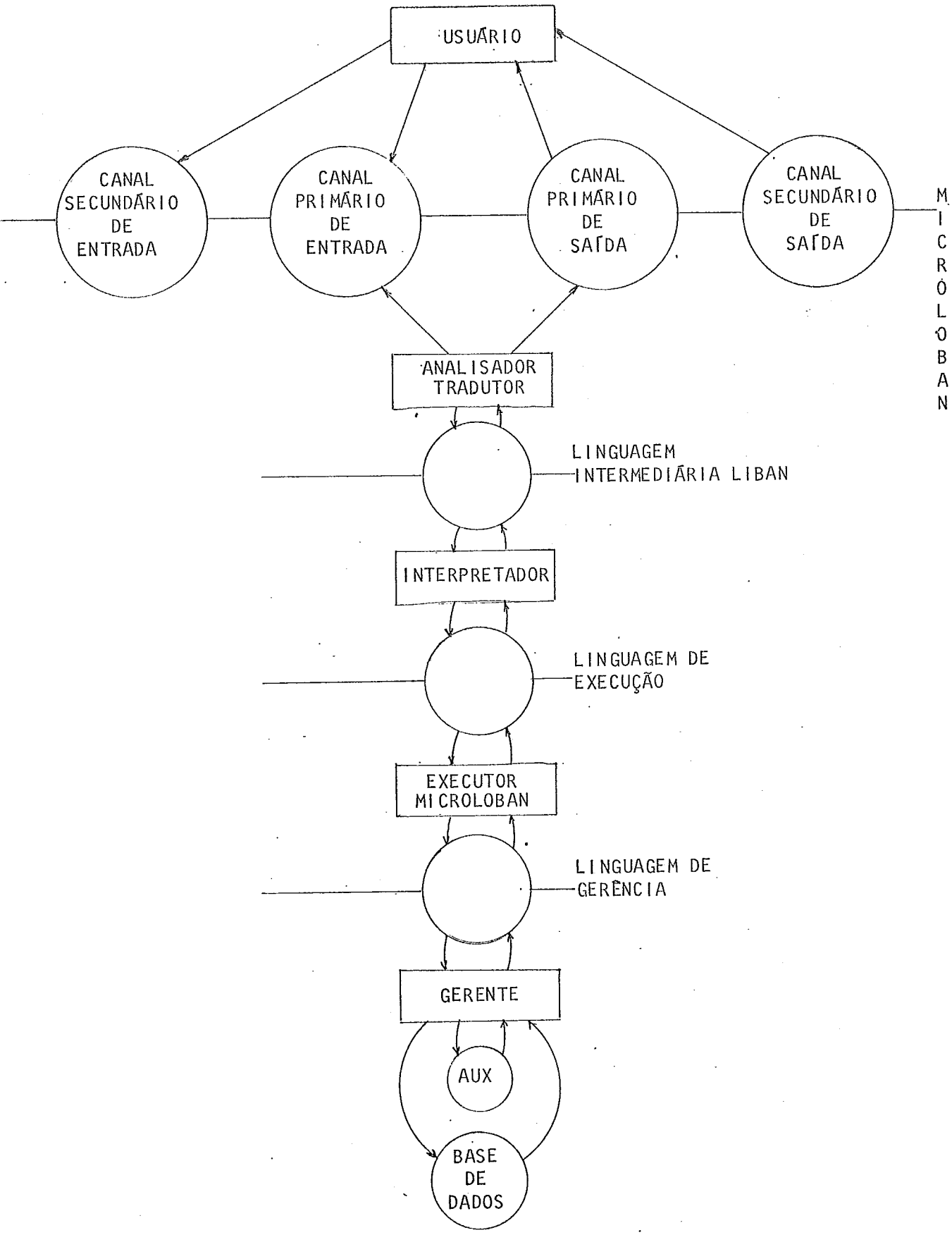



Figura 2.16 - Arquitetura Geral do Sistema MICROLOBAN

Para o analisador semântico optou-se por seguir a seguinte estratégia: toda análise semântica dependente de informações contidas no esquema conceitual (FOLHA) será transferida para a fase de execução mediante a inserção dos comandos correspondentes no código interpretável. Esta solução permitiu simplificar sensivelmente o analisador semântico e se justifica por duas razões: em primeiro lugar o tempo de análise, tradução e interpretação deverá ser desprezível frente ao tempo médio da execução de uma operação sobre a base de dados; em segundo lugar o modo preferencial de operação deverá ser "interativo", quando a execução se der comando a comando reduzindo o intervalo entre a análise e execução. O analisador de semântica estática assim reduzido compõe-se apenas de uma pilha de trabalho e de algumas rotinas comandadas pelo analisador sintático, ficando a função de análise semântica propriamente dita realmente a cargo do EXECUTOR MICROLOBAN.

O ANALISADOR também comanda o TRADUTOR que gera código intermediário especificado em forma de árvore,na qual estão inseridos os comandos de análise semântica que dependem do conteúdo da base de dados. Este código intermediário em forma de árvore, saída da estação TRADUTOR, é linearizado gerando uma lista de nós da árvore percorridos em pós-ordem (CRIVOROT (12)).

O INTERPRETADOR recebe de TRADUTOR uma seqüência de nós de árvore gerada a partir dos comandos fornecidos, contendo todas as informações semânticas e de controle de execução. Cada nó representa uma instrução intermediária podendo ser executada pelo próprio interpretador ou sua execução ser passada para o EXECUTOR. O interpretador comando o funcionamento da arquitetura.

Uma característica importante do Sistema é o fato de não ser necessária a criação de uma tabela de símbolos para o ANALISADOR, devido ao fato da análise semântica ser realizada em tempo de execução consultando o esquema conceitual do acervo setorial que é a folha correspondente.

O canal de comunicação entre as estações INTERPRETADOR e EXECUTOR é capaz de armazenar valores booleanos, definições de tipo e pretipo de construções, código intermediário e outras informações colocadas por estas duas estações. Este canal possui também uma área de "buffers" onde serão lidas/escritas tuplas, cadeias de caracteres e textos fontes a serem gravados na Folha da Base de Dados.

O EXECUTOR tem como função principal a execução das instruções "geradas" pelo interpretador e realizada através das rotinas semânticas. É esta estação que comanda o acesso e a manipulação do conteúdo do Canal Auxiliar e do Canal Base de Dados, além do que diz respeito a entrada e saída de dados (comunicação com o meio externo) e avaliação de expressões.

As Rotinas Semânticas utilizam as primitivas de gerenciamento interno para realização de acesso ao conteúdo da Base de Dados, além de tratar da parte referente a entrada e saída de dados externos. Maiores detalhes sobre o gerenciamento interno e métodos de acesso a Base de Dados e a Construção Auxiliar, são especificados em DANTAS (14).

A comunicação do Sistema com o meio externo é feita pelos Canais Secundários de Entrada/Saída, possibilitando a leitura/gravação de dados em dispositivos de armazenamento secundário.

# CAPÍTULO III

## Uma Proposta de Especificação de Código Intermediário para Linguagens de Acesso a Banco de Dados

### 3.1. Problemas de Análise Semântica em DDL

As linguagens de acesso a banco de dados, ditas autocontidas, englobam comandos de definição, manipulação e gerência de dados, podendo ser implementadas sem o auxílio de uma linguagem de programação(linguagem hospedeira). Estas linguagens geralmente armazenam as definições dos dados na própria base de dados, podendo as mesmas serem posteriormente utilizadas pelos programas de aplicação escritos pelo usuário.

Esta característica leva a tomar decisões importantes com respeito a realização da análise semântica estática. Desta forma dois enfoques possíveis são analisados em termos de custos e benefícios. A primeira possibilidade seria deixar a análise semântica ser realizada pelo Analisador/Tradutor que acessaria a base de dados para validar os operandos das instruções fornecidas. Isto inviabilizaria a operação do tradutor para processamento em lotes, pois as definições dos dados na base de dados são dinâmicas e nada garante que um determinado identificador teria a mesma definição no momento em que o código intermediário estivesse sendo gerado pelo tradutor e posteriormente, quando este código fosse executado. Vale a pena frisar que um programa, escrito em uma determinada linguagem de acesso a banco de dados, poderia alterar os tipos de dados durante a sua execução.

Outro enfoque a ser analisado seria postergar a aná

lise semântica estática para a fase de Interpretação/Execução, fazendo com que o Analisador/Tradutor execute apenas as funções de análise léxica e sintática dos comandos fornecidos (CRIVOROT (12)).

Um exemplo de implementação deste tipo de linguagem é a que está sendo realizada pelo Departamento de Computação da Universidade Federal do Rio Grande do Sul. A linguagem que está sendo implementada é um subconjunto da linguagem LOBAN no minicomputador LABO/8034. Para resolver o problema referente à análise semântica, eles optaram por simularem as operações intermediárias LOBAN para execução posterior dos testes semânticos. Este método perde um pouco em eficiência devido o fato de "executar" um mesmo comando mais de uma vez, ou seja, na fase de geração do código intermediário e na fase de execução propriamente dita. Além disso para a execução acima descrita, terá que ser acessado a base de dados para validação dos identificadores utilizados cujas definições se encontram armazenadas na base de dados. (Maiores informações ver HEUSER (18), LIMA (20)).

Como se pode notar, nem sempre é vantajosa a especificação do analisador semântico da forma acima descrita. Desta forma a solução viável a ser usada quando temos este tipo de linguagem é realizar a análise semântica estática durante as fases de interpretação e execução. A solução desenvolvida para este tipo de problema foi incluir o código, que vai efetuar os testes semânticos, nas especificações das árvores a serem passadas para o interpretador, que a insere, por sua vez, no código executável.

## 3.2. Formalização da Interface LINGUÍSTICA/INTERMEDIÁRIA

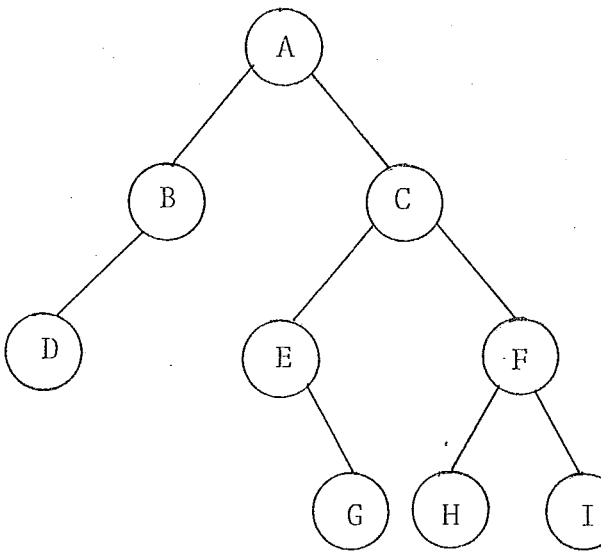
Fara a formalização da solução fornecida para resolver os problemas referentes a análise semântica nas linguagens de acesso à banco de dados, abordaremos os aspectos de especificação e implementação separadamente.

ESPECIFICAÇÃO:

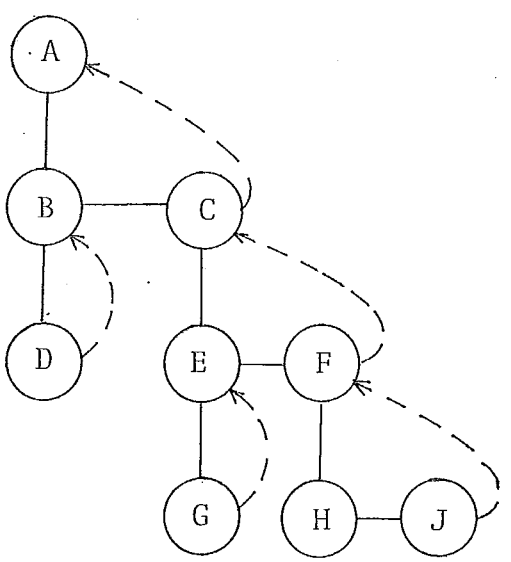
A especificação do código intermediário gerado pelo Analisador/Tradutor é fornecida através de árvores binárias costuradas onde estão incluídos, além das operações intermediárias, os testes de semântica estática e opcionalmente as operações de controle de execução. Desta forma, a especificação da interface entre o Tradutor e Interpretador ficou reduzida unicamente a árvore de códigos interpretáveis.

Estas árvores de código são percorridas em pós-ordem, ou seja, percorre a sub-árvore esquerda, depois a sub-árvore a direita e por fim visita a raíz. Assim sendo, um percurso completo através de uma árvores nos fornece um arranjo linear de nós, de modo que é possível falar-se em nó sucessor e nó predecessor de outro, em uma dada seqüência.

As árvores são representadas graficamente com os filhos de um mesmo pai ligados através de linhas horizontais, existindo apenas uma ligação de um pai a seu primeiro filho, filho primogênito, através de linhas verticais. Esta representação é denominada de árvore binária com ligações primogênito/irmão e mostrada graficamente na figura 3.1.




Árvore binária




Árvore binária com ligações primogenito/irmão costurada

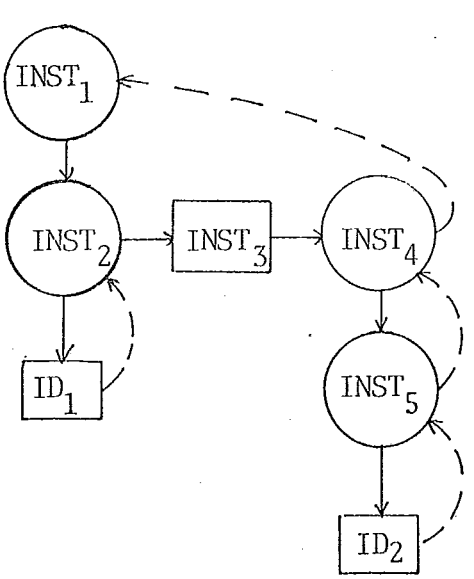
Figura 3.1 - Representação gráfica de uma árvore binária

A árvore binária utilizada para especificação do código a ser operado para cada produção de uma determinada linguagem é formada basicamente por dois tipos de nós: nó intermediário e nó folha. O nó intermediário pode conter instruções cuja

execução resulta em um teste semântico, em uma ação de controle tomada pelo interpretador ou em uma operação realizada sobre o ambiente interno do sistema. O nó folha é usado,na maioria das vezes, para transportar elemento de dado extraído de uma determinada instrução juntamente com as suas informações semânticas, para serem utilizados pelas operações sobre a base de dados. Desta forma o conteúdo dos nós folha fica dependente das informações fornecidas pelo usuário em cada instrução fonte. Outros tipos de nós compõem a árvore binária gerada, cuja execução ficará a cargo do tradutor. São eles: nó EXECUTE e nó COSTURA. O nó COSTURA determina que o nó cuja ligação "irmão" está em aberto, deverá apontar para o seu nó "pai". O nó EXECUTE, por sua vez, foi incluido no código com a finalidade de indicar o momento de passar o controle ao interpretador. A existência deste nó se fez necessária devido as exigências de recuperação do estado do interpretador em caso de erro sintático, nos modos interativos e em lote (CRIVOROT (12)).

A seguir será fornecido um exemplo da especificação da interface descrita acima:

EXEMPLO




Árvore binária costurada.

Seqüência de execução:

$ID_1$, $INST_2$, $INST_3$, $ID_2$, $INST_5$, $INST_4$, $INST_1$

Como se deve notar, após a geração da árvore correspondente a uma instrução fonte existe um processo de linearização que fornece como resultado uma seqüência de nós da árvore percorrida em pós-ordem.

IMPLEMENTAÇÃO:

A implementação da especificação da interface entre o Tradutor e o Interpretador não constrói a árvore binária efetivamente.

O Tradutor fornecerá para o interpretador uma seqüência de "nós" (instruções intermediárias), que será armazenada numa fila de execução. Esta seqüência corresponde ao percurso da árvore na ordem estabelecida até encontrar uma instrução intermediária EXECUTE, que funciona como uma instrução de controle de transferência entre o Tradutor e o Interpretador.

Como podemos observar, a comunicação entre Tradutor e Interpretador é realizada através de uma estrutura de fila, onde o interpretador executará as instruções fornecidas, uma a uma, na ordem de percurso determinada pelo Tradutor.

## 3.3. Interpretação de Linguagens de Acesso à Banco de Dados

No processo de desenvolvimento e implementação d uma linguagem de acesso à banco de dados é necessário defini sua estrutura geral levando em consideração a distinção entre os aspectos de especificação e implementação. Nos dois casos

serão definidos os módulos e as interfaces entre os mesmos, facilitando a realização do trabalho proposto. Consequentemente subdivide-se as tarefas de execução a serem realizadas para obtenção de um Sistema de Banco de Dados.

Este processo formaliza as tarefas de execução de modo que um determinado sistema totalmente especificado poderá ser implementado fisicamente sem possibilidades de erros, levando em consideração todos os detalhes de especificação. Esta formalização é feita através das unidades funcionais denominadas estação e canal. Uma estação é uma unidade funcional cuja função consiste na execução de atividades de processamento de informações. Um canal corresponde a uma unidade funcional cuja função consiste em assumir estados para a representação de informações entre estações, as quais geram estados e percebem estados nos canais. Maiores informações sobre estação e canal ver RICHTER (25).

A especificação proposta da estrutura geral de um sistema de banco de dados fornece várias máquinas virtuais definidas em cascata. A interface entre cada uma dessas máquinas define as informações necessárias a realização de operações sobre os seus registradores. A representação da especificação de uma estrutura geral de um SBD é apresentada na figura 3.2.

Assim definido, um Sistema de banco de dados é modularizado de maneira a subdividir o trabalho de implementação, permitindo que a execução de cada módulo seja feita separadamente e paralelamente aos demais que compõem o sistema.

Um desses módulos de grande importância é o interpretador cuja especificação corresponde a uma máquina virtual

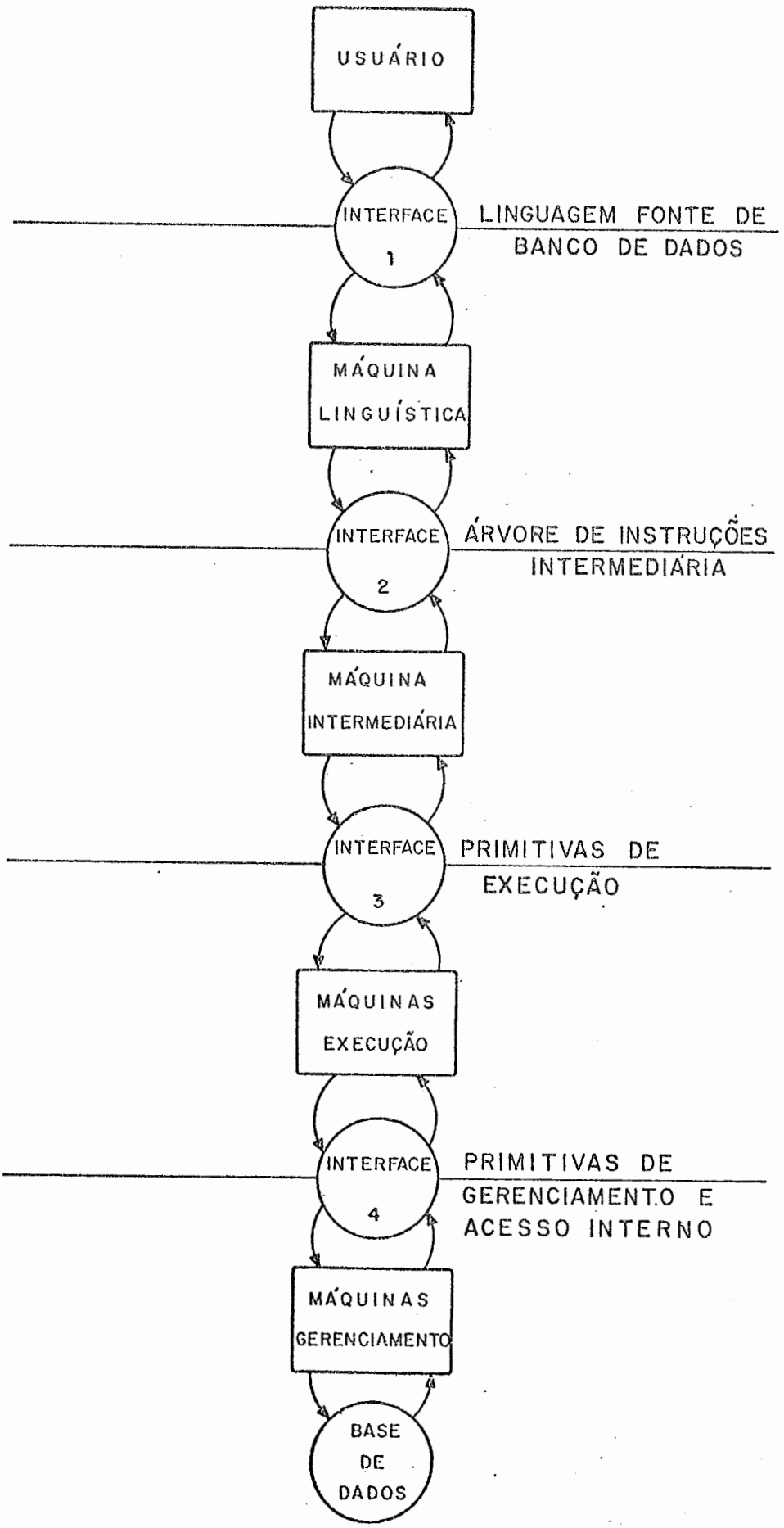



Figura 3.2 - Especificação de uma Estrutura Geral de um SBD.

de função intermediária entre o nível externo e o nível interno. O interpretador, como vemos, é a responsável pelo funcionamento satisfatório de toda estrutura global especificada inicialmente.

Utilizando-se a interface especificada na seção 3.2 para estabelecer a comunicação entre a Máquina Virtual Intermediária e a Máquina Virtual Linguística, temos a implementação do Interpretador razoavelmente simplificada.

O Interpretador terá como funções principais a execução dos códigos intermediários e a detecção de erro de execução. As instruções intermediárias são fornecidas pelo Tradutor que as armazena numa fila de execução. Cada instrução intermediária gera um conjunto de primitivas de execução que são fornecidas, uma a uma, ao Executor pelo Interpretador. A responsabilidade de detectar e tratar dos erros de execução que poderão ocorrer ficará também a cargo do interpretador.

## 3.4. Aplicação do Método de Interpretação Proposto à MICROLOBAN: a Linguagem Intermediária LIBAN

Nesta seção será descrita uma aplicação do método proposto de definição do código intermediário através de árvores binárias, e especificado o método de interpretação a ser utilizado usando tal código.

Utilizaremos a linguagem MICROLOBAN como exemplo, por esta possuir as características necessárias para a definição do referido método de tradução. Uma visão geral das características principais desta linguagem foi fornecida no Capítulo II, seção 2.2.

Um dos aspectos importante a ser relembrado é o fa-

to de MICROLOBAN ser uma linguagem autocontida englobando os comandos de definição, gerência e manipulação de dados, não necessitando desta forma de nenhuma outra linguagem para a geração da Base de Dados. Outro fato importante é que toda a definição dos dados contidos na base de dados se encontra armazenada na construção Folha que é componente imediato do acervo setorial. Como se pode observar a maioria dos testes semânticos a serem realizados deverá acessar à base de dados. Assim sendo, a análise semântica foi transferida para a fase de execução mediante a inserção dos códigos correspondentes no código intermediário. Esta solução reduziu o tempo de análise e tradução, tornando-o desprezível face ao tempo médio gasto na execução de uma operação sobre a base de dados.

Como conseqüência desta decisão, MICROLOBAN será traduzida para código interpretável especificado em forma de árvores binárias costuradas. Estas árvores são geradas e linearizadas pelo tradutor ficando o controle de execução a cargo do Interpretador que comanda o funcionamento de todo o sistema.

O código intermediário especificado desta forma permitiu uma visualização da seqüência de execução dos comandos intermediários gerados a partir de um programa fonte MICROLOBAN, não sendo necessária informação adicional sobre as ligações entre os nós da árvore. Porém, existem certas informações incluidas neste código além das referentes aos testes semânticos. Estas informações servem para auxiliar o interpretador na execução da seqüência de instruções geradas a partir de uma instrução fornecida pelo usuário. Como conseqüência dessas características foi possível modularizar a especificação de tal código fazendo com que a função de cada instrução intermediária esteja

bem determinada e sua execução independa da execução de outras instruções. Para se obter esta independência de execução foi utilizada a especificação de uma Máquina Virtual denominada Máquina Virtual Intermediária MICROLOBAN, descrita posteriormente neste Capítulo.

Após a especificação do método de tradução a ser realizado bem como do conteúdo do código a ser gerado, iniciou-se um processo de validação e otimização da forma de representação e do conteúdo dos nós das árvores binárias, chegando-se a uma especificação considerada razoavelmente satisfatória. (Vide Anexo III)

No decorrer da realização deste processo, foi inicializado, em paralelo, o trabalho de especificação dos algoritmos a serem utilizados na execução de uma instrução MICROLOBAN. Portanto, no final desta etapa, tivemos como resultados as especificações das árvores binárias das produções e dos procedimentos utilizados na execução de cada código intermediário a um nível de detalhe desejável. A linguagem intermediária resultante da linearização da árvore binária realizada pelo tradutor é denominada LIBAN, tendo algumas de suas instruções descritas no Anexo IV.

### 3.4.1. Máquina Virtual Intermediária MICROLOBAN

Uma ferramenta de grande importância utilizada na especificação de execução do código intermediário é a máquina virtual intermediária MICROLOBAN.

Sua estrutura física é composta de registradores capazes de armazenar valores booleanos, identificadores e de outras

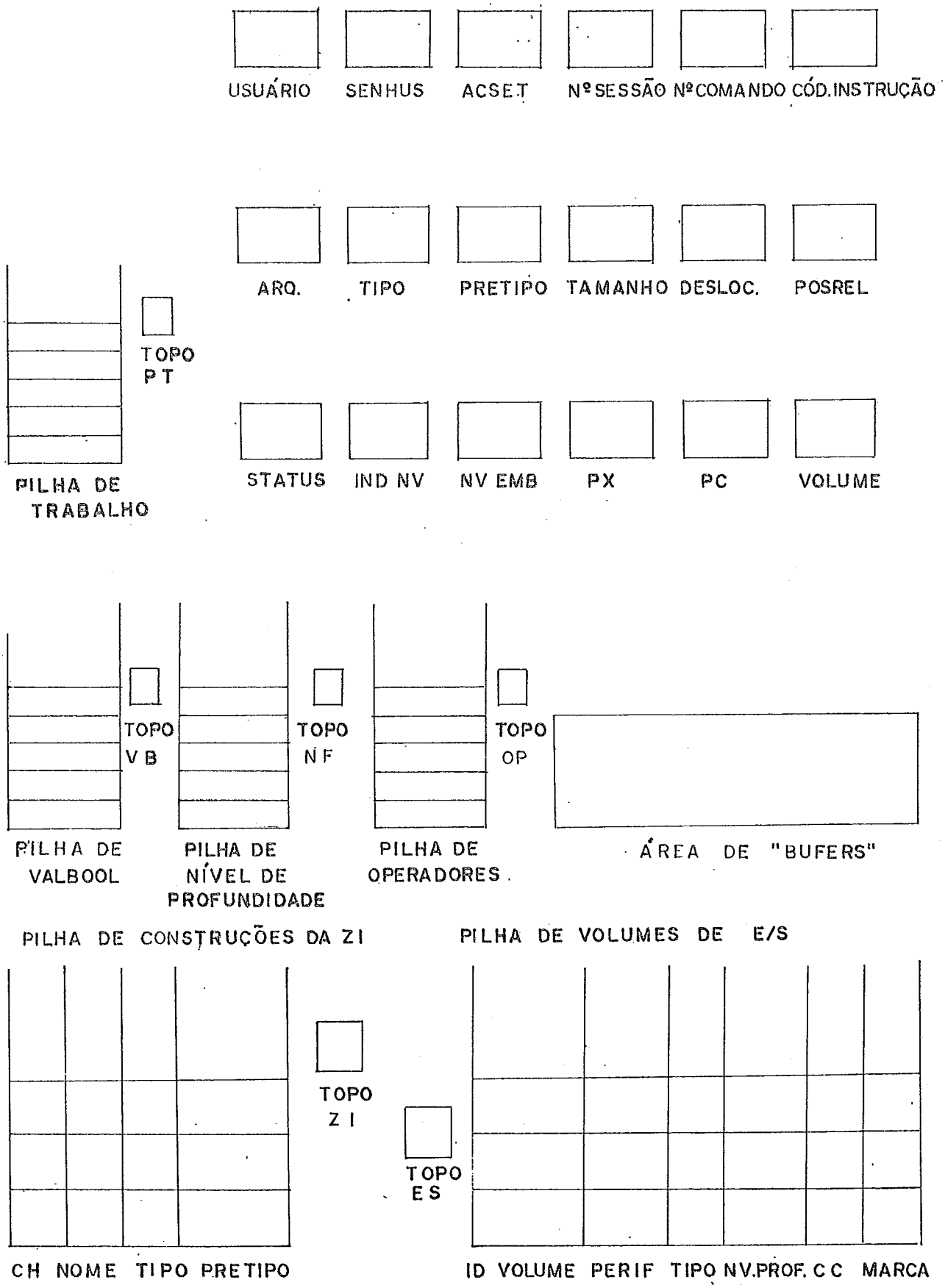



Figura 3.3 - Máquina Virtual Intermediária MICROLOBAN.

informações necessárias à execução do referido codigo, fornecido pelo tradutor, além de possuir pilhas que conterão operadores, descrição das construções existentes na Zona Intermediária, descrição dos volumes abertos para uma execução e outras informações. A máquina MICROLOBAN possui também uma área de "buffers" onde serão lidas/escritas tuplas, cadeias de caracteres e textos fontes MICROLOBAN a serem gravados na folha do acervo setorial.

Após a execução de uma instrução LIBAN, a máquina virtual terá em seus componentes os resultados obtidos e que eventualmente servirão na execução das próximas instruções. A estrutura física da máquina virtual MICROLOBAN é apresentada na figura 3.3.

Portanto, todo acesso e manipulação da Base de Dados, Canal Auxiliar e Zona Intermediária, é realizado por meio da máquina MICROLOBAN. Outras funções dessa máquina diz respeito a entrada e saída de dados e avaliação de expressões.

### 3.4.2. Interpretador MICROLOBAN

O Interpretador MICROLOBAN é responsável pela execução dos comandos LIBAN gerados e fornecidos pelo tradutor. Além desta função, ele possui a incumbência de detectar a ocorrência de erros nas execuções das instruções intermediárias e de controlar os laços internos contidos no referido código.

A interface entre o tradutor e o interpretador é realizada através de uma seqüência de instruções LIBAN, contendo as operações sobre o ambiente interno, as operações que realizam os testes semânticos necessários e as operações de contro

le de execuçao. Para receber esta "seqüência" de instruções, o interpretador MICROLOBAN dispõe de uma "fila" de execução onde as instruções são armazenadas na ordem de execução (execução sequencial).

O Interpretador chama o tradutor que inicia a passagem das instruções intermediárias, colocando-as na fila de execução até encontrar a instrução especial EXECUTE. A partir desse ponto o interpretador inicia o processo de execução das instruções fornecidas, acessando-as na fila para identificação das rotinas semânticas (primitivas de execução) cujas chamadas se fazem necessárias para realização das ações sobre a Base de Dados.

O Interpretador também é responsável pelo controle de laços usando uma estrutura em pilha para tal operação. Ele identificará o início e o fim de laço pelas instruções intermediárias de controle INIC LAÇO e FIM LAÇO respectivamente.

Quando algum erro ocorre durante a execução de uma instrução LIBAN, o interpretador poderá detectar a ocorrência de tal erro e ativar o módulo do Sistema responsável pelo tratamento do mesmo. Todas as instruções contidas na fila de execução serão desprezadas e a base de dados será reconstruida a um estado anterior a execução da referida instrução MICROLOBAN fornecida pelo usuário. Vale ressaltar que o procedimento descrito acima é válido apenas para o modo interativo.

No modo "batch", o programa fornecido pelo usuário é traduzido e enviado em sua totalidade para o interpretador. Sendo assim, quando uma ocorrência de erro é detectada pelo Interpretador, toda instrução de trabalho será desconsiderada e

a base de dados será reconstruida a uma versão anterior.

### 3.4.3. Exemplificação da Aplicação do Método Proposto

Após a descrição do ambiente necessário à execução dos comandos LIBAN, partiremos para especificar e exemplificar algumas produções da linguagem MICROLOBAN, cuja sintaxe, escrita como uma gramática regular, se encontra no Anexo II.

Serão consideradas apenas algumas das produções que descrevem as expressões operandos utilizadas pelas instruções autônomas MICROLOBAN.

EXPRESSÃO *end-ponto*

No processo de especificação da árvore binária referente ao código a ser gerado pelo tradutor para a expressão endereço de ponto, foram feitas restrições consideradas razoáveis devido ao porte do sistema portador.

Uma das restrições feitas foi o fato do endereçamento a componentes imediatos de nominação só ser realizado através dos seus respectivos nomes. Por outro lado, o endereçamento a componentes imediatos de coleção será feito apenas por conteúdo. Vale a pena observar que nenhuma restrição foi feita quanto ao nível de embutimento deixando o endereçamento de ponto tão "potente" quanto no sistema LOBAN.

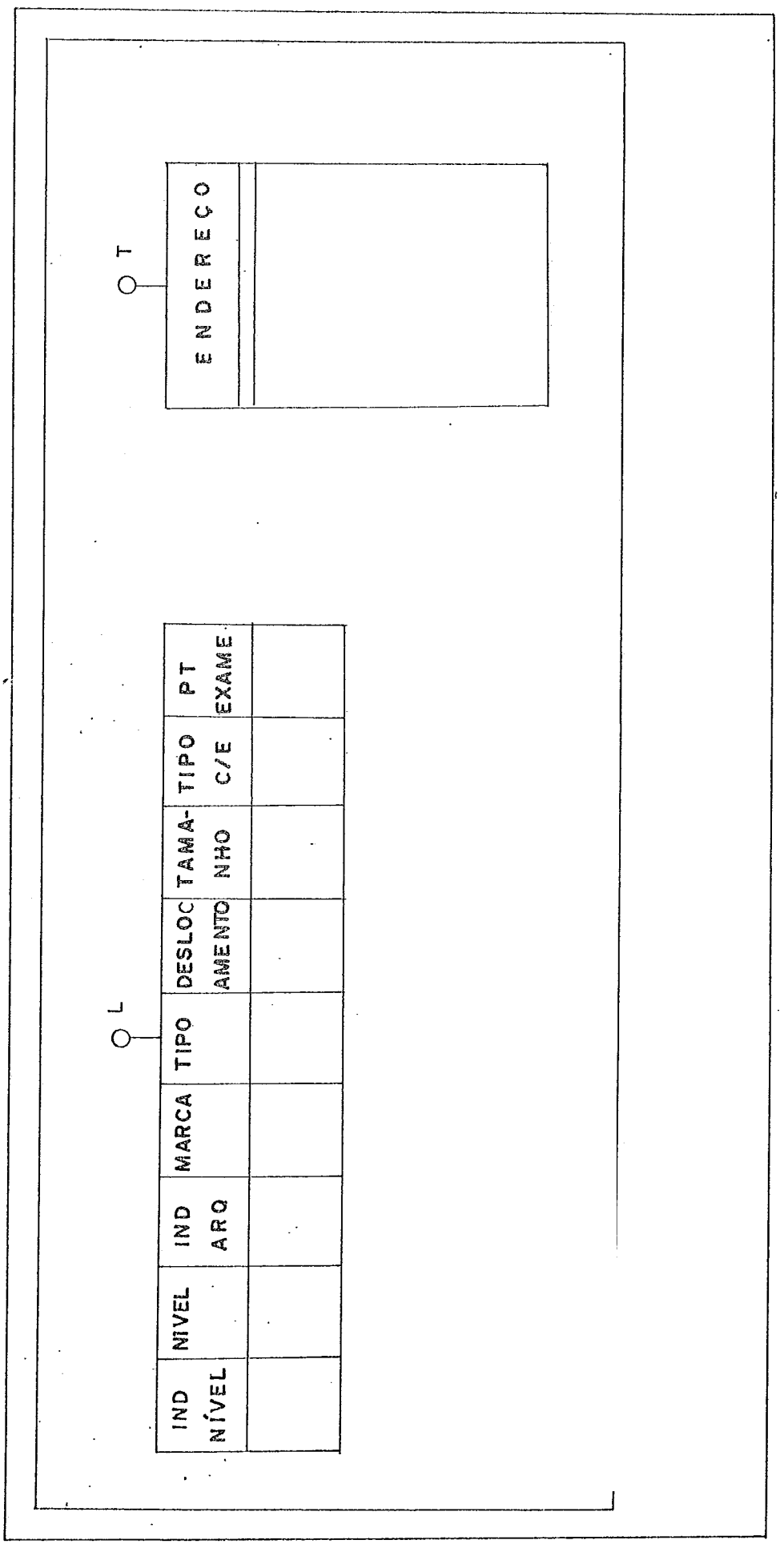



Figura 3.4 - Tabela de Avaliação de Endereço de Ponto (TABAVL).

| CÓDIGO | C O N S T R U Ç Ã O |
|---|---|
| #00 | nível de ACTRAB |
| #10 | nível de AUX |
| #01 | nível de FOLHA |
| #02 | nível de FICHA |
| #03 | nível de arquivo |
| #n4 | nível de TAREL |
| #n5 | nível de TALIG |
| #n6 | nível de COLITENS |
| #n7 | nível de TUPLA |
| #n8 | nível de LIGAÇÃO |
| #n9 | nível de ligante |
| #nA | nível de atributo composto |
| #nB | nível de atributo simples |
| #nC | nível de componente de coleção de tuplas |
| #nD | nível de componente de coleção de ligações |
| #nE | nível de componente de coleção de itens |
| #nF | nível de construção sem tipo |

onde n = $\begin{cases} 0 \rightarrow \text{contexto de referência ACTRAB} \\ 1 \rightarrow \text{contexto de referência AUX} \end{cases}$

Tabela 3.1. - Indicadores de nível.

Para a especificação do algoritmo a ser empregado na avaliação de um endereço de ponto, fez-se necessária a associação de valores específicos a cada pretipo padrão de construção que pode compor o conteúdo do acervo setorial. Esses valores estão descritos na tabela 3.1.

Além do ambiente especificado pela máquina virtual intermediária MICROLOBAN, tivemos que definir uma construção, do pretipo TALIG, para ser utilizada na avaliação desta expressão. Com esta estrutura ficou garantida a liberdade de se especificar qualquer nível de embutimento de endereço de ponto perdendo-se um pouco em eficiência pois esta tabela, denominada TABAVL, será armazenada em memória secundária. A representação gráfica da tabela TABAVL se encontra na figura 3.4.

Posteriormente será descrito o algoritmo desenvolvido para avaliação da expressão endereço de ponto.

ALGORITMO DE AVALIAÇÃO DE ENDEREÇO DE PONTO

- Incrementar nível de embutimento.
- Avaliar o identificador de ponto fornecido.
- Preparar conjunto de pontos candidatos.
- Executar para cada critério fornecido:
  - Tornar um ponto candidato em ponto examinado:
  - Executar laço sobre pontos candidatos:
    - Avaliar critério fornecido para o ponto examinado.
    - Caso resultado do critério seja verdadeiro, incluir ponto examinando no conjunto de pontos escolhidos.

- Verificar a existência de pontos candidatos. Caso exista tais pontos, transformar um ponto candidato em ponto examinando.
- Verificar a existência de conjunto de pontos escolhidos. Caso afirmativo, transformar este conjunto em conjunto de pontos candidatos.

- Decrementar nível de embutimento.

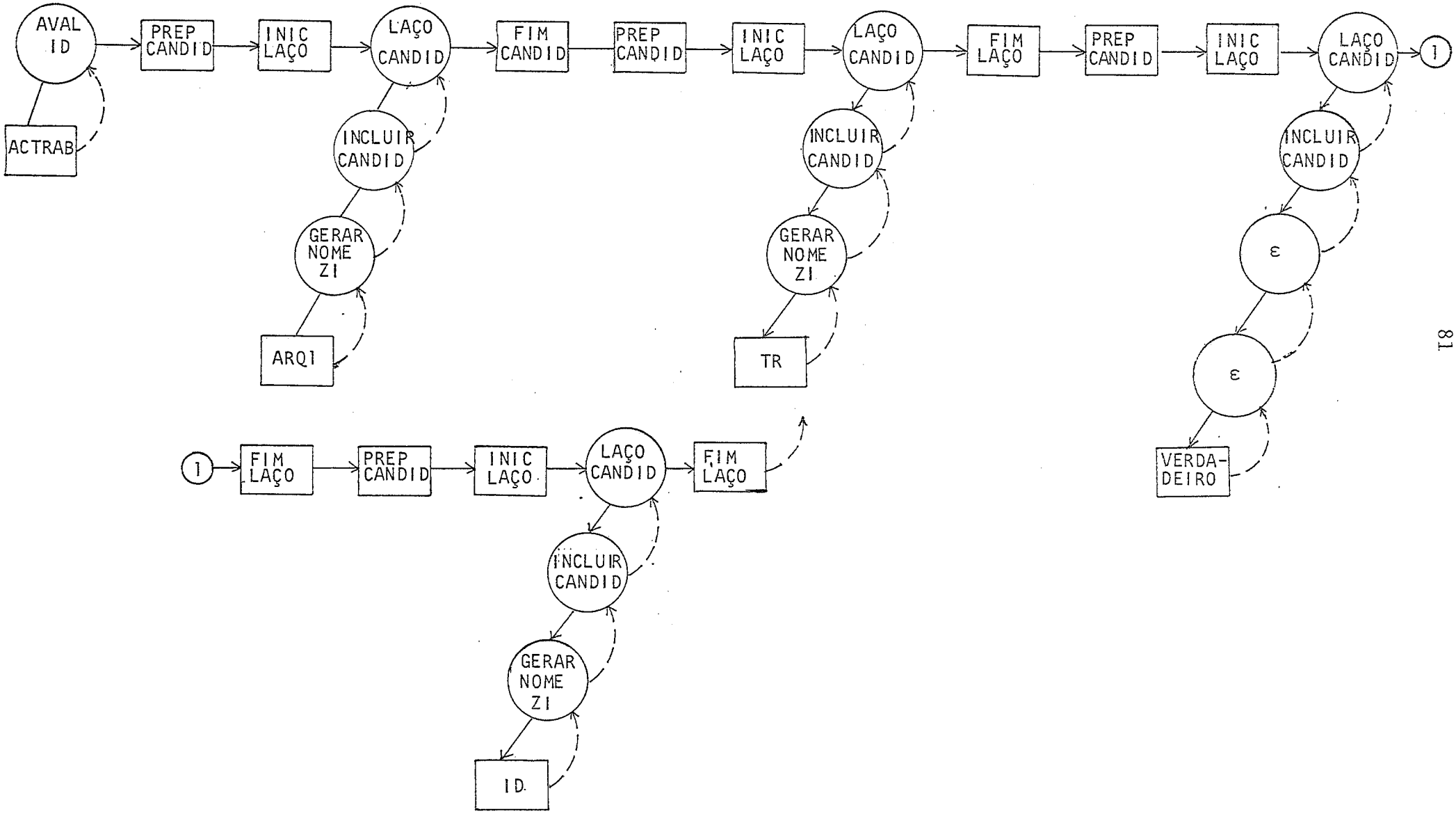
Com base no algoritmo descrito acima foi realizado o processo de definição do código intermediário a ser gerado para a expressão de endereço de ponto. A árvore binária para esta produção está especificada detalhadamente no Anexo III e a definição das instruções LIBAN no anexo IV.

Um exemplo de endereço de ponto será fornecido mostrando a árvore binária gerada pelo tradutor e a sequência de instruções LIBAN após o processo de linearização.

Exemplo de endereço de ponto:

. ACTRAB
. ARQ1
. TR
. VERDADEIRO
. ID

Especificação da Saída do Tradutor



| Conteúdo da fila de execução | Página |
|---|---|
| ACTRAB | 130 |
| AVAL ID | 123 |
| PREP CANDID | 132 |
| INIC LAÇO | 163 |
| ARQ1 | 130 |
| GERAR NOME ZI | 136 |
| INCLUIR CANDID | 137 |
| LAÇO CANDID | 136 |
| FIM LAÇO | 164 |
| PREP CANDID | 132 |
| INIC LAÇO | 163 |
| TR | 130 |
| GERAR NOME ZI | 136 |
| INCLUIR CANDID | 137 |
| LAÇO CANDID | 136 |
| FIM LAÇO | 164 |
| PREP CANDID | 132 |
| INIC LAÇO | 163 |
| VERDADEIRO | 160 |
| ε | |
| ε | |
| INCLUIR CANDID | 137 |
| LAÇO CANDID | 136 |
| FIM LAÇO | 164 |
| PREP CANDID | 132 |
| INIC LAÇO | 163 |

Conteúdo da fila de execução (Continuação) Página



Expressão *exp-booleana*

Esta expressão teve a especificação de seu código intermediário bastante longa devido a certas flexibilidades oferecidas pela linguagem MICROLOBAN. O seu uso adquire grande extensão quando aplicado no processo de endereçamento de ponto devido a possibilidade de fornecimento de critérios de vários tipos que executam as operações de seleção de pontos.

Dentre as flexibilidades citadas existe a que possibilita ao usuário fornecer um critério de ponto imediato em notação explícita, ou seja, seleção dos pontos através de seus nomes utilizando os operadores intermediários MICROLOBAN N e o que obtém um literal nome na ZI. Para que este tipo de endereçamento fosse acessível ao usuário, foi necessário desenvolver as instruções LIBAN de forma a considerar-se algumas características como a indicação que o ambiente externo à avaliação da expressão booleana diz respeito a endereçamento de ponto ou não.

As expressões operandos desse tipo de expressão resultam nos mais diversos tipos de construções, fazendo com

que os testes semânticos a serem realizados sobre uma expressão booleana venham a serem os mais rígidos possíveis.

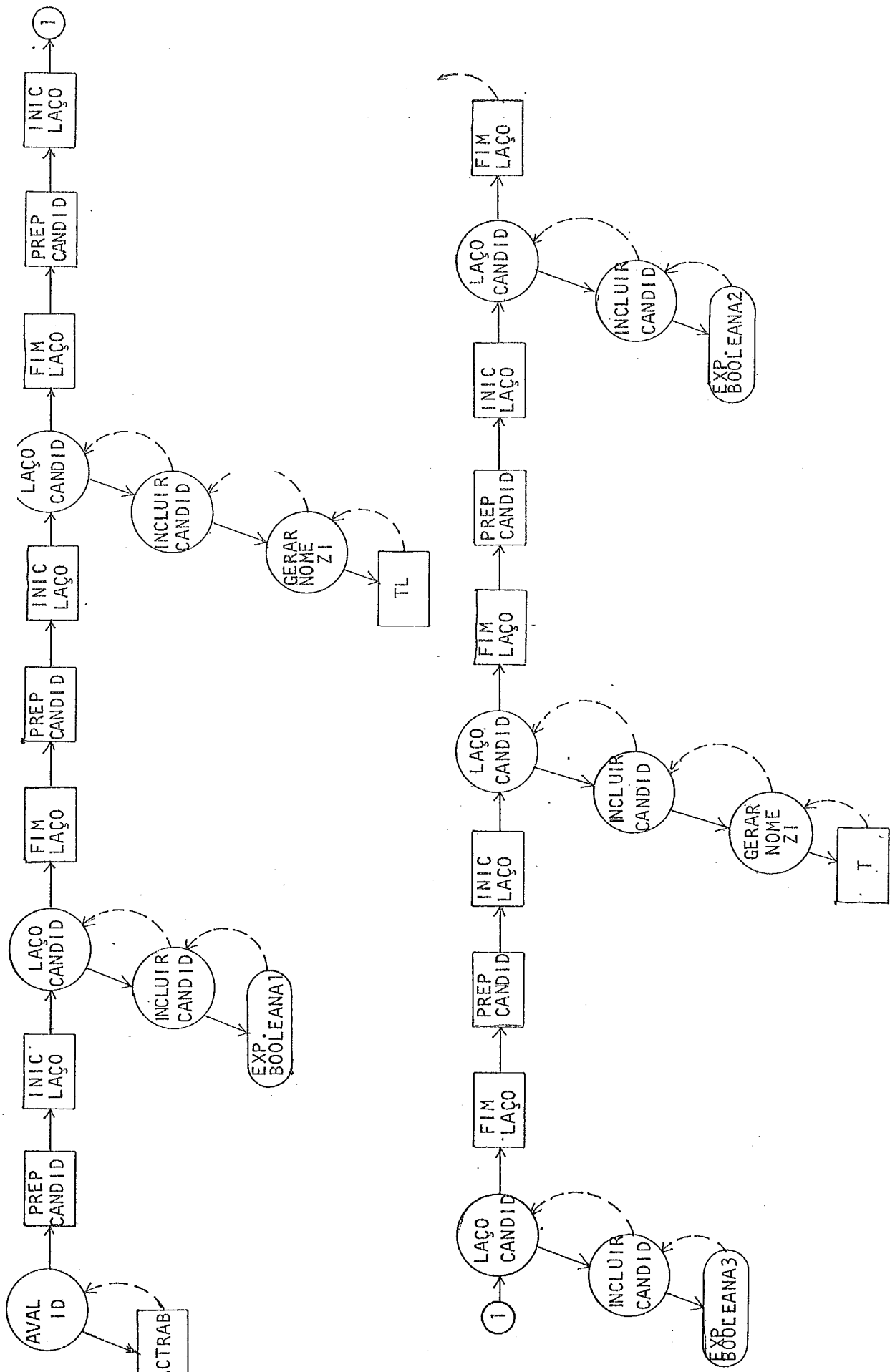
Para a avaliação de uma expressão booleana não será necessário criar nenhum outro ambiente além do oferecido pela máquina virtual MICROLOBAN.

A seguir é fornecido um exemplo de uma expressão booleana utilizada dentro da expressão de endereçamento de ponto.

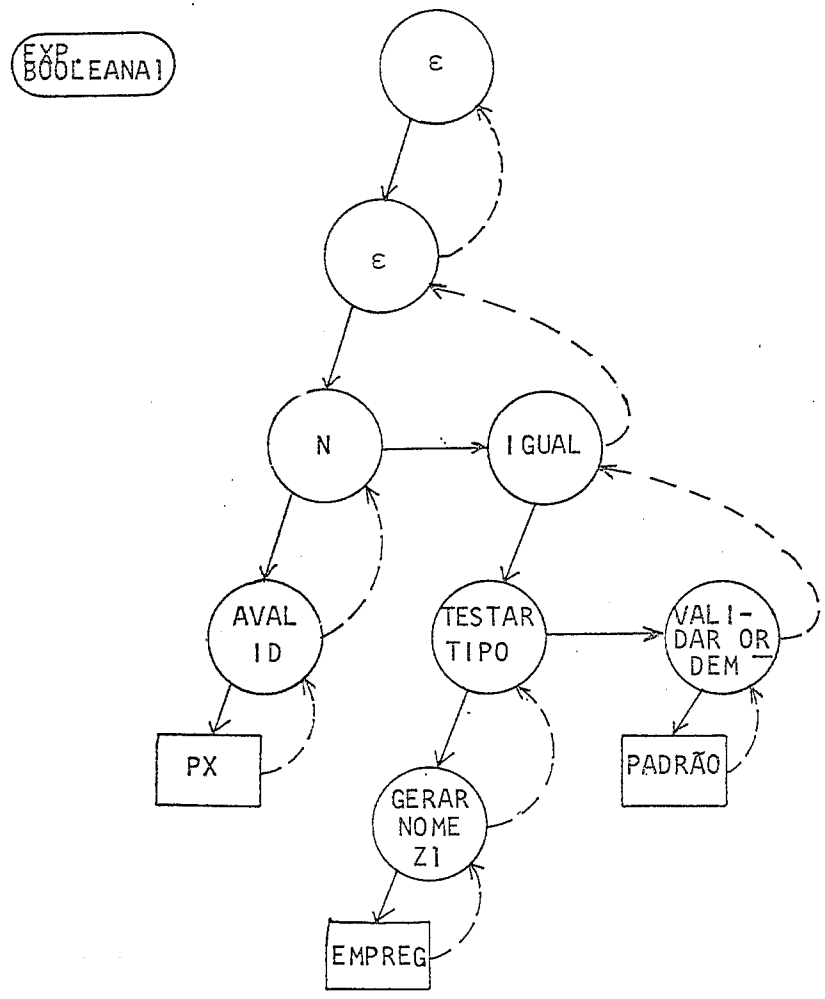
Exemplo de uma aplicação de expressão booleana

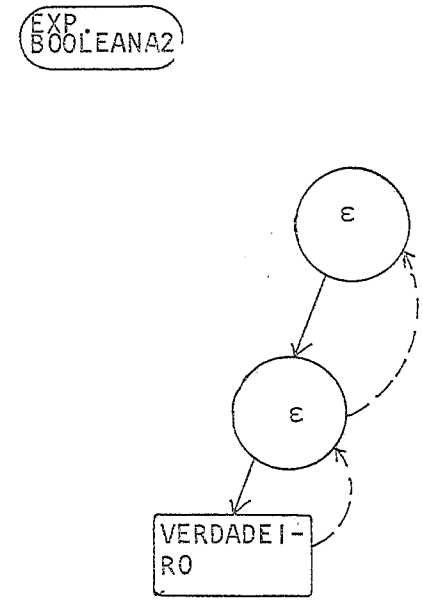
```
 ACTRAB
.N PX = EMPREG
.TL
.C PX
  .L
  .NUMID
 =
 id
.T
.VERDADEIRO
```

Especificação da Saída do Tradutor

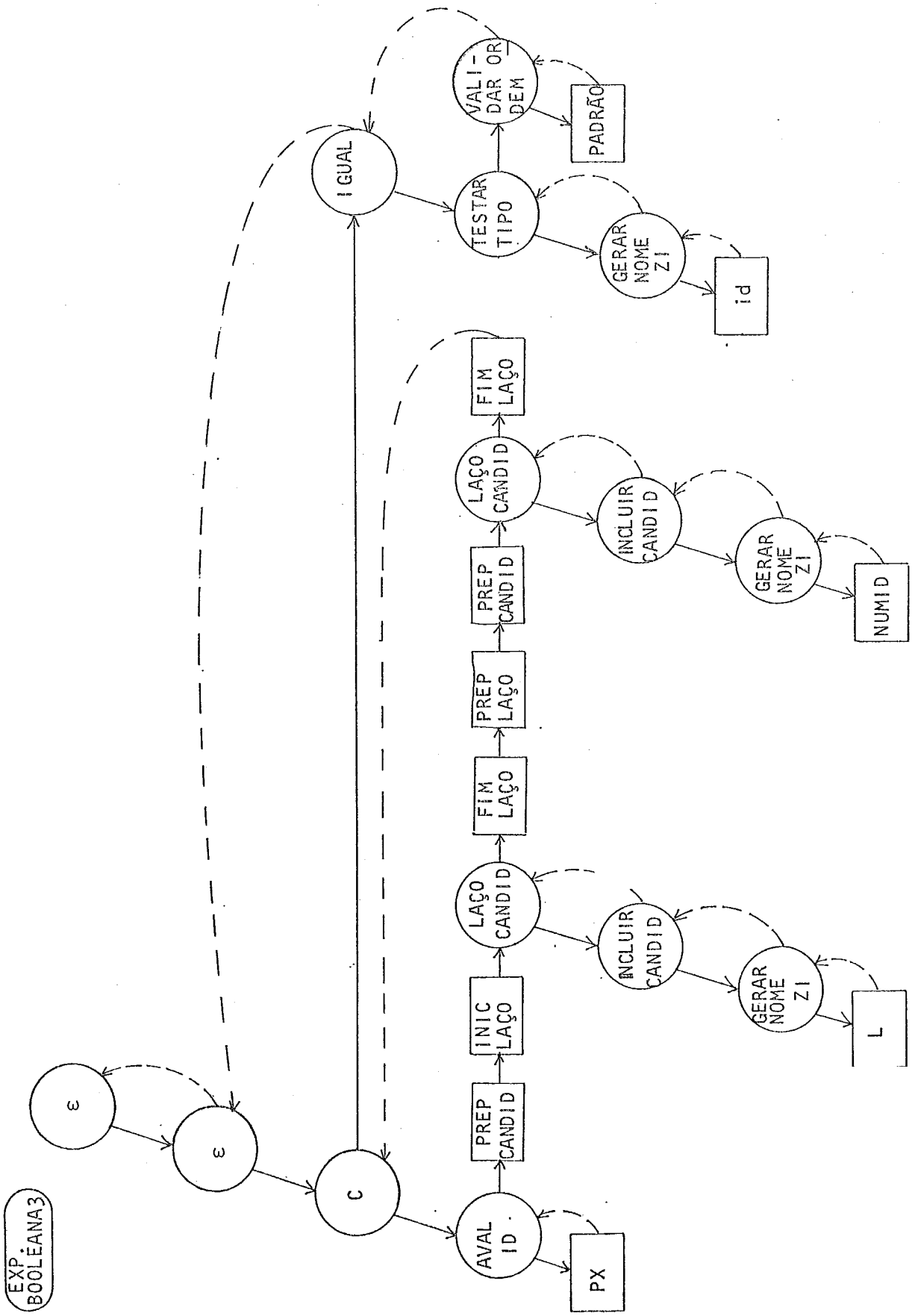


Especificação da Saída do Tradutor (Continuação)



Especificação da Saída do Tradutor (Continuação)



| Conteúdo da Fila de Execução | Página |
|---|---|
| ACTRAB | 130 |
| AVAL ID | 123 |
| PREP CANDID | 132 |
| INIC LAÇO | 163 |
| PX | 130 |
| AVAL ID | 123 |
| N | 163 |
| EMPREG | 130 |
| GERAR NOME ZI | 136 |
| TESTAR TIPO | 163 |
| PADRÃO | 162 |
| VALIDAR ORDEM | 163 |
| IGUAL | 163 |
| ε | |
| ε | |
| INCLUIR CANDID | 137 |
| LAÇO CANDID | 136 |
| FIM LAÇO | 164 |
| PREP CANDID | 132 |
| INIC LAÇO | 163 |
| TL | 130 |
| GERAR NOME ZI | 136 |
| INCLUIR CANDID | 137 |
| LAÇO CANDID | 136 |
| FIM LAÇO | 164 |
| PREP CANDID | 132 |
| INIC LAÇO | 163 |
| PX | 130 |
| AVAL ID | 123 |
| PREP CANDID | 132 |
| INIC LAÇO | 163 |
| L | 130 |
| GERAR NOME ZI | 136 |
| INCLUIR CANDID | 137 |
| LAÇO CANDID | 136 |
| FIM LAÇO | 164 |

| Conteúdo da Fila de Execução (continuação) | Página |
|---|---|
| PREP CANDID | 132 |
| INIC LAÇO | 163 |
| NUMID | 130 |
| GERAR NOME ZI | 136 |
| INCLUIR CANDID | 137 |
| LAÇO CANDID | 136 |
| FIM LAÇO | 164 |
| C | 162 |
| id | 130 |
| GERAR NOME ZI | 136 |
| TESTAR TIPO | 163 |
| PADRÃO | 162 |
| VALIDAR ORDEM | 163 |
| IGUAL | 163 |
| ε | |
| ε | |
| INCLUIR CANDID | 137 |
| LAÇO CANDID | 136 |
| FIM LAÇO | 164 |
| PREP CANDID | 132 |
| INIC LAÇO | 163 |
| VERDADEIRO | 160 |
| ε | |
| ε | |
| INCLUIR CANDID | 137 |
| LAÇO CANDID | 136 |
| FIM LAÇO | 164 |

# CAPÍTULO IV

# CONCLUSÕES

Como resultado do trabalho de pesquisa foi desenvolvido um método original de especificação de código intermediário não somente para linguagens de acesso a banco de dados, mas podendo o mesmo ser utilizado nas linguagens de programação convencionais. O método tem a sua formalização bastante simplificada devido ao critério adotado no desenvolvimento da especificação do projeto do sistema ser a modularização. Como conseqüência as funções de cada parte componente foram bem determinadas o que possibilitou o desenvolvimento dos módulos serem feitos paralelamente. Como podemos observar, a aplicação deste método se mostra bastante flexível pois permite que várias modificações possam ser feitas na linguagem fonte sem alterar o andamento do trabalho.

A separação entre os aspectos de especificação e implementação na definição de um determinado projeto resultou na possibilidade de utilização de um sistema desenvolvido na modelagem de outras linguagens. Como exemplo poderíamos citar o sistema MICROLOBAN que permite modelar outras linguagem sem alterar o núcleo básico, bastando para isto construir o tradutor da mesma. Um trabalho interessante a ser feito seria a modelagem de uma linguagem natural ou de uma linguagem de acesso a banco de dados já existente tal como SEQUEL (SOUZA (22), ASTRAHAN (01)).

Finalizando, é interessante informar que o desenvolvimento do Projeto MICROBAN só será realmente completo quando forem especificadas as funções do Executor e também da interface entre a máquina virtual intermediária e a máquina virtual utili-

zada na especificação do executor. Devido a este fato não foi possível a formalização da especificação da interface "de saída" do interpretador pois não foram definidas as operações e estrutura de dados da máquina virtual utilizada na especificação do executor.

# ANEXO I

## Sintaxe MICROLOBAN

## (Notação LOBAN)

# ANEXO I

## Sintaxe MICROLOBAN

<instrução de trabalho>::=
    <executar><lista de instruções><encerrar>

<executar>::=
  {EXECUTAR | COMPILAR} PARA <termo de apresentação>
  {INSTRUÇÕES | <id trab>} :

<encerrar>::=
  ENCERRAR

<lista de instruções>::=
  <instrução autonoma><fim texto><lista de instruções> |
  <instrução autonoma><fim texto>

<fim texto>::=
    ;

<instrução autonoma>::=
  <instrução de gerência> |
  <instrução de operação> |
  <instrução autônoma de controle de fluxo> |
  <instrução de navegação explícita> |
  <instrução de administração> |
  <instrução de proteção> |
  <bloco de transação> |
  <instrução de saída> |
  <instrução de providência de reconstrução> |
  <vazio>

<instrução de gerência>::=
  <criar acset> |
  <abolir acset> |
  <criar arquivo> |
  <abolir arquivo>

<instrução de operação>::=
<incluir construção> |
<excluir construção> |
<substituir construção>

<instrução autônoma de controle de fluxo ::=
<fazer para - autônomo> |
<fazer se - autônomo>

<instrução de navegação explícita>::=
<fixar marca> |
<tirar marca>

<instrução de administração>::=
<abrir acset> |
<fechar acset> |
<abrir volume> |
<fechar volume>

<instrução de proteção>::=
<estabelecer proteção> |
<abandonar proteção>

<bloco de transação>::=
<considerar transação>
(([<fazer ao acontecer falha>]
[<lista de instruções de transação>]...))

<instrução de providência de reconstrução>::=
<arquivar actrab> |
<abandonar acrec> |
<reconstruir comandos>

<instrução de saída>::=
<representar>

<criar acset>::=
CRIAR ACSET <lit nome> [A PARTIR DE ACSET
<n acset> ATÉ VERSAO <lit nint>]

<abolir acset>::=
ABOLIR ACSET <n acset>

<criar arquivo>::=
CRIAR { AREL | ALIG } <lit nome>

<abolir arquivo>::=
ABOLIR { AREL | ALIG } <n arq>

<incluir construção>::=
INCLUIR <obter construção> EM <end ponto> |
INCLUIR <obter construção> := <obter construção>
EM <end ponto> |
INCLUIR CADA COMPONENTE DE
<termo de conjunto de construções> EM <end ponto>

<excluir construção>::=
EXCLUIR DE <end ponto>

<substituir construção>::=
SUBSTITUIR EM <end ponto> POR <obter construção>

<fazer para-autônomo>::=
FAZER <prescrição de repetição>
<bloco fazer para-autônomo>

<fazer se-autônomo>::=
FAZER SE <obter valor booleano><bloco fazer se-autônomo>
SENÃO <bloco fazer se-autônomo>

<fixar marca>::=
FIXAR <lit marca> EM <end ponto>

<tirar marca>::=
TIRAR <n marca> ,,, [DE <end ponto>]

<abrir acset>::=
ABRIR ACSET <n acset> [COM <senhac> ,,,]

<fechar acset>::=
FECHAR ACSET [<n acset>]

<abrir volume>::=
ABRIR VOLUME DE { ENTRADA | SAIDA | INTERACAO } <id volume>
<descr volume>

<fechar volume>::=

FECHAR VOLUME DE { ENTRADA | SAÍDA | INTERAÇÃO } <id volume>

<estabelecer proteção>::=

ESTABELECER PROTEÇÃO <id proteção>

PARA{LER|ALTERAR} SOBRE { <n arq> | FOLHA | FICHA | ACTRAB } UUU [E PARA{LER|ALTERAR}SOBRE { <n arq> | FOLHA | FICHA | ACTRAB } UUU ]

<abandonar proteção>::=

ABANDONAR PROTEÇÃO <id proteção>

<considerar transação>::=

CONSIDERAR TRANSAÇÃO

<fazer ao acontecer falha>::=

AO ACONTECER {<dt falha>: <bloco fazer ao acontecer>} ,,,

<arquivar actrab>::=

ARQUIVAR ACTRAB

<abandonar acrec>::=

ABANDONAR ACREC

<reconstruir comandos>::=

RECONSTRUIR ACSET { EXECUTANDO | DESFAZENDO } <lit nint> COMANDOS

<representar>::=

REPRESENTAR SEGUNDO <id regra rep>

<obter construção> EM <termo de representação>

<bloco fazer ao acontecer>::=

CONT NREC |

DESCONT { TRANS | TRAB | INST } { REC | NREC }

<bloco fazer se - autônomo>::=

<bloco fazer para-autônomo>

<bloco fazer para - autônomo>::=

(( { <incluir construção> | <excluir construção> | <substituir construção> | <fixar marca> | <tirar marca> | <fazer para - autônomo> | <fazer se - autônomo> | <bloco de transação> | <representar> } ;;; ))

<lista de instruções de transação>::=

{ <incluir construção> | <substituir construção> | <excluir construção> | <fixar marca> | <tirar marca> | <fazer para - autônomo> | <fazer se - autônomo> | <representar> } ;;;

<verbete>::=

<verbete de coerência> |
<verbete de acesso> |
<verbete de autorização> |
<verbete de texto fonte>

<verbete de acesso>::=

SENHAC <senhac> {PERMITE <termo de acesso>} ,,,

<verbete de autorização>::=

USUÁRIO <id usuário> ((<senhus>)) COM <senhac> ,,;

<verbete de texto fonte>::=

FONTE <id texto fonte> : <lit fonte> #

<verbete de coerência>::=

<d tipo> : <especificação da coerência>

<especificação da coerência>::=
TIPO DE <d pretipo>
<verbete de consistência>

<verbete de consistência>::=
TAL QUE <termo de consistência>

<termo de apresentação>::=
USUÁRIO <id usuário> ((<senhus>))

<termo de conjunto de construções>::=
<d pretipo> |
<d tipo> |
{ <literal> ;;;} |
{ {VERBETE <verbete>};;;} |
<obter construção>

<termo de representação>::=
EM [AREA LIVRE] <end campo>

<termo de avaliação end ponto>::=
[ [= <lit marca>] ] <cri ponto imed>

<termo de construção>::=
INTR SEGUNDO <d tipo><end campo>
<literal>
N <end ponto>
C <end ponto>
COLEC <end ponto>
M <end ponto>
CONT <end ponto>
CALC <id cálculo> SOBRE <end ponto>
ALARG <obter tarel> DE {<atributo> ((<obter construção>))} ,,,
CARD <obter coleção>
DESAGRUP <obter talig>
ESTREIT <obter tarel> DE <atribut> ,,,
AGRUP <obter tarel> POR <atributo> ,,,
RENOM <obter tarel> SUBST{<atributo> POR <atributo>} ,,,
JUNT <obter tarel>$_1$ COM <obter tarel>$_2$ [EXCL]
([C <atributo>$_1$ {=|<|>|<=|>=|<>} C <atributo>$_2$ ])

LIGA <obter tarel>$_1$ COM <obter tarel>$_2$ [EXCL]
(([C <atributo>$_1$ {=|<|>|<=|>=|<>} C <atributo>$_2$]))
VAZ
HORA-CORR
DATA-CORR
VERBETE <verbete>
(( <obter construção> ))

<termo elementar valbool>::=
PARA TODO <end ponto> ((<obter valor booleano>))
EXIST<end ponto>(([<obter valor booleano>]))
C <end ponto> ELEM <termo de conjunto de construções>
{C / N} <end ponto> {=|>|<|>=|<=|<>} {<obter construção> / <lit nome>}
[EM ORDEM <d ordem>]
VERDADEIRO
(( <obter valor booleano> ))

<termo de controle>::=
<end campo> [ [= <n marca>] ]
<end ponto> [ [= <n marca>] ] [EM ORDEM
POR {C <atributo> {ASC / DESC}} ,,,]

<termo de acesso>::=
<d função> {ACTRAB / <n arq> / FICHA / FOLHA}

<termo de consistência>::=
COMPOS <termo de composição>
COMPOS <termo de composição> CONEX <termo de conexão>
<subtermo de consistência>[E <subtermo de consistência>]

<termo de composição>::=
<termo de composição de nominação>
<termo de composição de numeração de caracteres>
<termo de composição de coleção>

<termo de composição de nominação>::=

{<lit nome> → { <d tipo> | DATA | HORA | INT | REAL } } ,,,

<termo de composição de numeração de caracteres>::=

POR <lit nint> CARACTERES

<termo de composição de coleção>::=

<d tipo>

<termo de conexão>::=

{COLEC <end ponto conex> SUBCONJ
COLEC<end ponto conex>} ,,, |

CHAVE PRIMARIA <atributo> E CARD C OCOR <= <lit nint>

<subtermo de consistência>::=

C OCOR ELEM {<literal> ;;;} |

C OCOR {=|>|<|>=|<=|<>} <literal>

<end ponto>::=

<id ponto> |

[<id ponto>.] <termo de avaliação end ponto> ...

<end campo>::=

CC [ | <n marca>] |

VOLUME <id volume> . CAMPO (( ))

<obter construção>::=

<termo de construção>[ {+|-|*|/|**|UNI|DIF|INTER}
<termo de construção>] ...

<prescrição de repetição>::=

PARA <termo de controle> [COM <termo de controle>]

<obter valor booleano>::=

[NÃO] <termo elementar valbool> [ {E|OU|OUEX|IMPL}
[NÃO] <termo elementar valbool>] ...

<end ponto conex>::=

{ [N PX =] <lit nome> / VERDADEIRO } ...

<atributo>::=

<lit nome> [. <lit nome>]

<cri ponto imed>::=

<obter valor booleano>

<cri ponto imed em notação elíptica>

<cri ponto imed em notação elíptica>::=

<lit nome>

<id ponto ::=

ACTRAB |
AUX |
PC [| <lit marca>] |
PX [| <lit marca>] |
: <lit marca>

<descr volume>::=

TECLADO |
VÍDEO |
IMPRESSORA |
DISCO |
FITA |
TELEIMPRESSORA

<dt falha>::=

T - VCOER |
T - VAUTO |
T - VPROT |
F - GRAVE |
F - LEVE |
F - ENTR |
F - SAID

<id regra rep>::=

RRITEM |
RRTUP |
RRLIG - 1 |
RRLIG - 2 |

RRLIG - 3 |
RRTAREL - 1 |
RRTAREL - 2 |
RRTAREL - 3 |
RRTALIG - 1 |
RRTALIG - 2 |
RRTALIG - 3 |
RRCOL - 1
RRCOL - 2

<d pretipo>::=
ACTRAB |
ALIG |
AREL |
COLITENS |
DATA |
HORA |
INT |
REAL |
NUMCAR |
LIG |
TUP |
TAREL |
TALIG |
NOME

<id cálculo>::=
TOTAL |
MEDIA |
MAX |
MIN |
DESVIO

<d ordem>::=
NUMÉRICO |
ALFANUMÉRICO |
DATA-ORD |
HORA-ORD

<d função>::=
    GERENCIAR |
    MODIFICAR |
    CONSULTAR

<literal>::=
    <lit nome> |
    <lit nint> |
    <lit marca> |
    <lit fonte> |
    <lit numcar> |
    <lit nreal> |
    <lit data> |
    <lit hora> |

<lit nome>::=
    <identificador>

<lit nint>::=
    [+ | -] <dígito> ...

<lit marca>::=
    <identificador>

<lit fonte>::=
    {<letra> | <dígito> | <caracter especial>}...

<lit numcar>::=
    " {<letra> | <dígito> | <caracter especial>} ... "

<lit nreal>::=
    { [+] | - } <dígito> ... , <dígito> ...

<lit data>::=
    <dígito><dígito>.<dígito><dígito>.<dígito><dígito><dígito><dígito>

<lit hora>::=
    <dígito><dígito>:<dígito><dígito>[: <dígito><dígito>]

<atributo>::=
    <lit nome>[, <lit nome>]

<obter tarel>::=
    <obter construção>

<obter coleção>::=
    <obter construção>

<obter talig>::=
    <obter construção>

<obter tabela>::=
    <obter construção>

<obter número>::=
    <obter construção>

<d tipo>::=
    <lit nome>

<id execução>::=
    <identificador>

<n acset>::=
    <lit nome>

<n arq>::=
    <lit nome>

<senhac>::=
    <lit nome>

<id volume>::=
    <identificador>

<id proteção>::=
    <identificador>

<id usuário>::=
    <identificador>

<senhus>::=
    <lit nome>

<marca>::=
   <lit nome>

<id texto fonte>::=
    <identificador>

<letra>::=
    A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|
    V|W|Y|X|Z

<dígito>::=
    0|1|2|3|4|5|6|7|8|9

<caracter especial ::=
    %|&|*|(|)|*|-|=|_|;|,|'|"|:|.|?|/|
    | | |$|<|>|[|]|

# ANEXO II

## Sintaxe MICROLOBAN

## (Notação Expressão Regular)

# ANEXO II
## Sintaxe MICROLOBAN

```
01.  trabalho ::=
     (EXECUTAR / COMPILAR) PARA USUÁRIO id '(' id ')'
            (id / INSTRUÇÕES)
     (comando ';')*
     ENCERRAR

02.  comando ::=
     (CRIAR ACSET id (A PARTIR DE ACSET id ATÉ VERSÃO nint)? /
      ABRIR ACSET id (COM id£ ',')?)';'
     (ABRIR VOLUME DE (ENTRADA / SAÍDA / INTERAÇÃO) id (TECLADO /
      VÍDEO / IMPRESSORA / DISCO / FITA / TELEIMPRESSORA) ';')*
     ESTABELECER PROTEÇÃO id (PARA (LER / ALTERAR) SOBRE (id /
      ACTRAB)£ 'U')£ E
   / INCLUIR (expressão (':=' expressão)? / CADA COMPONENTE DE
      termo-conjunto) EM end-ponto
   / EXCLUIR DE end-ponto
   / SUBSTITUIR EM end-ponto POR expressão
   / (CRIAR / ABOLIR) (AREL / ALIG) id
   / FIXAR id EM end-ponto
   / TIRAR id£ ',' (DE end-ponto)?
   / ABANDONAR PROTEÇÃO id
   / ARQUIVAR ACTRAB
   / ABANDONAR ACREC
   / RECONSTRUIR ACSET (EXECUTANDO / DESFAZENDO) nint COMANDOS
   / FECHAR ACSET id?
   / REPRESENTAR SEGUNDO id-regra-rep expressão EM (ÁREA LIVRE)?
     end-campo
   / FECHAR VOLUME DE (ENTRADA / SAÍDA / INTERAÇÃO) id
   / ABOLIR ACSET id
   / CONSIDERAR TRANSAÇÃO '(' (AO ACONTECER (dt-falha ':'
      (CONT NREC / DESCONT (TRANS / TRAB / INST) (REC / NREC))
     )£ ',')? (inst-transação£ ';')? ')'
   / FAZER PARA termo-controle (COM termo-controle)?
      '(' inst-fazer-para£ ';' ')'
```

```
   /  FAZER SE exp-booleana '(' inst-fazer-para£ ';' ')'
      SENÃO '(' inst-fazer-para£ ';' ')'

03.   end-campo ::=
      CC ('|' id)?
   /  VOLUME id? '.' CAMPO '(' ')'

04.   end-ponto ::=
      ((ACTRAB / AUX / (PC / PX) ('|' id)? / ':' id) '.')?
      (('[' '=' id ']')? (exp-booleana / id))£ '.'
   /  (ACTRAB / AUX / (PC / PX)('|' id)? / ':' id)

05.   exp-booleana ::=
      (NÃO? prim-booleano)£ (E / OU / OUEX / IMPL)

06.   prim-booleano ::=
      PARA TODO end-ponto '(' exp-booleana ')'
   /  EXIST end-ponto '(' exp-booleana? ')'
   /  C end-ponto ELEM termo-conjunto
   /  (C / N) end-ponto ('=' / '<' / '>' / '<=' / '>=' / '<>' )
      (expressão / id) (EM ORDEM d-ordem)?
   /  VERDADEIRO
   /  '(' exp-booleana ')'

07.   expressão ::=
      termo£ ('+' / '-' / '*' / '/' / '**' / UNI / DIF / INTER)

08.   termo ::=
      INTR SEGUNDO id end-campo
   /  constante
   /  (N / C / COLEC / CONT / M) end-ponto
   /  CALC id-calc SOBRE end-ponto
   /  ALARG expressão DE (id ('.' id)? '(' expressão ')' )£ ','
   /  (CARD / DESEGRUP) expressão
   /  (ESTREIT expressão DE / AGRUP expressão POR)(id ('.' id)?)£
   /  RENOM expressão SUBST (id POR id)£ ','
   /  (JUNT / LIGA) expressão COM expressão EXCL? '(' (C id
      ('.' id)? ('=' / '<' / '>' / '<=' / '>=' / '<>' )
      C id ('.' id)?)? ')'
```

```
    /  VERBETE verb
    /  (VAZ / HORA-CORR / DATA-CORR)
    /  '(' expressão ')'

09. termo-conjunto ::=
     d-pretipo
    /  id
    /  '{' constante£ ';' '}'
    /  '{' (VERBETE verb)£ ';' '}'
    /  expressão

10. termo-controle ::=
     end-campo ('[' '=' id ']')?
     end-ponto ('[' '=' id ']')?  (EM ORDEM POR
     (C id ('.' id)?  (ASC / DESC))£ ',')?

11. verb ::=
     USUÁRIO id '(' id ')' COM id£ ','
    /  SENHAC id (PERMITE (GERENCIAR / MODIFICAR / CONSULTAR)
     (ACTRAB / id))£ ','
    /  FONTE id ':' texto '#'
    /  id ':' TIPO DE d-pretipo TAL QUE
     (COMPOS (id
            /(id '->'(id / DATA / HORA / INT / REAL))£ ','
            / POR nint CARACTERES)
      (CONEX (( COLEC end-ponto-conex SUBCONJ COLEC
                 end-ponto-conex)£ ','
            /CHAVE PRIMÁRIA id E CARD C OCOR '<=' nint))?
     /  C OCOR (ELEM '{' constante£ ';' '}'
              /('=' / '<' / '>' / '<=' / '>=' / '<>') constante
               (E C OCOR ('=' / '<' / '>' / '<=' / '>=' / '<>')
               constante)?)
     )

12. end-ponto-conex ::=
     ((N PX '=')? id / VERDADEIRO)£ '.'
```

13. inst-transação ::=
INCLUIR (expressão (':=' expressão)? / CADA COMPONENTE DE termo-conjunto) EM end-ponto
/ EXCLUIR DE end-ponto
/ SUBSTITUIR EM end-ponto POR expressão
/ FIXAR id EM end-ponto
/ TIRAR id£ ',' (DE end-ponto)?
/ FAZER PARA termo-controle (COM termo-controle)? '(' inst-transação£ ';' ')'
/ FAZER SE exp-boolenana '(' inst-transação£ ';' ')' SENÃO '(' inst-transação£ ';' ')'
/ REPRESENTAR SEGUNDO id-regra-rep expressão em (ÁREA LIVRE)? end-campo

14. inst-fazer-para ::=
INCLUIR(expressão(':=' expressão)? / CADA COMPONENTE DE termo-conjunto) EM end-ponto
/ EXCLUIR DE end-ponto
/ SUBSTITUIR EM end-ponto POR expressão
/ FIXAR id EM end-ponto
/ TIRAR id£ ',' (DE end-ponto)?
/ FAZER PARA termo-controle (COM termo-controle)? '(' inst-fazer-para£ ';' ')'
/ FAZER SE exp-booleana '(' inst-fazer-para£ ';' ')' SENÃO '(' inst-fazer-para£ ';' ')'
/ CONSIDERAR TRANSAÇÃO '(' (AO ACONTECER (dt-falha ':' (CONT NREC / DESCONT (TRANS / TRAB / INST) (REC / NREC)))£ ',')? (inst-transação£ ';')? ')'
/ REPRESENTAR SEGUNDO id-regra-rep expressão EM (ÁREA LIVRE)? end-campo

15. d-ordem ::=
NUMÉRICO
/ ALFANUMÉRICO
/ DATA-ORD
/ HORA-ORD

16. id-regra-rep ::=
    RRITEM
  / RRTUP
  / RRLIG-1
  / RRLIG-2
  / RRLIG-3
  / RRTAREL-1
  / RRTAREL-2
  / RRTAREL-3
  / RRTALIG-1
  / RRTALIG-2
  / RRTALIG-3
  / RRCOL-1
  / RRCOL-2

17. id-calc ::=
    TOTAL
  / MÉDIA
  / MAX
  / MIN
  / DESVIO

18. dt-falha ::=
    T-VCOER
  / T-VAUTO
  / T-VPROT
  / F-GRAVE
  / F-LEVE
  / F-ENTR
  / F-SAID

19. d-pretipo ::=
    ACTRAB
  / ALIG
  / AREL
  / COLITENS
  / DATA
  / HORA
  / INT

/ REAL
/ NUMCAR
/ LIG
/ TUP
/ TAREL
/ TALIG
/ NOME
/ VALBOOL

20. constante ::=
constante-nint
/ constante-nreal
/ constante-numcar
/ constante-data
/ constante-hora

21. constante-nint ::=
('+'? / '-') dígito+

22. constante-nreal ::=
('+'? / '-') dígito+ ',' dígito+

23. constante-data ::=
dígito dígito '.' dígito dígito '.' dígito dígito dígito dígito

24. constante-hora ::=
dígito dígito ':' dígito dígito (':' dígito dígito)?

25. constante-numcar ::=
' " ' (letra / dígito /caracter-especial)+80 ' " '

26. texto ::=
(letra /dígito /caracter-especial)+

27. letra ::=
A / B / C / D / E / F / G / H / I / J / K / L / M / N / O /
P / Q / R / S / T / U / V / X / Y / W / Z

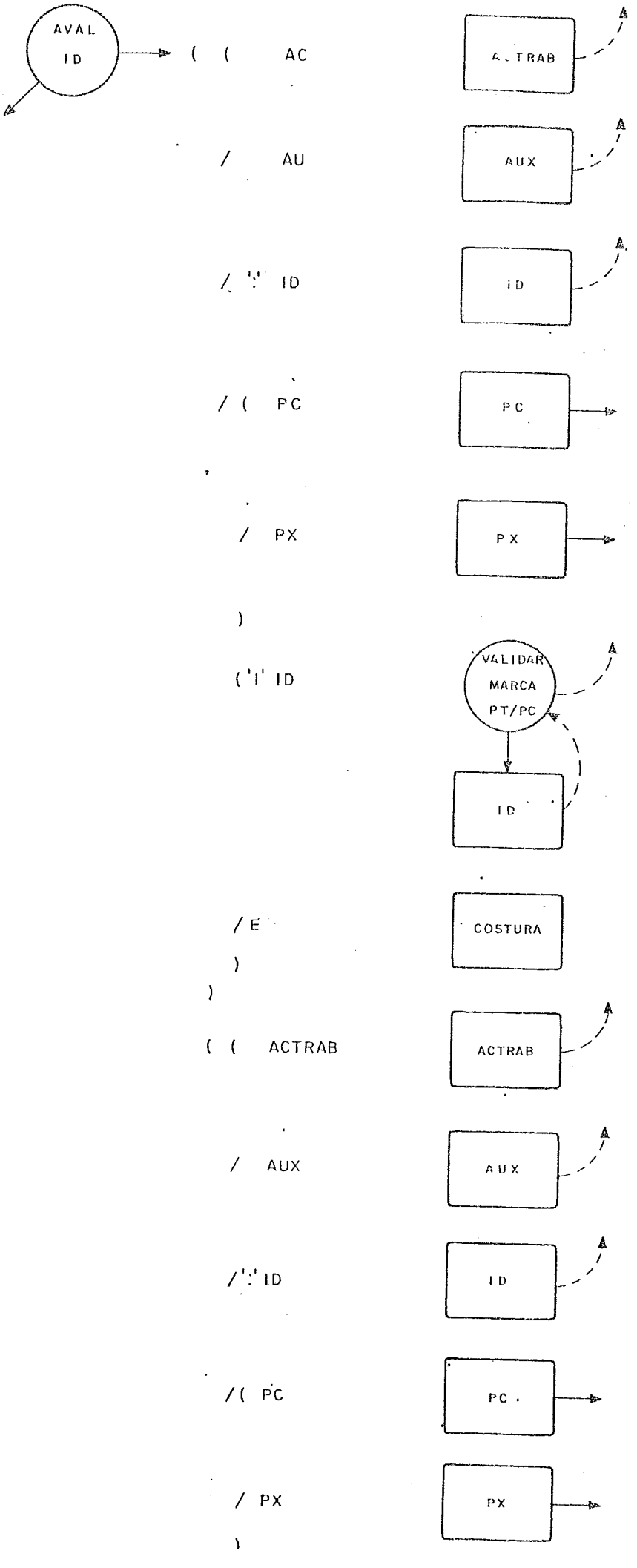
28. dígito ::=
0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9

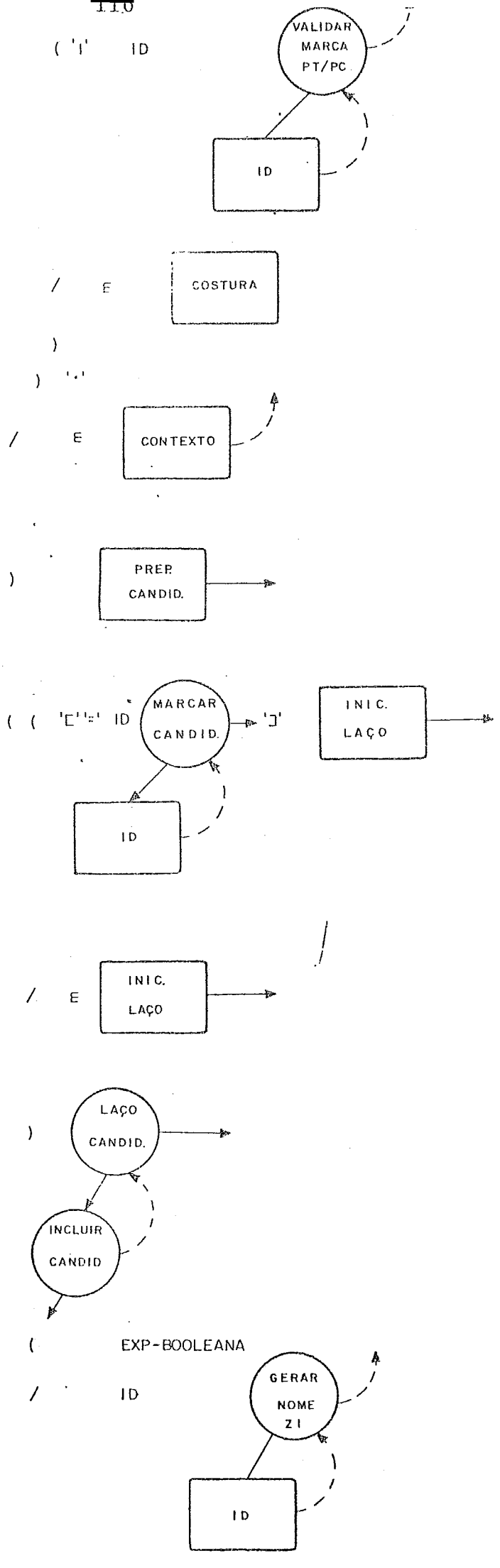
29. caracter-especial ::=
% / & / * / ( / ) / + / - / = / — / ; / , / ' / " / : / . /
? / / / | / $ / < / > / [ / ] / @

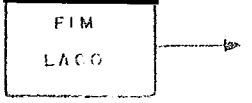
# ANEXO III

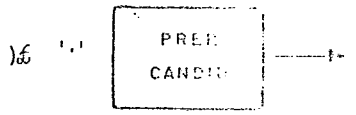
## Especificação do Código Intermediário de MICROLOBAN

1. END-PONTO ::=


AVAL
ID
( ( AC
ACTRAB
/ AU
AUX
/ ':' ID
ID
/ ( PC
PC
/ PX
PX
)
('I' ID
VALIDAR
MARCA
PT/PC
ID
/ E
COSTURA
)
)
( ( ACTRAB
ACTRAB
/ AUX
AUX
/ ':' ID
ID
/ ( PC
PC
/ PX
PX
)

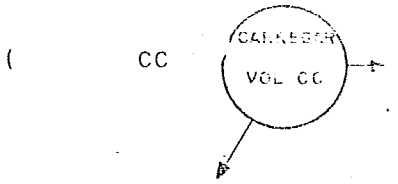

( 'I' ID
VALIDAR MARCA PT/PC
ID
/ E
COSTURA
)
) '.'
/ E
CONTEXTO
)
PREP. CANDID.
( ( 'E''=' ID
MARCAR CANDID.
']'
INIC. LAÇO
ID
/ E
INIC. LAÇO
)
LAÇO CANDID.
INCLUIR CANDID
( EXP-BOOLEANA
/ ID
GERAR NOME Z1
ID

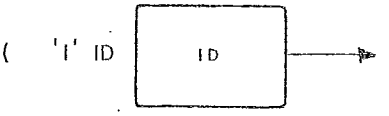


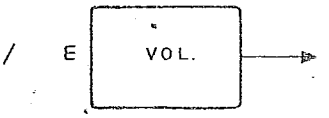





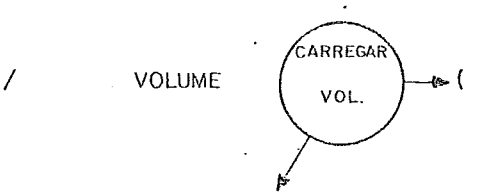
2. END-CAMPO::=

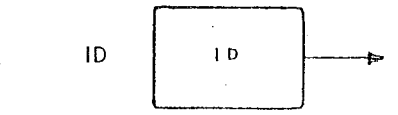


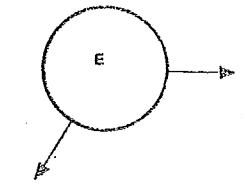




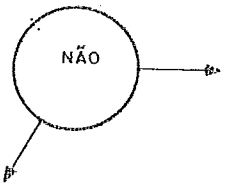





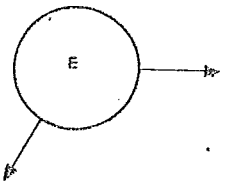

'.' CAMPO '(' ')'



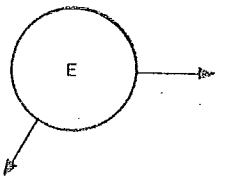

( ( ( NÃO



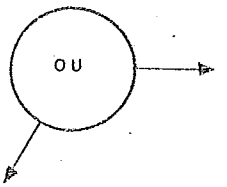

/ E



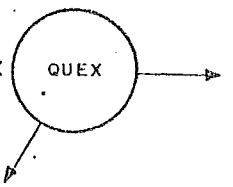

( E



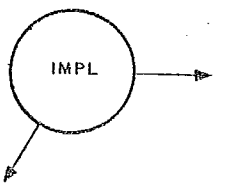

/ OU




/ QUEX



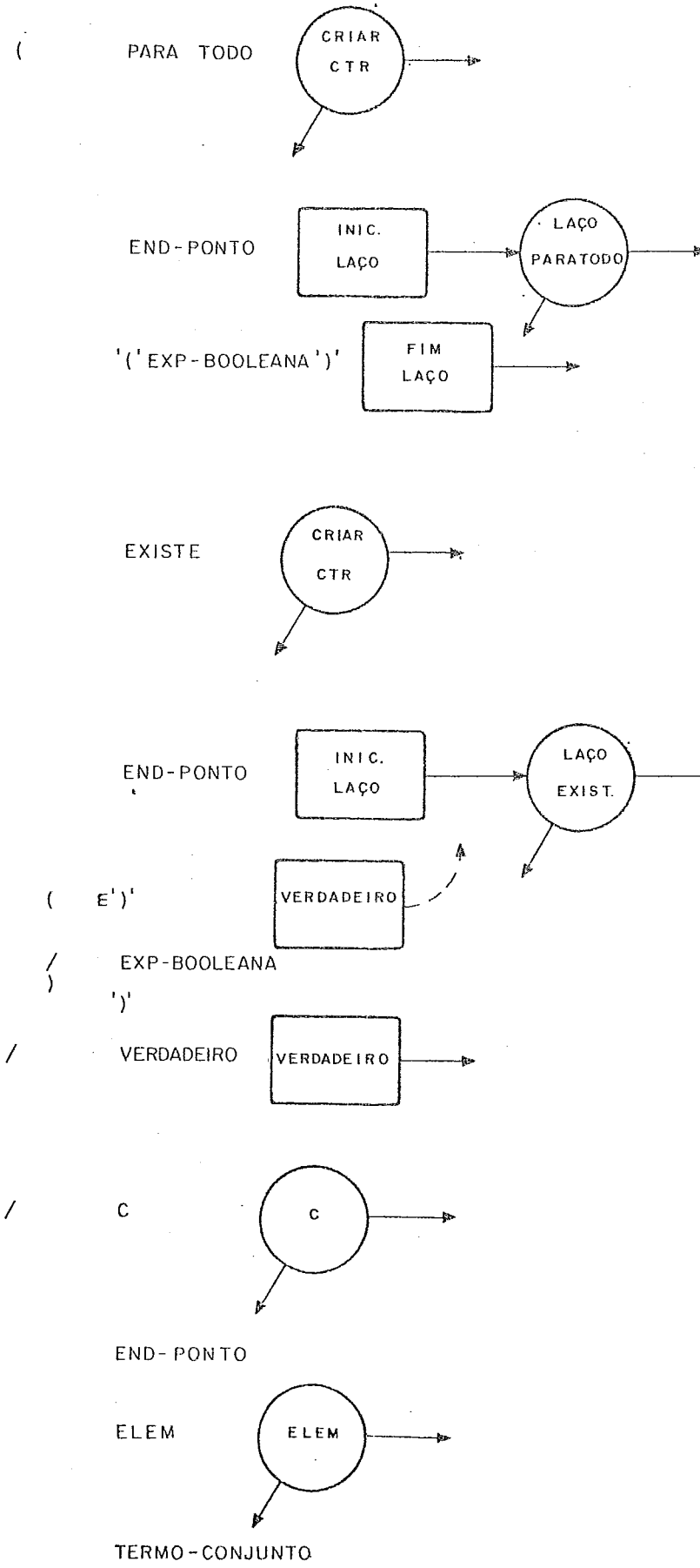

/ IMPL

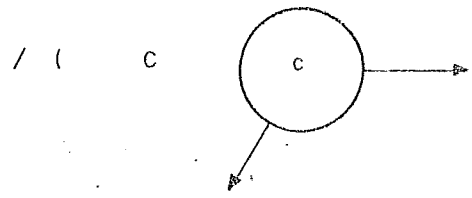


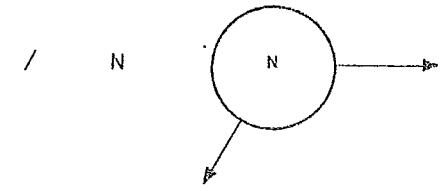

)

)

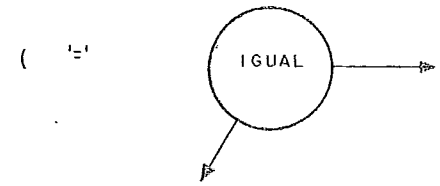

4. PRIM - BOOLEANO ::=

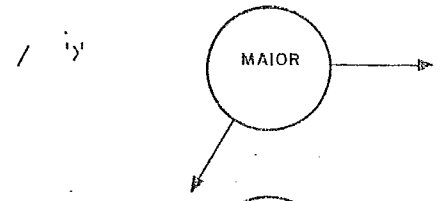


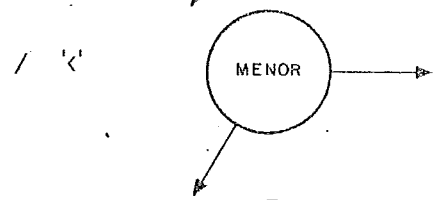

/ ( C
C

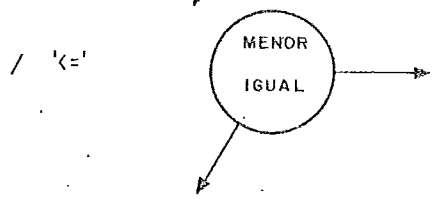

/ N
N

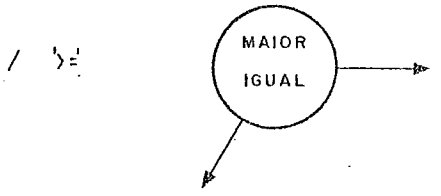
). END-PONTO

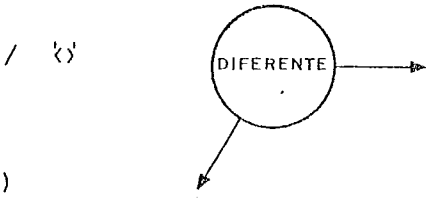

( '='
IGUAL


/ '>'
MAIOR

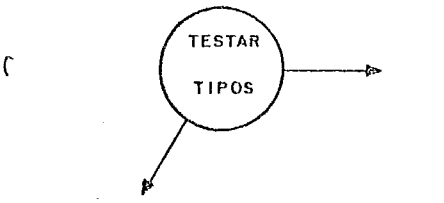

/ '<'
MENOR

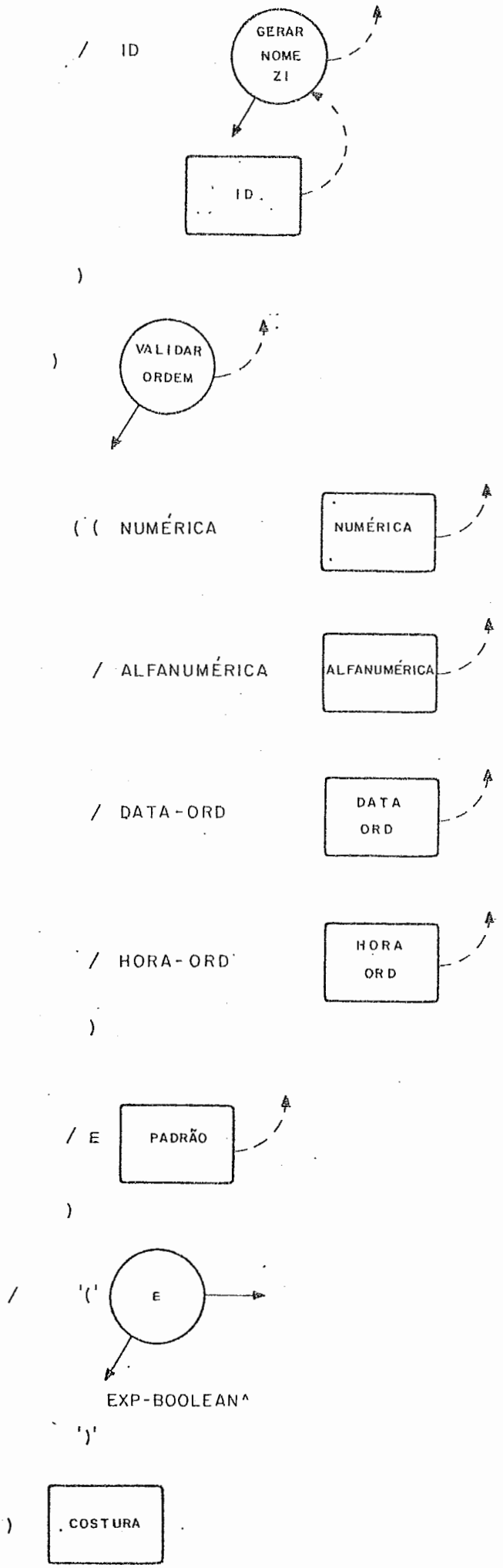

/ '<='
MENOR
IGUAL


/ '>='
MAIOR
IGUAL


/ '<>'
DIFERENTE

)


(
TESTAR
TIPOS


/ ID
GERAR NOME ZI
ID
)
)
VALIDAR ORDEM
( ( NUMÉRICA
NUMÉRICA
/ ALFANUMÉRICA
ALFANUMÉRICA
/ DATA-ORD
DATA ORD
/ HORA-ORD
HORA ORD
)
/ E
PADRÃO
)
/ '('
E
EXP-BOOLEAN^
')'
)
COSTURA

# ANEXO IV

## Instruções Intermediárias LIBAN

AVAL ID

RESUMO - Avaliar o identificador de ponto

PROCEDIMENTO:

- Verificar se registrador STATUS possui conteúdo vazio; se sim faça:
  - Acessar o conteúdo do tipo da pilha de trabalho. Dependendo deste, temos:
    1. ACTRAB

       Tratamento:

       - Alterar o conteúdo do indicador de nível corrente (IND NV) para conter a indicação que o contexto de referência utilizado é o acervo de trabalho (# 00).
       - Formatar e incluir uma ligação em TABAVL, cuja tabela de ligados será a construção vazia, e o ligante conterá as seguintes informações:

         IND NÍVEL ← conteúdo do registrador IND NV.

         NÍVEL ← conteúdo do registrador NV EMB.

         IND ARQ ← endereço da descrição do acervo de trabalho no Diretório de Arquivos e Tabelas (DIRARQTAB).

         MARCA ← conteúdo vazio.

         TIPO ← pretipo padrão ACTRAB.

         DESLOCAMENTO ← conteúdo vazio.

         TAMANHO ← conteúdo vazio.

         TIPO C/E ← Tipo Escolhido (E).

         PT EXAME ← conteúdo vazio.

    2. AUX

       Tratamento:

- Alterar o conteúdo do registrador IND NV para conter a indicação do contexto de referência utilizado, isto é, AUX (#10).
- Formata e inclui uma ligação em TABAVL, cuja tabela de ligados será a construção vazia e o ligante conterá as seguintes informações:

  IND NÍVEL ← conteúdo do registrador IND NV.

  NÍVEL ← conteúdo do registrador NV EMB.

  IND ARQ ← endereço da descrição da construção auxiliar em DIRARQTAB.

  MARCA ← conteúdo vazio.

  TIPO ← pretipo padrão AUX.

  DESLOCAMENTO ← conteúdo vazio.

  TAMANHO ← conteúdo vazio.

  TIPO C/E ← Tipo Escolhido (E).

  PT EXAME ← conteúdo vazio.

3. ID (Identificador de marca fixada através de uma instrução <fixar marca>)

Tratamento:

- Acessar a tabela de Marcas(TABMARCA) com chave formatada com o identificador de marca forneçido. Caso esta marca não exista, enviar mensagem de erro ao usuário e armazenar código de erro no registrador STATUS.
- Caso contrário, marca existente, verificar se a mesma foi estabelecida através de uma instrução <fixar marca>. Se marca válida, armazenar informações extraídas dos atributos do ligante de TABMARCA nos seguintes registradores da máquina virtual MICROLOBAN:

IND NV ← conteúdo do atributo IND NV.

ARQ ← conteúdo do atributo IND ARQ.

TIPO ← conteúdo do atributo TIPO.

TAMANHO ← conteúdo do atributo TAM.

DESLOC ← conteúdo do atributo DESLOC.

Formatar e incluir uma ligação em TABAVL cujo ligante será composto das seguintes informações:

IND NÍVEL ← conteúdo do registrador IND NV.

NÍVEL ← conteúdo do registrador NV EMB.

IND ARQ ← conteúdo do registrador ARQ.

MARCA ← conteúdo vazio.

TIPO ← conteúdo do registrador TIPO.

DESLOCAMENTO ← conteúdo do registrador DESLOC.

TAMANHO ← conteúdo do registrador TAMANHO.

TIPO C/E ← Tipo Escolhido (E).

PT EXAME ← conteúdo vazio.

A tabela de ligados será composta da tabela de ligados da referida ligação em TABMARCA.

- Caso a marca seja inválida armazena código de erro no registrador STATUS e envia mensagen de erro ao usuário.

4. PC (Ponto Corrente)

Tratamento:

- Acessar a TABMARCA levando em consideração as duas possibilidades de acesso a esta tabela, isto é:

  1. PC possui marca válida

     Acesso feito através do nome da marca fornecido e validado.

  2. PC sem marca

Acesso feito por meio do conteúdo do topo da pilha de profundidade e conteúdo da marca vazio. Caso não exista ligação correspondente, armazenar código de erro no registrador STATUS e enviar mensagem de erro ao usuário.

- Caso o conteúdo do registrador STATUS seja vazio, ligação acessada em TABMARCA, armazenar informações extraídas dos atributos do ligante nos seguintes registradores da máquina virtual:

IND NV ← conteúdo do atributo IND NV.

ARQ ← conteúdo do atributo IND ARQ.

TIPO ← conteúdo do atributo TIPO.

TAMANHO ← conteúdo do atributo TAM.

DESLOC ← conteúdo do atributo DESLOC.

PC ← conteúdo do atributo PC.

Formata e inclui uma ligação em TABAVL cujo ligante será composto das informações extraídas da TABMARCA:

IND NÍVEL ← conteúdo do registrador IND NV.

NÍVEL ← conteúdo do registrador NV EMB.

IND ARQ ← conteúdo do registrador ARQ.

MARCA ← conteúdo vazio.

TIPO ← conteúdo do registrador TIPO.

DESLOCAMENTO ← conteúdo do registrador DESLOC.

TAMANHO ← conteúdo do registrador TAMANHO.

TIPO C/E ← Tipo Escolhido (E).

PT EXAME ← conteúdo vazio.

A Tabela de ligados será composta do conteúdo do registrador PC.

5. PX (Ponto EXaminando)

. Tratamento:

- Acessar a Tabela de Avaliação de endereço de pontos (TABAVL), considerando as seguintes possibilidades:

1. PX com marca válida

Acessa a TABAVL sequencialmente para encontrar uma tupla ligante que tenha o conteúdo do atributo MARCA igual a marca fornecida.(pilha de trabalho)

2. PX sem marca

Acessar a TABAVL com chave formatada com o conteúdo do registrador NV EMB e tipo do ponto a ser acessado como ponto candidato. Caso não exista ligação com chave correspondente, armazenar código de erro no registrador STATUS e enviar mensagem ao usuário.

- Caso ligação acessada em TABAVL (STATUS = vazio), estrair informações do seu ligante para ser armazenada nos seguintes registradores:

IND NV ← conteúdo do atributo IND NÍVEL.

ARQ ← conteúdo do atributo IND ARQ.

TIPO ← conteúdo do atributo TIPO.

TAMANHO ← conteúdo do atributo TAMANHO.

DESLOC ← conteúdo do atributo DESLOCAMENTO.

PX ← conteúdo do atributo PT EXAME.

Formatar e incluir uma ligação em TABAVL cujo ligante será composto das seguintes informações:

IND NÍVEL ← conteúdo do registrador IND NV

NÍVEL ← conteúdo do registrador NV EMB.

IND ARQ ← conteúdo do registrador ARQ.

MARCA ← conteúdo vazio.

TIPO ← conteúdo do registrador TIPO.

TAMANHO ← conteúdo do registrador TAMANHO.

DESLOCAMENTO ← conteúdo do registrador DESLOC.

TIPO C/E ← Tipo Escolhido (E).

PT EXAME ← conteúdo vazio.

O conteúdo da tabela de ligados dependerá do conteúdo do registrador.IND NV. Se esse registrador indicar que o nível de endereçamento é diferente de componente de coleção de tuplas, ligações ou itens, implica que a tabela de ligados será a construção vazia. Caso contrário, incluir o conteúdo do registrador PX como único ligado da referida ligação.

6. CONTEXTO

Tratamento:

- Acessar TABAVL com chave formatada com o conteúdo do registrador NV EMB menos um e tipo do ponto como sendo Escolhido.
- Caso exista ligação correspondente, armazenar os conteúdos de alguns atributos do seu ligante nos registradores relacionados abaixo:

  IND NV ← conteúdo do atributo IND NÍVEL.

  ARQ ← conteúdo do atributo ARQ.

  TIPO ← conteúdo do atributo TIPO.

  TAMANHO ← conteúdo do atributo TAMANHO.

  DESLOCAMENTO ← conteúdo do atributo DESLOC.

  PX ← conteúdo do atributo PT EXAME.

  Caso contrário, não existe ligação correspondente em TABAVL, esses registradores terão os seguintes conteúdos:

IND NV ← nível de acervo de trabalho (# 00).

ARQ ← Endereço da descrição de ACTRAB no DIRARQTAB.

TIPO ← Tipo padrão ACTRAB.

TAMANHO ← conteúdo vazio.

DESLOC ← conteúdo vazio.

PX ← conteúdo vazio.

Formata e inclui uma ligação em TABAVL, cuja tupla ligante será composta da seguinte forma:

IND NÍVEL ← conteúdo do registrador IND NV.

NÍVEL ← conteúdo do registrador NV EMB.

IND ARQ ← conteúdo do registrador ARQ.

MARCA ← conteúdo vazio.

TIPO ← conteúdo do registrador TIPO.

TAMANHO ← conteúdo do registrador-TAMANHO.

DESLOCAMENTO ← conteúdo do registrador DESLOC.

TIPO C/E ← Tipo Escolhido (E).

PT EXAME ← conteúdo vazio.

O conteúdo da tabela de ligados dependerá do conteúdo do registrador IND NV. Se o indicador de nível corrente especificar nível de tupla, de ligação, de item composto, de item simples ou de ligante, será incluido como ligado o conteúdo do registrador PX. Caso contrário esta ligação de TABAVL terá a construção vazia como tabela de ligados.

- Desempilhar topo da pilha de trabalho.

ACTRAB

RESUMO - Empilhar palavra chave ACTRAB

PROCEDIMENTO:

- Armazenar a palavra chave ACTRAB no topo da pilha de trabalho.

AUX

RESUMO - Empilhar palavra chave AUX

PROCEDIMENTO:

- Armazenar a palavra chave AUX no topo da pilha de trabalho.

ID

RESUMO - Empilhar identificador fornecido

PROCEDIMENTO:

- Armazenar o identificador fornecido no topo da pilha de trabalho.

PC

RESUMO - Empilhar palavra chave PC

PROCEDIMENTO:

- Armazenar a palavra chave PC no topo da pilha do trabalho.

PX

RESUMO - Empilhar palavra chave PX

PROCEDIMENTO:

- Armazenar a palavra chave PX no topo da pilha de trabalho.

CONTEXTO

RESUMO - Empilhar palavra chave CONTEXTO

PROCEDIMENTO:

- Armazenar a palavra chave CONTEXTO no topo da pilha de trabalho.

VALIDAR MARCA PT/PC

RESUMO - Validar o identificador fornecido como nome de marca sobre o conjunto controlador de pontos ou de campos.

PROCEDIMENTO:

- Acessar conteúdo do (Topo-1) da pilha de trabalho. De acordo com o conteúdo acessado temos dois casos a considerar:

1. PC

   Tratamento:

   - Acessar a TABMARCA com chave formatada tendo como conteúdo o nome da marca fornecido. Caso não exista uma ligação acessada pela chave, armazenar código de erro no registrador STATUS e enviarmensagem de erro ao usuário.
   - Caso contrário, marca existente, verificar se o identificador fornecido é um nome de marca sobre um conjunto controlador de pontos. Se não, armazenar código de erro no registrador STATUS e enviar mensagem de erro ao usuário.

2. PX

   Tratamento:

   - Acessar a TABAVL sequencialmente pesquisando todos os ligantes até encontrar o conteúdo do atributo MARCA igual ao nome da marca fornecido.

- Caso não exista ligação correspondente, armazenar código de erro no registrador STATUS e enviar mensagem de erro ao usuário.

- Verificar se o conteúdo do registrador STATUS é igual a vazio, trocar os conteúdos das posições topo e (topo-1) da pilha de trabalho.

PREP CANDID

RESUMO - Preparar o conjunto de pontos candidatos a ser validado segundo o critério fornecido.

PROCEDIMENTO:

- Verificar se o conteúdo do registrador IND NV se refere a nível de item simples (# nB) ou de construção sem tipo (# 1F). Caso afirmativo, armazenar código de erro no registrador STATUS e enviar mensagem de erro ao usuário. Além disto, acessar TABAVL com chave formatada com o conteúdo do registrador NV EMB e indicação de ligação do tipo escolhida (pontos escolhidos). Se existe, excluir esta ligação da tabela de Avaliação de endereço de ponto.

- Caso contrário, indicador de nível diferente de # nB ou # 1F, acessar a TABAVL com chave formada com o conteúdo do registrador NV EMB e indicação de conjunto de pontos escolhidos (E). Se não existe ligação correspondente, armazenar código de erro no registrador STATUS e enviar mensagem de erro ao usuário.

  Caso exista conjunto de pontos escolhidos no nível anterior, faça:

  - Alterar o ligante, da ligação acessada em TABAVL, para con-

ter a indicação de conjunto de pontos candidatos (Tipo C/E = C).

- Dependendo do conteúdo do atributo IND NÍVEL do ligante lido em TABAVL, temos:
  - #00(ACTRAB), #10(AUX), #01(FOLHA), #02(FICHA), #03(arquivo, #n7(tupla), #n8(ligação), #nC(componente de coleção de tupla), #nD(componente de coleção de ligação).
    Tratamento:
    - Alterar fisicamente o ligante modificado em TABAVL.
  - Nível de TR, ou coleção de tuplas ou tabelas de ligados (T): #n4
    Tratamento:
    - Alterar o conteúdo do atributo IND NÍVEL do ligante acessado em TABAVL para indicar nível de componente de coleção de tuplas (#nC).
    - Acessar a Tabela de Consistência de Coleção (TABCONSCOL) com chave formatada com o conteúdo do atributo TIPO do ligante acessado em TABAVL. Alterar o conteúdo do atributo TIPO deste ligante com o conteúdo do atributo TERM TIPO da tupla lida em TABCONSCOL.
    - Armazenar fisicamente o ligante modificado em TABAVL.
    - Formatar e incluir como ligados, os endereços das tuplas que compõem a tabela em questão.
    - Acessar o primeiro ligado da tabela de ligados, armazenada anteriormente, e atribuir o seu valor ao atributo PT EXAME do seu ligante (TABAVL).

- Nível de TL ou coleção de ligações: #n5

  Tratamento:

  - Alterar o conteúdo do atributo IND NÍVEL do ligante acessado em TABAVL para conter a indicação de componente de coleção de ligação (#nD).
  - Acessar a TABCONSCOL com a chave formatada com o conteúdo do atributo TIPO do ligante acessado em TABAVL. Alterar o conteúdo deste atributo para ter o conteúdo do atributo TERMTIPO da tupla acessada em TABCONSCOL.
  - Armazenar fisicamente o ligante alterado em TABAVL.
  - Formatar e incluir como elementos da tabela de ligados da referente ligação, os endereços das ligações que compõem a tabela endereçada.
  - Acessar o ligante armazenado anteriormente, para alterar o conteúdo do atributo PT EXAME para conter o do primeiro ligado da ligação em questão (TABAVL).

- Nível de seleção de itens: #n6.

  Tratamento:

  - Alterar o conteúdo do atributo IND NÍVEL do ligante acessado em TABAVL para conter a indicação de componente de coleção de itens (#nE).
  - Acessar a TABCONSCOL com chave formatada com o conteúdo do atributo TIPO do ligante da ligação acessada em TABAVL. Alterar o conteúdo desse atributo para ter o conteúdo do atributo TERMTIPO da tupla acessada em TABCONSCOL.
  - Armazenar fisicamente o ligante alterado em TABAVL.
  - Formatar e incluir, como ligados dessa ligação em TABAVL, os endereços dos itens que compõe a colitens.

- Acessar o ligante em TABVL e alterar o conteúdo do atributo PT EXAME para ter o valor do primeiro ligado desta ligação.

- Nível de atributo composto (#nA), Nível de ligante (#n9).

Tratamento:

- Alterar o conteúdo do atributo IND NÍVEL do ligante acessado em TABAVL para conter a indicação de nível de tupla (#n7).
- Armazenar fisicamente o ligante alterado em TABAVL.

MARCAR CANDID

RESUMO - Marcar o conjunto de pontos candidatos criado para avaliação de determinado critério.

PROCEDIMENTO:

- Acessar a TABAVL sequencialmente, comparando o conteúdo do atributo MARCA do ligante com o do topo da pilha de trabalho. Caso exista a marca fornecida, armazenar código de erro no registrador STATUS, enviar mensagem de erro ao usuário e excluir ligação de TABAVL cuja chave é composta do conteúdo do registrador NV EMB e indicação de ligação candidata (TIPO C/E = C).
- Caso o nome de marca não exista em nenhuma ligação de TABAVL, acessar TABAVL com chave formatada com o conteúdo do registrador NV EMB e indicação de conjunto de pontos candidatos. Alterar o conteúdo do atributo MARCA do ligante acessado para registrar o nome de marca existente no topo da pilha de trabalha.

LAÇO CANDID

RESUMO - Executar laço sobre os pontos candidatos.

PROCEDIMENTO:

- Acessar a TABAVL com chave formatada com conteúdo do registrador NV EMB e indicação de conjunto de pontos candidatos (TIPO C/E = C).
- Dependendo do conteúdo do atributo IND NÍVEL do ligante acessado em TABAVL, temos:
  1. Nível de nominação:
     - Excluir a referida ligação de TABAVL acessada anteriormente.
  2. Nível de coleção:
     - Acessar a tabela de ligados, da ligação acessada, sequencialmente comparando o conteúdo do atributo PT EXAME do ligante com o conteúdo do ligado. Caso seja o último ligado da tabela, exclui esta ligação de TABAVL.
     - Caso contrário, ainda existem pontos candidatos a serem avaliados, acessar o próximo ligado na tabela de ligados colocando o seu conteúdo no atributo PT EXAME do ligante desta ligação.

GERAR NOME ZI

RESUMO - Gerar construção de pretipo NOME na Zona Intermediária

PROCEDIMENTO:

- Acessar TABAVL com chave formatada com o conteúdo do registrador NV EMB e indicação de conjunto de pontos candidatos (TIPO C/E = C).

- Verificar se o conteúdo do atributo IND NÍVEL do ligante lido em TABAVL indica nível de componente de coleção, isto é, #nC, #nD ou #nE. Caso afirmativo, enviar mensagem de erro do erro ao usuário, armazenar código de erro no registrador STATUS e excluir de TABAVL a ligação acessada anteriormente.
- Caso contrário, não é componente de coleção, fazer:
  - Armazenar o valor booleano VERDADEIRO no topo da pilha de valbool da máquina virtual MICROLOBAN.
  - Armazenar, no topo da pilha da ZI da máquina virtual, as seguintes informações:

    CH ← conteúdo vazio

    NOME ← conteúdo vazio

    TIPO ← conteúdo do topo da pilha de trabalho

    PRETIPO ← pretipo padrão NOME
  - Desempilhar conteúdo do topo da pilha de trabalho.

INCLUIR CANDID

RESUMO - Incluir ponto candidato como ponto escolhido em TABAVL.

PROCEDIMENTO:

- Acessar o conteúdo do topo da pilha da Z.I. Verificar se existe construção cujo pretipo seja igual ao pretipo padrão NOME; se afirmativo, faça:

  Tratamento:
  - Acessar TABAVL com chave formatada com o conteúdo do registrador NV EMB e indicação de conjunto de pontos candidatos (TIPO C/E = C). Armazenar os conteúdos dos seguintes atributos do ligante acessado nos seguintes registradores:

IND NV ← conteúdo do atributo IND NÍVEL.

ARQ ← conteúdo do atributo IND ARQ.

TIPO ← conteúdo do atributo TIPO.

TAMANHO ← conteúdo do atributo TAMANHO.

DESLOC ← conteúdo do atributo DESLOCAMENTO.

- Verificar se o conteúdo do topo da pilha de valbool é VERDADEIRO. Caso não o seja, desempilhar conteúdo do topo da pilha de valbool.
- Caso contrário, executar um dos procedimentos abaixo dependendo do conteúdo do registrador IND NV:

1. Nível de Acervo de Trabalho (#00)

   PROCEDIMENTO:

   - Dependendo do conteúdo do campo TIPO do topo da Pilha da ZI, temos:

     FOLHA:

     - Acessar DIRARQTAB com chave formatada com CV-COERÊNCIA. Verificar se esta construção esta sob proteção. Caso não tenha sido protegido, enviar mensagem de erro ao usuário, armazenar código de erro no registrador STATUS.
     - Caso a folha esteja sob proteção, formatar e incluir ligação em TABAVL com as seguintes informações:

       IND NÍVEL ← indicador de nível #01.

       NÍVEL ← conteúdo do registrador NV EMB.

       IND ARQ ← conteúdo do registrador ARQ.

       MARCA ← conteúdo vazio.

       TIPO ← pretipo padrão FOLHA.

       DESLOCAMENTO ← conteúdo vazio.

TAMANHO ← conteúdo vazio.

TIPO C/E ← tipo escolhido (E).

PT EXAME ← conteúdo vazio.

Tabela de Ligados: construção vazia

FICHA

- Acessar DIRARQTAB com chave formatada com a palavra reservada DIRARQTAB. Verificar se a construção FICHA componente do acervo de trabalho está sob proteção. Se não, armazenar código de erro no registrador STATUS, enviar mensagem de erro ao usuário.
- Caso a ficha do ACTRAB esteja protegida, formatar e incluir uma ligação em TABAVL com as seguintes informações:

Ligante:

IND NÍVEL ← indicador de nível #02.

NÍVEL ← conteúdo do registrador NV EMB.

IND ARQ ← conteúdo do registrador ARQ.

MARCA ← conteúdo vazio.

TIPO ← pretipo padrão FICHA.

DESLOCAMENTO ← deslocamento do atributo FICHA dentro da tupla que compõe DIRARQTAB.

TAMANHO ← tamanho do atributo FICHA dentro da tupla que compõe DIRARQTAB.

TIPO C/E ← tipo escolhido (E).

PT EXAME ← conteúdo vazio.

Tabela de ligados: construção vazia.

ID (nome de arquivo)

- Acessar a Tabela de Composição de Nominação (TABCOMP NOM) com chave formatada com a palavra reservada AC-TRAB, para acessar a definição conceitual do acervo de trabalho.
- Acessar a tabela de ligados da ligação acessada em TABCOMPNOM com chavè formatada com o conteúdo do campo TIPO da pilha de construções da ZI. Caso não exista ligados com chave correspondente, armazenar código de erro no registrador STATUS, enviar mensagem de erro ao usuário.
- Caso contrário, nome fornecido é nome de componente imediato de ACTRAB, armazenar o conteúdo do atributo TERMTIPO do ligado lido em TABÇOMPNOM no registrador TIPO. Acessar DIRARQTAB com chave formatada com o conteúdo do campo TIPO do topo da pilha de construções da ZI; se não existe tupla com a chave correspondente, enviar mensagem de erro ao usuário, armazenar código de erro no registrador STATUS.
- Se o arquivo estiver criado, verificar se o mesmo foi protegido. Se não armazenar código de erro no registrador STATUS, enviar mensagem de erro ao usuário.
- Caso arquivo protegido, formatar e incluir uma ligação em TABAVL contendo as seguintes informações:

  Ligante:

  IND NÍVEL ← indicador de nível #03.

  NÍVEL ← conteúdo do registrador NV EMB.

IND ARQ ← Endereço da descrição do arquivo em DIRARQTAB.

MARCA ← conteúdo vazio.

TIPO ← conteúdo do registrador TIPO.

DESLOCAMENTO ← conteúdo vazio.

TAMANHO ← conteúdo vazio.

PT EXAME ← conteúdo vazio.

Tabela de ligados: construção vazia.

2. Nível de FOLHA (#01)

PROCEDIMENTO:

- Verificar se o conteúdo do campo TIPO do topo da pilha de construções da ZI é uma das palavras reservadas: CV-COERÊNCIA, CV-ACESSO, CV-FONTE ou CV-USUÁRIO. Caso não o seja, enviar mensagem de erro ao usuário, armazenar código de erro no registrador STATUS.
- Caso o identificador fornecido seja nome de componente imediato da construção FOLHA, acessar DIRARQTAB com chave formatada com o conteúdo do campo TIPO do topo da pilha de construções da ZI. Armazenar o endereço da tupla acessada em DIRARQTAB no registrador ARQ, e o conteúdo do atributo DTIPO no registrador TIPO.

  Formatar e incluir uma ligação em TABAVL com as seguintes informações:

  Ligante:

  IND NÍVEL ← indicador de nível #05.

  NÍVEL ← conteúdo do registrador NV EMB.

  IND ARQ ← conteúdo do registrador ARQ.

  MARCA ← conteúdo vazio.

TIPO ← conteúdo do registrador TIPO.

DESLOCAMENTO ← conteúdo vazio.

TAMANHO ← conteúdo vazio.

TIPO C/E ← tipo escolhido (E).

PT EXAME ← conteúdo vazio.

Tabela de ligados: construção vazia.

3. Nível de FICHA DE ACTRAB ou de Arquivo (#02)

PROCEDIMENTO:

- Verificar se o conteúdo do campo TIPO do topo da pilha de construção da ZI é DATA DA CRIAÇÃO ou DATA DA ÚLTIMA ATUALIZ. Caso não o seja, enviar mensagem de erro ao usuário, armazenar código de erro no registrador STATUS.
- Se o identificador fornecido é nome de componente imediato da construção FICHA, acessar a definição da tupla que define DIRARQTAB na TABCOMPNOM, para obter o tamanho, deslocamento e o tipo do componente imediato desejado, e armazenar nos registradores TAMANHO, DESLOC e TIPO respectivamente.

  Formatar e incluir uma ligação em TABAVL com as seguintes informações:

  Ligante:

  IND NÍVEL ← indicador de nível #0C.

  NÍVEL ← conteúdo do registrador NV EMB.

  IND ARQ ← conteúdo do registrador ARQ.

  MARCA ← conteúdo vazio.

  TIPO ← conteúdo do registrador TIPO.

  DESLOCAMENTO ← conteúdo do registrador DESLOC.

TAMANHO ← conteúdo do registrador TAMANHO.

TIPO C/E ← tipo escolhido (E).

PT EXAME ← conteúdo vazio.

Tabela de ligados:

Formatar ligado com os conteúdos dos registradores ARQ e DESLOC (endereço do item).

4. Nível de Arquivo (#03)

PROCEDIMENTO:

- Verificar se o conteúdo do campo TIPO do topo da pilha de construção da ZI é FICHA, TL ou TR. Se não, armazenar código de erro no registrador STATUS, enviar mensagem de erro ao usuário.
- Caso o identificador fornecido seja nome de componente imediato da construção de tipo arquivo, executar uma das três possibilidades de tratamento descritas abaixo:
  1. Campo TIPO do topo da pilha de construções da ZI possui conteúdo a palavra reservada FICHA.
     - Armazenar o indicador de nível correspondente no registrador IND NV (#02).
     - Armazenar o nome do pretipo padrão FICHA no registrador TIPO.
     - Acessar a descrição da tupla que compõe DIRARQTAB na folha interna e calcular o tamanho e deslocamento do atributo FICHA, armazenando os resultados nos registradores TAMANHO e DESLOC respectivamente.

2. Campo TIPO do topo da pilha de construções da ZI possui como conteúdo uma das palavras reservadas TR ou TL.

- Armazenar o indicador de nível correspondente (TR → #04 ou TL → #05), no registrador IND NV.
- Acessar DIRARQTAB através do endereço contido no registrador ARQ.

  Verificar se o conteúdo do atributo TIPO da TABELA é compatível com o tipo de tabela fornecido (TR ou TC). Caso não o seja, armazenar o código de erro no registrador STATUS, enviar mensagem de erro ao usuário.
- Caso tipo de tabela compatível, armazenar o conteúdo do atributo DTIPO da tupla lida em DIRARQTAB no registrador TIPO. Formatar e incluir uma ligação em TABAVL, com as seguintes informações:

  Ligante:

  IND NÍVEL ← conteúdo do registrador IND NV.

  NÍVEL ← conteúdo do registrador NV EMB.

  IND ARQ ← conteúdo do registrador ARQ.

  MARCA ← conteúdo vazio.

  TIPO ← conteúdo do registrador TIPO.

  DESLOCAMENTO ← conteúdo do registrador DESLOC.

  TAMANHO ← conteúdo do registrador TAMANHO.

  TIPO C/E ← tipo escolhido (E).

  PT EXAME ← conteúdo vazio.

  Tabela de ligados:

  FICHA

  Formatar ligado com conteúdo dos registradores ARQ e DESLOC (endereço da FICHA).

TR ou TL

Construção vazia.

5. Nível de Tupla (#n7)

PROCEDIMENTO:

- Acessar TABCOMPNOM com chave formatada com conteúdo do registrador TIPO. Acessar a tabela de ligados da ligação acessada com chave formatada com conteúdo do campo TIPO do topo da pilha de construção da ZI. Caso não exista ligado correspondente, enviar mensagem de erro ao usuário armazenar código de erro no registrador STATUS.
- Caso exista componente imediato com o nome fornecido, armazenar os conteúdos dos seguintes atributos do ligado acessado em TABCOMPNOM nos seguintes registradores:

IND NV ← Depende do pretipo do tipo do ligado acessado ou seja, pretipo TUP => #nA ou pretipo ITEM => #nB.

TIPO ← conteúdo do atributo TERMTIPO do ligado acessado em TABCOMPNON.

DESLOC ← conteúdo do atributo DESLOC do ligado acessado em TABCOMPNOM.

TAMANHO ← conteúdo do atributo TAMANHO do ligado acessado em TABCOMPNOM.

Formatar e incluir uma ligação em TABAVL com as seguintes informações:

Ligante:

IND NÍVEL ← conteúdo do registrador IND NV.

NÍVEL ← conteúdo do registrador NV EMB.

IND ARQ ← conteúdo do registrador ARQ.

MARCA ← conteúdo vazio.

TIPO ← conteúdo do registrador TIPO.

DESLOCAMENTO ← conteúdo do registrador DESLOC.

TAMANHO ← conteúdo do registrador TAMANHO.

TIPO C/E ← tipo escolhido (E).

PT EXAME ← conteúdo vazio.

Tabela de ligados:

Copiar a tabela de ligados da ligação candidata acessada anteriormente para a ligação que está sendo inserida em TABAVL.

6. Nível de Ligação (#n8)

PROCEDIMENTO:

- Verificar se o conteúdo do campo TIPO do topo da pilha de construções da ZI é L ou T. Se não enviar mensagem de erro ao usuário, armazenar código de erro no registrador STATUS.
- Caso identificador válido, acessar TABCOMPNOM com chave formatada com o conteúdo do registrador TIPO. Acessar a tabela de ligados da ligação acessada, com chave formatada com o conteúdo do campo TIPO do topo da pilha de construções da ZI. Armazenar o conteúdo do atributo TERMTIPO do ligado acessado no registrador TIPO.

  Armazenar no registrador IND NV a indicação de nível da construção endereçada, ou seja: L => #n9 e T => #n4. Formatar e incluir uma ligação em TABAVL com as seguintes informações:

  Ligante:

  IND NÍVEL ← conteúdo do registrador IND NV.

NÍVEL ← conteúdo do registrador NV EMB.

MARCA ← conteúdo do registrador ARQ.

TIPO ← conteúdo do registrador TIPO.

DESLOCAMENTO ← conteúdo vazio.

TAMANHO ← conteúdo vazio.

TIPO C/E ← tipo escolhido (E).

PT EXAME ← conteúdo vazio.

Tabela de Ligados:

Copia a tabela de ligados da ligação candidato acessada anteriormente para a ligação em questão a ser inserida em TABAVL.

7. Nível de Componente de Coleção de Tuplas (#nC)

PROCEDIMENTO:

- Alterar o conteúdo do registrador IND NV para conter a indicação de nível de tupla (#n7).
- Executar mesmo procedimento de nível de tupla.

8. Nível de componente de Coleção de Ligação (#nD)

PROCEDIMENTO:

- Alterar o conteúdo do registrador IND NV para conter a indicação de nível de ligação (#n8).
- Executar mesmo procedimento de nível de ligação.

9. Nível de Construção Auxiliar (#10)

PROCEDIMENTO:

- Acessar Tabela de Construções do Canal Auxiliar (TAB-CONSAUX) com chave formatada com conteúdo do campo TIPO do tipo da pilha de construção da ZI. Caso não exista a chave correspondente, enviar mensagem de erro ao usuá-

rio, armazenar código de erro no registrador STATUS.

- Caso contrário, existe componente imediato com nome fornecido, armazenar o endereço da tupla acessada em TABCONSAUX no registrador PX. Verificar se o conteúdo do atributo DPRETIPO da tupla lida tem como conteúdo a construção vazia; caso afirmativo, armazenar as seguintes informações nos seguintes registradores:

IND NV ← indicador de nível #1F (construção sem tipo).

TIPO ← conteúdo vazio.

DESLOC ← conteúdo vazio.

TAMANHO ← conteúdo vazio.

Caso contrário, construção endereçada tenha tipo, armazenar as seguintes informações nos seguintes registradores:

IND NV ← dependendo do conteúdo do pretipo do tipo da construção acessada em TABCONSAUX, temos:

TAREL => #14

TALIG => #15

COLITENS => #16

TUP => #17

LIG => #18

DATA<br>HORA<br>INT<br>REAL<br>NUNCAR } => #1C

ARQ → dependendo do conteúdo do registrador IND NV, temos:

#1C => endereço da entrada em DIRARQTAB que descreve TABCONSAUX.

#14, #15, #16, #17, #18 => acessar DIRARQTAB com chave formatada com o conteúdo do campo TIPO do topo da pilha de construções da ZI. Armazenar neste registrador o endereço da descrição em DIRARQTAB da construção desejada.

TIPO ← dependendo do pretipo do tipo da construção acessada no canal auxiliar, temos:

TAREL, TALIG, COLITENS, ITEM => conteúdo do atributo DTIPO da tupla lida em TABCONSAUX

TUP OU LIG:

- Acessar TABCONSCOL com chave formatada com o conteúdo do atributo DTIPO da tupla lida em TABCONSAUX.
- Armazenar no referido registrador o conteúdo do atributo TERMTIPO da tupla acessada em TABCONSCOL.

DESLOC → conteúdo vazio.

TAMANHO → dependendo do conteúdo do registrador IND NV, temos:

#1C => tamanho do item acessado.

Qualquer outro valor => conteúdo vazio.

Formatar e incluir uma ligação em TABAVL com as seguintes informações:

IND NÍVEL ← conteúdo do registrador IND NV.

NÍVEL ← conteúdo do registrador NV EMB.

IND ARQ ← conteúdo do registrador ARQ.

MARCA ← conteúdo vazio.

TIPO ← conteúdo do registrador TIPO.

DESLOCAMENTO ← conteúdo vazio.

TIPO C/E ← tipo escolhido (E).

PT EXAME ← conteúdo do registrador PX.

Tabela de ligados:

Dependendo do pretipo da construção endereçada temos os seguintes casos a considerar:

1. ITEM => ligado formado pelo endereço contido no atributo PONTEIRO da tupla acessada em TABCONSAUX.
2. TUP ou LIG => ligado formado com o endereço da única tupla ou ligação existente na tabela criada para armazenar a mesma no canal auxiliar.
3. Coleção => construção vazia.

10. Nível de Tabela Ligacional (#n4), de Tabela Relacional (#n5), de Coleção de Itens (#n6) e de Componente de Coleção de Itens (#nE)

PROCEDIMENTO:

- Armazenar código de erro no registrador STATUS.
- Enviar mensagem de erro para o usuário

- Desempilhar conteúdos do topo da pilha de valbool e da pilha de Construções da ZI.
- Excluir ligação Candidato acessada anteriormente em TABAVL.
- Caso pretipo da Construção, cuja descrição se encontra no topo da pilha de construções de ZI, seja diferente do pretipo padrão NOME, faça:

Tratamento:

- Verificar o conteúdo do topo da pilha de valbool é VERDADEIRO. Caso afirmativo, faça:
  - Acessar a TABAVL com chave formatada com o conteúdo do registrador NV EMB e indicação de conjunto de pontos candidatos (TIPO C/E = C). Verificar se o conteúdo do atributo IND NÍVEL do ligante acessado é #nC, #nD ou #nE. Se diferente de um desses níveis, enviar mensagem de erro ao usuário, armazenar código de erro no registrador STATUS, e excluir a ligação candidato de TABAVL.
  - Caso contrário, nível de coleção, salvar o conteúdo do atributo PT EXAME do ligante candidado acessado no registrador PX. Acessar TABAVL com chave formatada com conteúdo do registrador NV EMB e indicação de ligação do tipo escolhido (TIPO C/E = E). Caso não exista ligação correspondente, formatar e incluir um ligante numa ligação do tipo

Escolhida, ou seja:

- Alterar o ligante candidato acessado anteriormente para modificar o conteúdo de alguns atributos:

  MARCA ← conteúdo vazio.

  TIPO C/E ← tipo escolhido (E).

  PT EXAME ← conteúdo vazio.

- Incluir ligante alterado em uma nova ligação em TABAVL. Formatar e incluir na Tabela de ligados da ligação do tipo escolhida em TABAVL, um ligado cujo conteúdo será o do registrador PX.

- Desempilhar conteúdos do topo da pilha de valbool e da pilha de construções da ZI.

VOL

RESUMO - Empilhar a palavra chave VOL.

PROCEDIMENTO:

- Armazenar a palavra chave VOL no topo da pilha de trabalho.

CARREGAR VOL CC

RESUMO - Carregar o volume, sobre o qual foi definido um conjunto Controlador de Campo, no registrador da máquina virtual VOLUME.

PROCEDIMENTO:

- Verificar se existe algum volume aberto para a execução MICROLOBAN. Caso não exista, armazenar código de erro no registrador STATUS e enviar mensagem de erro para o usuário.
- Caso contrário, existência de volume aberto, dependendo do conteúdo do topo da pilha de trabalho, temos:

VOL

Tratamento:

- Limpar registrador VOLUME.
- Varrer as entradas definidas na pilha de volumes de E/S, executando o procedimento descrito abaixo para cada volume:
  - Comparar o conteúdo do campo NV PROF da referida pilha com o conteúdo do topo da pilha de nível de profundidade.
  - Caso a igualdade ocorra, verificar se o conteúdo do campo CC desta entrada na pilha de volumes de E/S é igual a -1 (definição de conjunto controlador).
  - Se conjunto controlador definido, verificar se o conteúdo do registrador VOLUME é vazio. Caso afirmativo, armazenar o endereço da entrada descrita do volume na pilha no registrador VOLUME.
  - Se o registrador VOLUME já possuir conteúdo, pesquisar se um dos conjuntos controladores não possui marca. Se sim, armazenar o endereço da descrição do volume sobre o qual está definido o conjunto controlador sem marca no registrador VOLUME. Caso os dois conjuntos não tenham marcas ou tenham, enviar mensagem de erro ao usuário e armazenar código de erro no registrador STATUS (sai do laço).
- Verificar se conteúdo do registrador VOLUME possui conteúdo válido. Se não enviar mensagem de erro ao usuário e armazenar código de erro no registrador STATUS (Inexistência de conjunto controlador).

ID

Tratamento:

- Varrer as entradas da pilha de volume de E/S, verificando se o conteúdo do campo MARCA é igual ao do topo da pilha de trabalho. Caso não seja encontrado o nome da marca fornecida, enviar mensagem de erro ao usuário e armazenar código de erro no registrador STATUS.
- Caso contrário armazenar o endereço da entrada descritiva do volume sobre o qual está definido o conjunto controlador "marcado", no registrador VOLUME.

- Desempilhar conteúdo do topo da pilha de trabalho.

CARREGAR VOL

RESUMO - Transformar o volume fornecido em volume corrente.

PROCEDIMENTO:

- Verificar se existe volume aberto; se não enviar mensagem de erro para o usuário e armazenar código de erro no registrador STATUS.
- Se existe volume aberto, temos dois possíveis procedimentos a executar dependendo do conteúdo do topo da pilha de trabalho:

  VOL

  Tratamento:

  - Verificar a existência de apenas um volume aberto. Caso contrário, enviar mensagem de erro ao usuário e armazenar código de erro no registrador.
  - Caso contrário, armazenar o endereço da descrição do volume aberto de E/S, no registrador VOLUME.

ID

Tratamento:

- Acessar a pilha de volumes de E/S através do nome do volume que se encontra no topo da pilha de trabalho. Caso o volume não seja encontrado, armazenar código de erro no registrador STATUS e enviar mensagem de erro ao usuário.
- Caso contrário, volume acessado, armazenar o endereço de sua descrição na pilha de volumes de E/S, no registrador VOLUME.

- Desempilhar conteúdo do topo da pilha de trabalho.

NÃO

RESUMO - Executar o operador booleano unário de negação NÃO.

PROCEDIMENTO:

- Acessar o conteúdo do topo da pilha de valbool.
- Dependendo do valor booleano acessado temos:
  . Conteúdo VERDADEIRO => armazenar FALSO no topo da pilha de valbool.
  . Conteúdo FALSO => armazenar VERDADEIRO no topo da pilha de valbool.

E

RESUMO - Executar o operador booleano de disjunção E.

PROCEDIMENTO:

- Acessar os conteúdos das posições topo e (topo-1) na pilha de valbool.
- Dependendo dos valores booleanos acessados, temos:

| Conteúdo topo | Conteúdo (topo-1) | Resultado |
|---|---|---|
| VERDADEIRO | VERDADEIRO | VERDADEIRO |
| VERDADEIRO | FALSO | FALSO |
| FALSO | VERDADEIRO | FALSO |
| FALSO | FALSO | FALSO |

  O resultado é armazenado na posição (topo-1) da pilha de valbool.
- Desempilhar o conteúdo do topo da pilha de valbool.

OU

RESUMO - Executar o operador booleano de exclusão OU.

PROCEDIMENTO:

- Acessar os conteúdos das posições topo e (topo-1) da pilha de valbool.
- Dependendo dos valores booleanos acessados, temos:

| Conteúdo topo | Conteúdo (topo-1) | Resultado |
|---|---|---|
| VERDADEIRO | VERDADEIRO | VERDADEIRO |
| VERDADEIRO | FALSO | VERDADEIRO |
| FALSO | VERDADEIRO | VERDADEIRO |
| FALSO | FALSO | FALSO |

O resultado é armazenado na posição (topo-1) da pilha de valbool.

- Desempilhar o conteúdo do topo da pilha de valbool.

OUEX

RESUMO - Executar o operador booleano.

PROCEDIMENTO:

- Acessar os conteúdos das posições topo e (topo-1) da pilha de valbool.
- Dependendo dos valores booleanos acessados, temos:

| Conteúdo topo | Conteúdo (topo-1) | Resultado |
|---|---|---|
| VERDADEIRO | VERDADEIRO | VERDADEIRO |
| VERDADEIRO | FALSO | FALSO |
| FALSO | VERDADEIRO | FALSO |
| FALSO | FALSO | VERDADEIRO |

O resultado é armazenado na posição (topo-1) da pilha de valbool.

- Desempilhar o conteúdo do topo da pilha de valbool.

IMPL

RESUMO - Executar o operador booleano de implicação IMPL.

PROCEDIMENTO:

- Acessar os conteúdos das posições topo e (topo-1) da pilha de valbool.
- Dependendo dos conteúdos dos valores booleanos acessados, temos:

| Conteúdo topo | Conteúdo (topo-1) | Resultado |
|---|---|---|
| VERDADEIRO | VERDADEIRO | FALSO |
| VERDADEIRO | FALSO | FALSO |
| FALSO | VERDADEIRO | FALSO |
| FALSO | FALSO | VERDADEIRO |

O resultado é armazenado na posição (topo-1) da pilha de valbool.

- Desempilharo conteúdo do topo da pilha de valbool.

CRIAR CTR

RESUMO - Criar conjunto controlador temporário na tabela de marcas (TABMARCA).

PROCEDIMENTO:

- Acessar TABAVL com chave formatada com o conteúdo do registrador NV EMB somado de uma unidade e indicação de conjunto de pontos escolhidos (TIPO C/E = E). Caso não existam pontos escolhidos, enviar mensagem de erro ao usuário, armazenar o valor booleano FALSO no topo da pilha de valbool e o código de erro no registrador STATUS.

- Caso contrário, existem pontos endereçados, acessar o atributo TIPO do referido ligante para verificar se o seu pretipo é válido para tornar o conjunto de pontos endereçado em conjunto controlador, ou seja:

  Se o conteúdo do atributo TIPO for DATA, HORA, INT, REAL => pretipo válido.

  Se o conteúdo do atributo TIPO for ID => acessar a tabela que descreve a coerência (CV-COERÊNCIA) na folha do acervo, e verificar se o pretipo desse tipo é TUP, LIG ou NUMCAR. Se sim => pretipo válido; caso contrário, enviar mensagem de erro ao usuário, armazenar o valor booleano FALSO no topo da pilha de valbool e o código de erro no registrador STATUS.

Se pretipo válido, incrementar de uma unidade o conteúdo do topo da pilha de nível de profundidade.

Formatar e incluir uma ligação em TABMARCA, com as seguintes informações:

Ligante:

NOME DA MARCA ← conteúdo vazio.

TIPO DA MARCA ← conteúdo do topo da pilha de nível de profundidade.

IDENT. DO ARQ MARCADO ← conteúdo do atributo IND ARQ de TABAVL.

TIPO DO PONTO ← conteúdo do atributo TIPO de TABAVL.

DESLOC DO ITEM ← conteúdo do atributo DESLOCAMENTO DE TABAVL.

TAMANHO ← conteúdo do atributo TAMANHO DE TABAVL.

PC ← conteúdo do primeiro ligado da tabela de ligados de TABAVL.

Tabela de ligados:

Tabela de ligados da ligação acessada em TABAVL.

- Excluir ligação do tipo escolhida acessada em TABAVL.

LAÇO PARA TODO

RESUMO - Executar laço sobre os pontos pertencentes ao conjunto controlador criado para a instrução PARA TODO.

PROCEDIMENTO:

- Acessar conteúdo do topo da pilha de valbool, verificar se o mesmo é igual ao valor booleano VERDADEIRO. Se não, excluir ligação de TABMARCA e decrementa de uma unidade o conteúdo do topo da pilha de nível de profundidade.
- Caso o seja, acessar TABMARCA com chave formatada com conteúdo do nome de marca vazia e do tipo da marca igual ao conteúdo do topo da pilha de nível de profundidade.

  Acessar a tabela de ligados comparando o conteúdo do atributo PC do ligance com o conteúdo de cada ligado. Caso seja encontrado o ligado correspondente, verificar se ele é o último ponto do conjunto controlador. Se não armazenar o conteúdo do "próximo" ligado no atributo PC do referido ligante, e continua laço.

  Caso seja detectado fim de conjunto controlador, exclui ligação de TABMARCA.

VERDADEIRO

RESUMO - Executar o operador booleano VERDADEIRO.

PROCEDIMENTO:

- Acessar TABAVL com chave formatada com o conteúdo do registrador NV EMB e indicação de conjunto de pontos candidatos (TIPO C/E = C). Caso não exista ligação correspondente em TABAVL, armazenar o valor booleano VERDADEIRO na pilha de valbool.
- Caso contrário, verificar se o conteúdo do atributo IND NÍVEL do ligante acessado em TABAVL, é igual a #nD (componente de coleção de tuplas) #nE (componente de coleção de ligações) ou #nF (componente de coleção de itens). Se não, armazenar código de erro no registrador STATUS e enviar mensagem de erro ao usuário.
- Caso seja componente de coleção, alterar o conteúdo do atributo IND NÍVEL do ligante de TABAVL de acordo com o tipo de elementos que compõem a coleção, ou seja:

  IND NÍVEL = #nD = IND NÍVEL ← #n7

  IND NÍVEL = #nE IND NÍVEL ← #n8

  IND NÍVEL = #nF IND NÍVEL ← #nC

  Armazenar VERDADEIRO na pilha de valbool.

NUM

RESUMO - Emplilhar palavra chave NUM na pilha de trabalho.

- Armazenar a palavra chave NUM na pilha de trabalho.

ALFA

RESUMO - Empilhar palavra chave ALFA na pilha de trabalho.

- Armazenar a palavra chave ALFA na pilha de trabalho.

DATA

RESUMO. - Empilhar palavra chave DATA na pilha de trabalho.

- Armazenar a palavra chave DATA na pilha de trabalho.

HORA

RESUMO - Empilhar palavra chave HORA na pilha de trabalho.

- Armazenar a palavra chave HORA na pilha de trabalho.

PADRÃO

RESUMO - Empilhar palavra chave PADRÃO na pilha de trabalho.

- Armazenarpalavra chave PADRÃO na pilha de trabalho.

As instruções restantes LIBAN pertencentes a tradução de uma expressão booleana, não serão descritas detalhadamente devido ao fato da idéia básica do funcionamento das mesmas já ter sido bastante enfatizado, e também por seus procedimentos serem razoavelmente longos. Desta forma só será fornecido um resumo da finalidade das instruções restantes. Qualquer dúvida quanto a sua execução, procurar a autora do trabalho.

C

RESUMO - Obtém uma cópia da construção endereçada. Sua semântica depende do contexto de aplicação.

N

RESUMO - Obtém uma cópia do nome da construção endereçada.

ELEM

RESUMO - Executa a operação de pertinência de elementos a conjunto de elementos. Vale ressaltar que os tipos dos operandos têm que ser compatíveis.

VALIDAR ORDEM

RESUMO - Valida a ordem fornecida pelo usuário com o pretipo dos operandos das operações de comparação.

TESTAR TIPOS

RESUMO - Valida os tipos dos operandos para a aplicação dos operadores de comparação.

IGUAL, MAIOR, MENOR, MAIOR IGUAL, MENOR IGUAL, DIFERENTE

RESUMO - Executa operações de comparação

INSTRUÇÕES DE CONTROLE DE EXECUÇÃO

INIC LAÇO

RESUMO - Determina o início de laço interno para o Interpretador MICROLOBAN.

FIM LAÇO

RESUMO - Determina o fim de laço interno para o Interpretador MICROLOBAN.

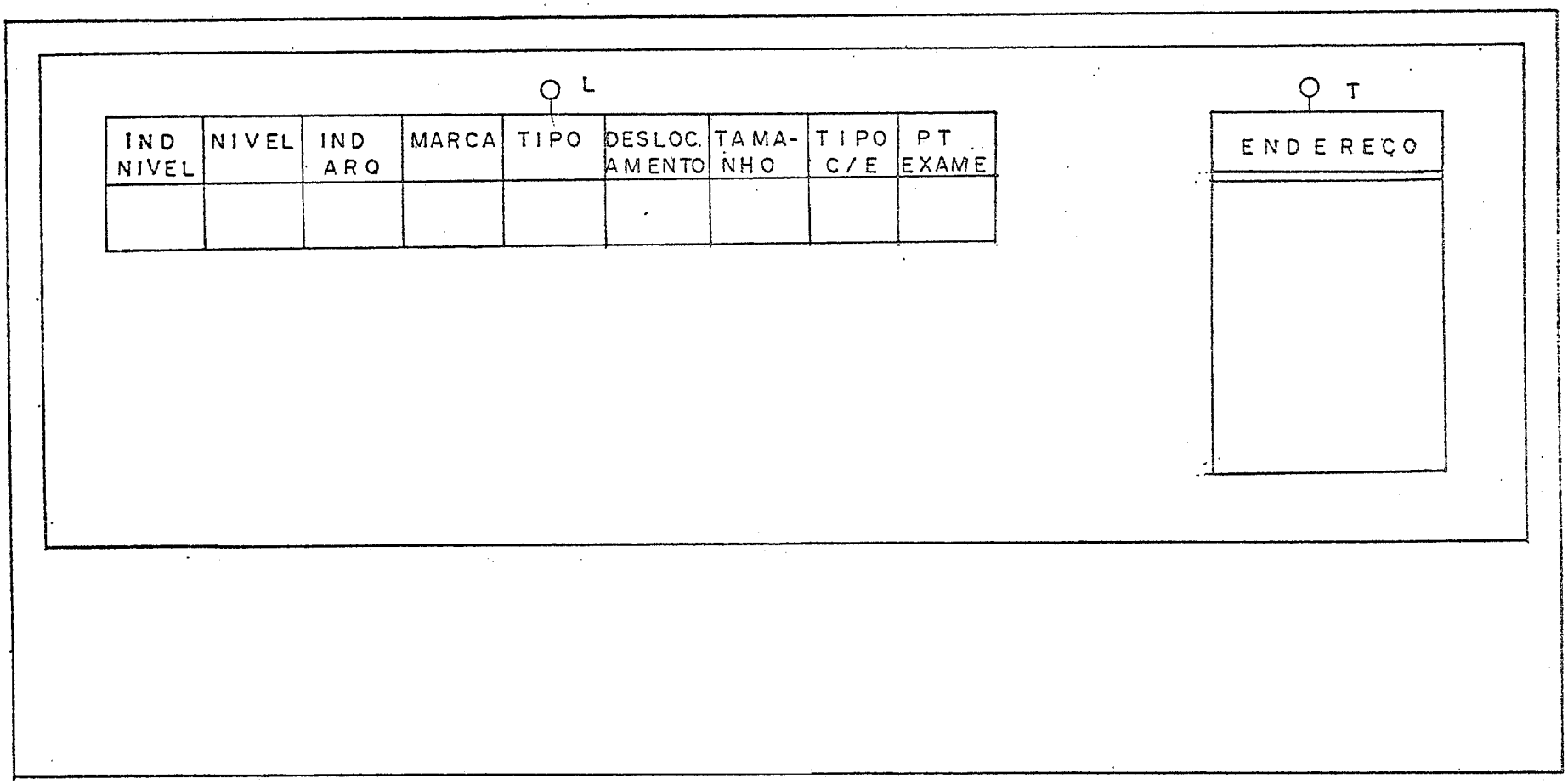



TABELA DE AVALIAÇÃO DE ENDEREÇO DE PONTO - TABAVL

| NOME DO ARQUIVO/ TABELA | TIPO DA TABELA | DTIPO | CARDINAL CORRENTE | CARDINAL MAXIMA | TAMANHO DA TUPLA | I-ENTR. NO IND PAG | I-ENTR. NO INDESP | DESLOC. DA CHAVE | TAMANHO DA CHAVE | LIMITE ATUAL | INDICAÇÃO DE ACESSO | DATA DA CRIAÇÃO | DATA DA ULTIMA ATUALIZ. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| <nome> | <tipo> | <dtipo> | <cardcor> | <cardmax> | <tamanho> | <primentip> | <primentie> | <desloc> | <tamchav> | <limite> | <indac> | <datcri> | <datat> |
| | | | | | | | | | | | | | |

DIRETORIO DE ARQUIVOS E TABELAS - DIRARQTAB

| NOME | DPRETIPO | DTIPO | PONTEIRO |
|---|---|---|---|
| ⟨nome⟩ | ⟨dpretipo⟩ | ⟨dtipo⟩ | ⟨ponteiro⟩ |
| | | | |

TABELA DE CONSTRUÇÕES DO CANAL AUXILIAR (TABCONSAUX)

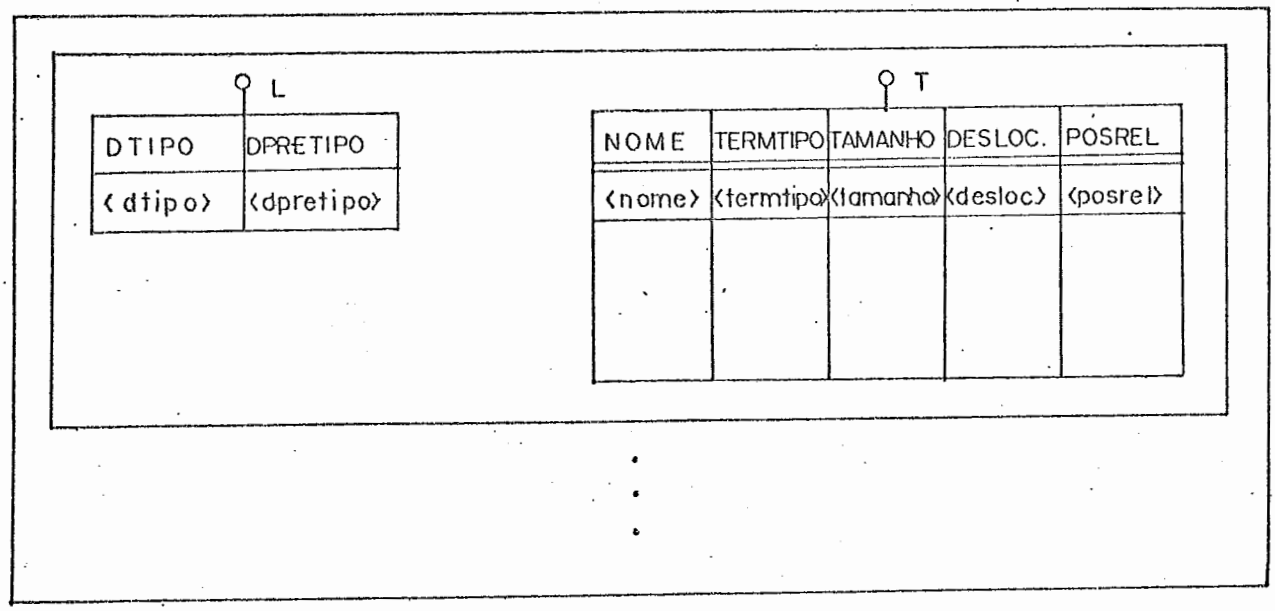



TABELA DE COMPOSIÇÃO DE NOMINAÇÕES (TABCOMPNOM)

| DTIPO | DPRETIPO | TERMTIPO | DESLCHAV | TAMCHAV | CARDINAL |
|---|---|---|---|---|---|
| ⟨dtipo⟩ | ⟨dpretipo⟩ | ⟨termtipo⟩ | ⟨deslchav⟩ | ⟨tamchav⟩ | ⟨cardinal⟩ |
| | | | | | |

TABELA DE CONSISTÊNCIA DE COLEÇÕES (TABCONSCOL)

| DTIPO | DPRETIPO | VALOR-1 | OPERADOR-1 | VALOR-2 | OPERADOR-2 |
|---|---|---|---|---|---|
| ⟨dtipo⟩ | ⟨dpretipo⟩ | ⟨valor 1⟩ | ⟨operador 1⟩ | ⟨valor 2⟩ | ⟨operador 2⟩ |
| | | | | | |

TABELA DE DEFINIÇÃO POR INTERVALO (TABDEFINT)

| IDENT. DO ARQ 1 | ATRIBUTO 1 | IDENT. DO ARQ 2 | ATRIBUTO 2 |
|---|---|---|---|
| ⟨arq 1⟩ | ⟨atributo 1⟩ | ⟨arq 2⟩ | ⟨atributo 2⟩ |
| | | | |

TABELA DE CONEXÃO DO ACTRAB (TABCONACT)

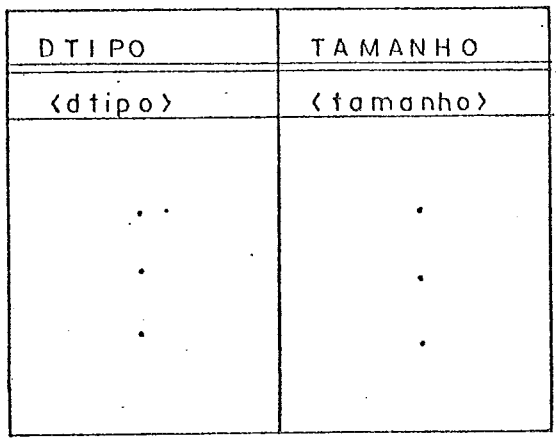



TABELA DE DEFINIÇÃO DE SIGLAS (TABDEFSIG)

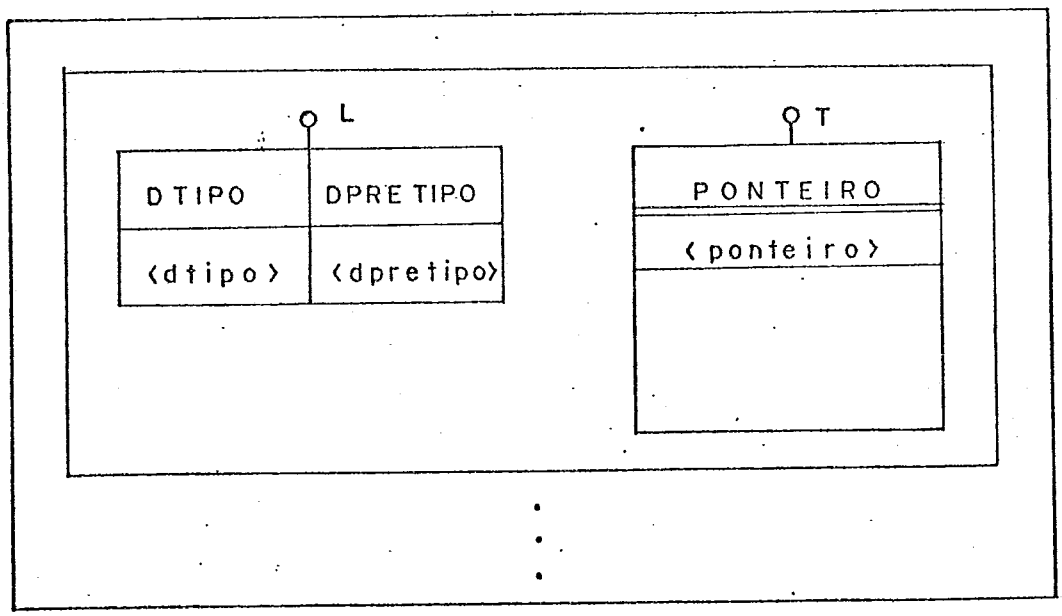



TABELA DE DEFINIÇÃO POR EXTENSÃO (TABDEFEXT)

BIBLIOGRAFIA

(01) ASTRAHAN, M.M. et alli, "A History and Evaluation of System R", Research Report RJ2843 , June (1980).

(02) ASTRAHAN, M.M. et alli, "Sistema R : A Relational Approach to Database Management", ACM-TODS, Vol.1, Nº 2, (1967).

(03) BRODIE, M.L., "ZETA : A Prototype Relational Data Base Management System", Technical Report CSRG-51. February (1975), Department of Computer Science of the University of Toronto.

(04) BRODIE, M.L. and SCHMIDT, J.W., "Final Report of the ANSI/X3/SPARC DBS-SG, Relational Database Task Group", September, (1981).

(05) BRODIE, M.L. and SCHMIDT, J.W. (Eds.), "Relational Database Systems: Analysis and Comparison, Springer-Verlag, Berlin Heidelberg New York, (1983).

(06) BRODIE, M.L. and SCHMIDT, J.W. (Eds.), "Report of the ANSI Relational Task Group", ACM SIGMOD, July, (1982).

(07) CASTILHO, J.M.V., CUNHA PEREIRA Fº, J. e RICHTER, G., "Projeto MINIBAN : Relatório Final da Segunda Etapa (Partes A e B, 420 págs.)", Rio de Janeiro, CNPq/DIGIBRÁS/GMD/UFRGS, Abril (1978).

(08) CHAMBERLIN, D.D. et alli., "SEQUEL 2: A Unified Approach to Data Definition, Manipulation and Control", IBM Journal of Research and Development, November, (1976).

(09) COBRA, "Manual de Referência do Sistema Operacional Monoprogramável - SOM COBRA/TD", T3V0100002-0, (1980).

(10) CODD, E.F., "A Database sublanguage founded on the Relational Calculus", Proc.ACM SIGFIDET Workshop, San Diego, Calif., November, (1971).

(11) CODD, E.F., "A Relational Model of Data for Large Shared Data Banks", CACM, vol.13, nº 6, June, (1970).

([12]) CRIVOROT, S.H., "Tradutor MICROLOBAN", Tese de M.Sc., COPPE/UFRJ, (1983).

([13]) CUNHA, P.Fº e RICHTER, G., "Projeto MINIBAN : Relatório Final da Primeira Etapa" (170 págs.), Rio de Janeiro, CNPq/DIGIBRÁS/GMD, Julho, (1977).

([14]) DANTAS, J.S., "Uma Proposta de Ambiente Interno para Implementação de SGBD em Sistemas de Pequeno Porte e sua Utilização na Implementação MICROLOBAN", Tese de M.Sc., COPPE/UFRJ (1982).

([15]) DATA, C.J., "An Introduction to Database Systems", 3a. edição, Addison-Wesley Publishing Company, (1981).

([16]) DURCHHOLZ, R. e RICHTER, G., "Concepts for Data Base Management Systems", in Data Base Management (Proc. IFIP Working Conf.), J.W.Klimbie and K.L.Hoffeman (Eds.), Amsterdam, North Holand Pub., (1974).

([17]) DURCHHOLZ, R. e RICHTER, G., "Information Management Concepts (IMC) for use with DBMS Interfaces", in Modelling in Data Base Management Systems (Proc. IFIP Working Conf.), G.M.Nijssen (Ed.), Amsterdam, North Holland Pub., (1976).

([18]) HEUSER, A.C. et alli, "Sistema L : Uma Implementação da Linguagem LOBAN", Anais do VIII SEMISH, Florianópolis, pp.169-186, Julho, (1981).

([19]) HUTT, A.T.F., "Relational Data Base Management System", Wiley-Interscience Publication, (1979).

([20]) LIMA, J.V. e HEUSER, C.A., "Um Analisador Semântico para a Linguagem LOBAN", Anais IX SEMISH, Ouro Preto, pp.15-27, Julho, (1982).

([21]) LIMA, V.L.S., "Um Estudo sobre Resolução de Operações de Consulta a Banco de Dados", Tese de M.Sc., UFRGS, (1982).

([22]) MYLOPOULOS, J. et alli, "A Multi-level Relational System". Proc. National Computer Conference, AFIPS Press, (1975).

([23]) PINTO, P.R.B., "Aspectos Conceituais sobre Concorrência em Banco de Dados", Tese de M.Sc., COPPE/UFRJ, (1979).

([24]) RICHTER, G., "On the Relationship Between Information and Data", Lectures Notes in Computer Science, vol.39, Springer-Verlag, Berlim, (1976).

([25]) RICHTER, G., "O Sentido e o Valor do Banco de Dados", Rio de Janeiro, Dados e Idéias, vol.2, nº 6, pp.2-14, Jun/Jul., (1977).

([26]) RICHTER, G. e CASTILHO, J.M.V., "Uma Interface para Sistemas de Informação : LOBAN - Linguagem de Operação de Banco de Dados", Anais do 11º CNPD-SUCESU, pp.349-354, (1978).

([27]) SANTOS, A.C. et alli, "Especificação da Linguagem LOBAN, 1980", Versão 2, Relatório Técnico , COPPE/UFRJ, (1981).

([28]) SIMONE, E.G., TELES, A.A., "Gerador de Analisadores Sintáticos RRP LL(1)", Anais VIII SEMISH, Florianópolis, pp.387-398, Julho (1981).

([29]) SOUZA, A.C.G., "IUC - Interface para Usuário Casual", Tese de M.Sc., COPPE/UFRJ, (1983).

([30]) SOUZA, A.C.G. et alli, "O Banco de Dados MICROLOBAN", Anais do IX SEMISH, Ouro Preto, pp.55-74, Julho, (1982).