UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM ESTUDO SOBRE RESOLUÇÃO DE OPERAÇÕES DE CONSULTA A BANCO DE DADOS

por

VERA LÚCIA STRUBE DE LIMA

Dissertação submetida como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Profª Lia Goldstein Golendziner
Orientadora

Prof. Carlos Alberto Heuser
Co-orientador

Porto Alegre, Janeiro de 1982.

AGRADECIMENTOS

À profª Lia Goldstein Golendziner, pelo estímulo e dedicação constantes, na orientação de todas as etapas deste trabalho.

Ao prof. Carlos Alberto Heuser, pelos conhecimentos introdutórios em banco de dados e co-orientação.

Ao Instituto de Informática da Pontifícia Universidade Católica do Rio Grande do Sul, na pessoa de sua diretora, profª Maria Lúcia Blanck Lisbôa, pelo apoio e incentivo durante todo o curso de Pós-Graduação.

Aos profs. Manoel Luiz Leão e José Palazzo Moreira de Oliveira, pelo interesse e consideração.

Ao Curso de Pós-Graduação em Ciência da Computação, na pessoa de seu coordenador, prof. Clesio Saraiva dos Santos, à IBM do Brasil Ltda. e à CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pelo apoio financeiro.

A Maria Helena Amaral Cardozo, pela datilografia, e a Cláudia Nerung Mattos, pelos desenhos.

A todos aqueles que, de alguma forma, vieram a contribuir para a realização deste trabalho.

Ao Valdeni
e aos meus pais.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ALGORITMOS

RESUMO

Este trabalho apresenta o estudo feito para resolução de operações de consulta que exigem o acesso e recuperação de dados, em um sistema de gerência de banco de dados. Os principais objetivos levados em conta são: a utilização de algoritmos que tirem proveito da situação específica em que se encontra cada operação, tal como a existência de ordenação entre os registros de um arquivo ou a disponibilidade de estruturas de acesso auxiliares, e a diminuição do número de recursos utilizados, incluindo tempo de execução e espaço de armazenamento.

O estudo foi feito para a linguagem LOBAN - Linguagem de Operação de Banco de Dados, em implementação através do Sistema L, na Universidade Federal do Rio Grande do Sul. A linguagem LOBAN é de alto nível, incluindo operações com funções equivalentes às da álgebra relacional.

ABSTRACT

This work presents a study for the resolution of retrieval operations on a data base management system. The main objectives considered are: the use of algorithms that take profit of the specific situation where each operation is found, such as the existence of a sort order among the records of a file or the availability of auxiliary access - paths, and the decrease of the number of resources used, involving execution time and storage space.

This study was done on LOBAN language - Linguagem de Operação de Banco de Dados (Data Base Operational Language), which is being implemented by the System L at Federal University of Rio Grande do Sul. LOBAN is a high-level language, including operations with equivalent functions to those of relational algebra.

1. INTRODUÇÃO

1.1 Motivação

As linguagens de operação de banco de dados com características da abordagem relacional contam com operadores capazes de manipular agregados maiores (por exemplo, tabelas inteiras), ao contrário das linguagens convencionais, capazes de manipular apenas registros de um arquivo. Na recuperação de dados, em linguagens desse tipo, é indispensável que sejam tomados certos cuidados, visando diminuir a quantidade de recursos utilizados, em termos de tempo de execução e espaço de armazenamento.

Na tentativa de melhorar a eficiência das instruções fornecidas pelo usuário, dois níveis de otimização podem ser aplicados /VER 76/: a otimização algébrica, que modifica a seqüência das operações fornecidas transformando-a em outra seqüência equivalente (operando em alto nível, sem necessidade de conhecimento da representação dos dados), e a otimização não-algébrica, que se preocupa com a utilização eficiente dos recursos disponíveis, a nível de implementação.

Este trabalho tem por objetivo estudar a geração de código para operações que manipulam agregados maiores como operandos, levando em conta que a otimização algébrica /GOL 80a, GOL 80b/ já foi realizada, e incluindo características da otimização não-algébrica. Entre as características incluídas, estão:

- o aproveitamento do contexto apresentado na seqüência de operações fornecidas pelo usuário, que envolve a existência de ordenação entre os registros de um arquivo e disponibilidade de estruturas de acesso auxiliares, como arquivos de inversões ou arquivos de associações, buscando reduzir o número de operações de entrada e saí-

da realizadas;

- a coordenação das ordens de classificação dos resultados das operações, com o objetivo de diminuir o número de arquivos intermediários gerados e também o número de operações de entrada e saída realizadas.

O estudo é feito sobre a linguagem LOBAN - Linguagem de Operação de Banco de Dados - descrita em /RIC 78, SAN 80/, conforme sua implementação através do Sistema L /HEU 81/, em andamento na Universidade Federal do Rio Grande do Sul, no minicomputador LABO 8034.

1.2 Organização da dissertação

A dissertação apresentada é composta de 8 capítulos. O capítulo 1 visa introduzir o leitor aos objetivos do estudo realizado, incluindo também a organização da dissertação.

O capítulo 2 descreve o sistema de banco de dados sobre o qual é feito o estudo, apresentando as características gerais da linguagem LOBAN e a descrição dos operadores estudados.

O capítulo 3 descreve a implementação da Interface LOBAN através do Sistema L, mostrando a arquitetura global utilizada e o modelo de dados disponível na base de dados interna. São aí apresentados o código de entrada e o código de saída utilizados no processo de geração de código para execução.

O capítulo 4 apresenta as comparações gráficas e as conclusões obtidas no estudo da recuperação de registros a partir de listas de identificadores de registros classificadas.

O capítulo 5 apresenta as características levadas em conta na elaboração dos algoritmos para resolução das várias operações estudadas, e a descrição detalhada de cada um desses algoritmos.

O capítulo 6 mostra, na forma de tabelas, a adequação de cada um dos algoritmos apresentados a situações específicas que podem ocorrer em um certo ponto da seqüência de operações a ser resolvida.

No capítulo 7 é apresentado o processo de seleção do algoritmo mais interessante a cada operação, com determinação das opções de ordem de classificação que o mesmo pode oferecer e, posteriormente, escolha da opção de ordem mais interessante à operação que irá utilizar como entrada este resultado, conforme idéias introduzidas em /SMI 75/.

Considerando que, no capítulo 5, são apresentados resumos para todos os algoritmos desenvolvidos, na leitura da dissertação pode ser omitida a descrição em detalhe desses algoritmos, sem prejuízo na compreensão global do assunto.

O capítulo 8, conclusão deste trabalho, inclui um retrospecto do estudo realizado e sugestões para continuá-lo.

2. A INTERFACE LOBAN

2.1 Introdução

A linguagem LOBAN - Linguagem de Operação de Banco de Dados - foi definida durante a execução da primeira etapa do projeto MINIBAN, e sua descrição completa pode ser encontrada junto à bibliografia /RIC 78, CAS 79, SAN 80, HEU 81, GOL 81a/.

LOBAN é a interface de comunicação entre o usuário e o Sistema de Gerência da Base de Dados, permitindo a manipulação das informações contidas na base de dados e no canal auxiliar (o qual armazena construções temporárias, existentes somente durante o tempo de execução do programa que as criou), e a execução de operações sobre os canais de entrada e saída.

A figura 2.1 mostra a interface LOBAN, utilizando uma estrutura de canais e estações descrita em /RIC 77/.

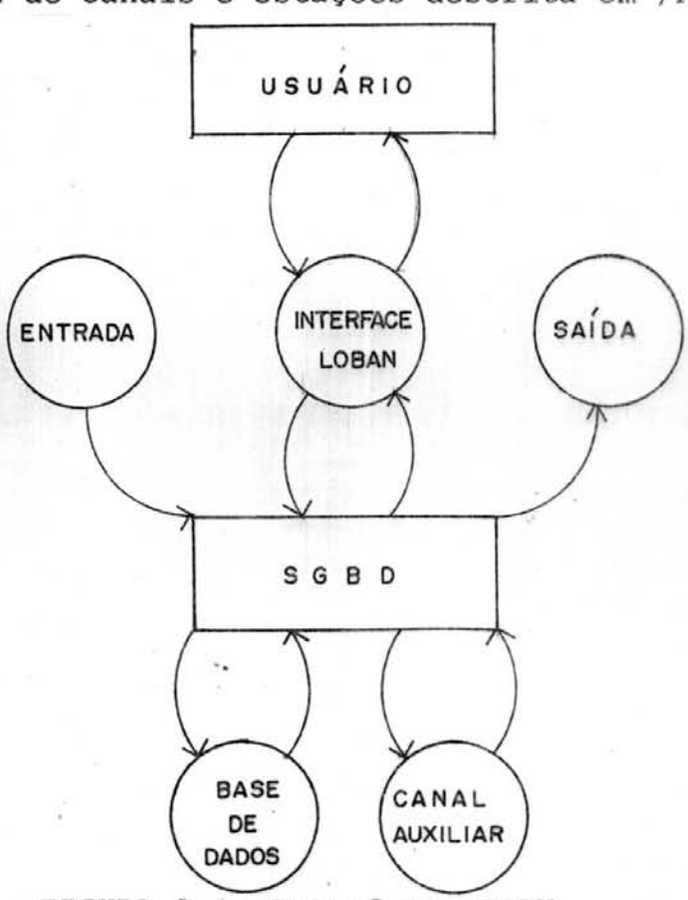


FIGURA 2.1: Interface LOBAN

2.2 O modelo de dados

É apresentada a seguir uma descrição, a nível abstrato, da organização das informações da base de dados. Para tanto é utilizado o conjunto de conceitos denominado IMC - "Information Management Concepts", desenvolvido para representar estruturas de informação /DUR 76/.

O conteúdo total da base de dados é denominado acervo total, e é composto por acervos setoriais, identificáveis por seus nomes. Na prática, um acervo setorial constitui o ambiente de trabalho de uma aplicação, não havendo relacionamento entre acervos setoriais /GOL 81a/.

Os acervos setoriais contêm, entre outras construções, os arquivos, também identificáveis por seus nomes, e que podem ser de dois tipos: relacional ou ligacional.

Um arquivo relacional (figura 2.2 e 2.3) é sempre composto por uma tabela relacional (sob nome TR), podendo incluir, opcionalmente, uma tupla sob nome FICHA, e uma relação de ordem, sob nome escolhido pelo usuário. A tabela relacional corresponde a um arquivo, nos sistemas convencionais, e é composta por tuplas, correspondentes aos registros desse arquivo.

A relação de ordem, quando especificada, permite que o usuário recupere as tuplas da tabela relacional em uma determinada ordem.

Um arquivo ligacional (figura 2.4) é composto de uma tabela ligacional, cujos componentes são denominados ligações; opcionalmente pode incluir, também, uma relação de ordem e uma tupla sob nome FICHA. Cada ligação equivale à ocorrência de um "set" da abordagem CODASYL/DBTG, e representa a associação entre uma tupla, denominada ligante (sob nome L), e uma tabela relacional, denominada tabela de ligados (sob nome T). A relação de ordem permite que as ligações sejam recuperadas em uma determinada ordem.

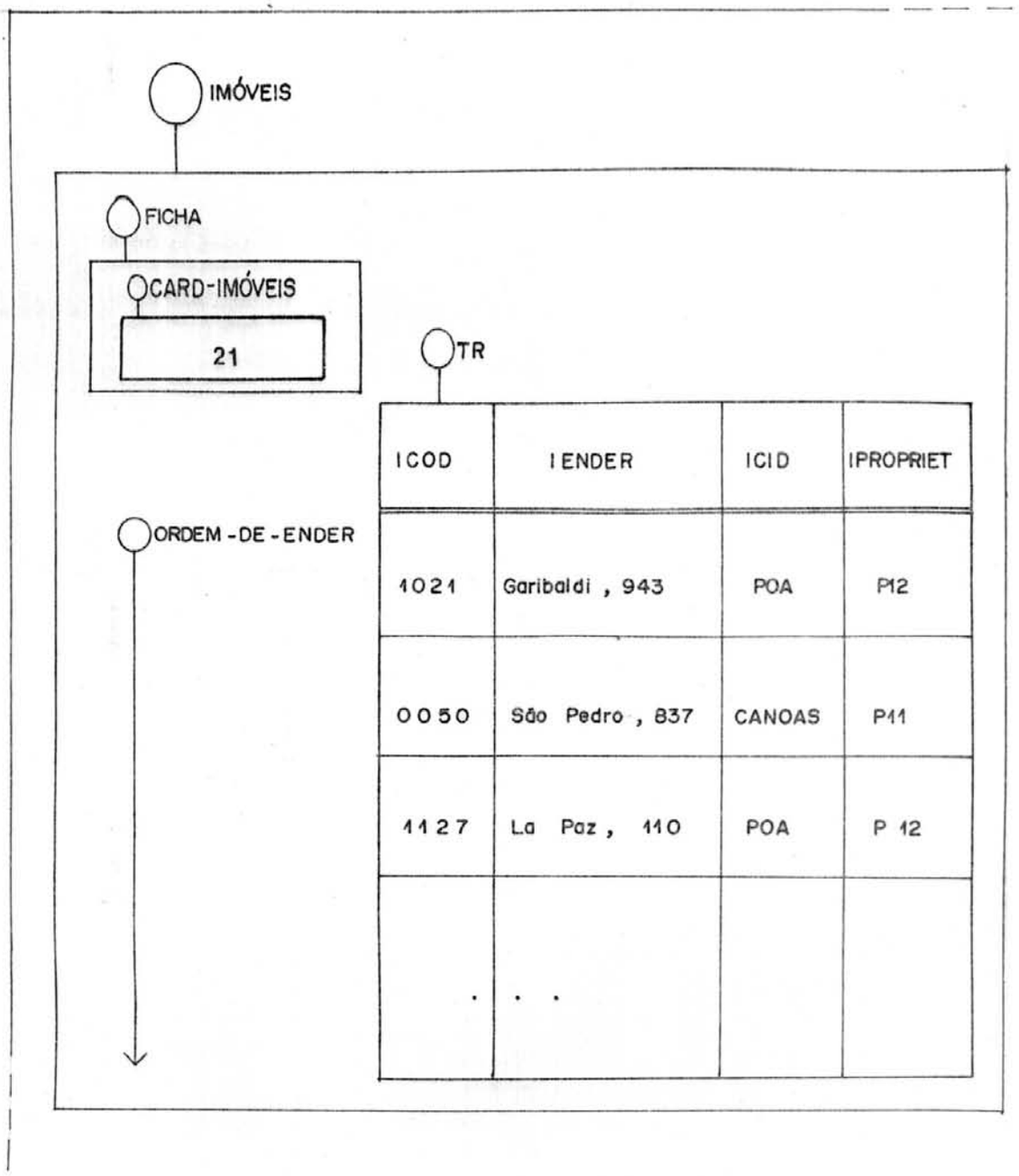


FIGURA 2.2: Arquivo relacional IMOVEIS

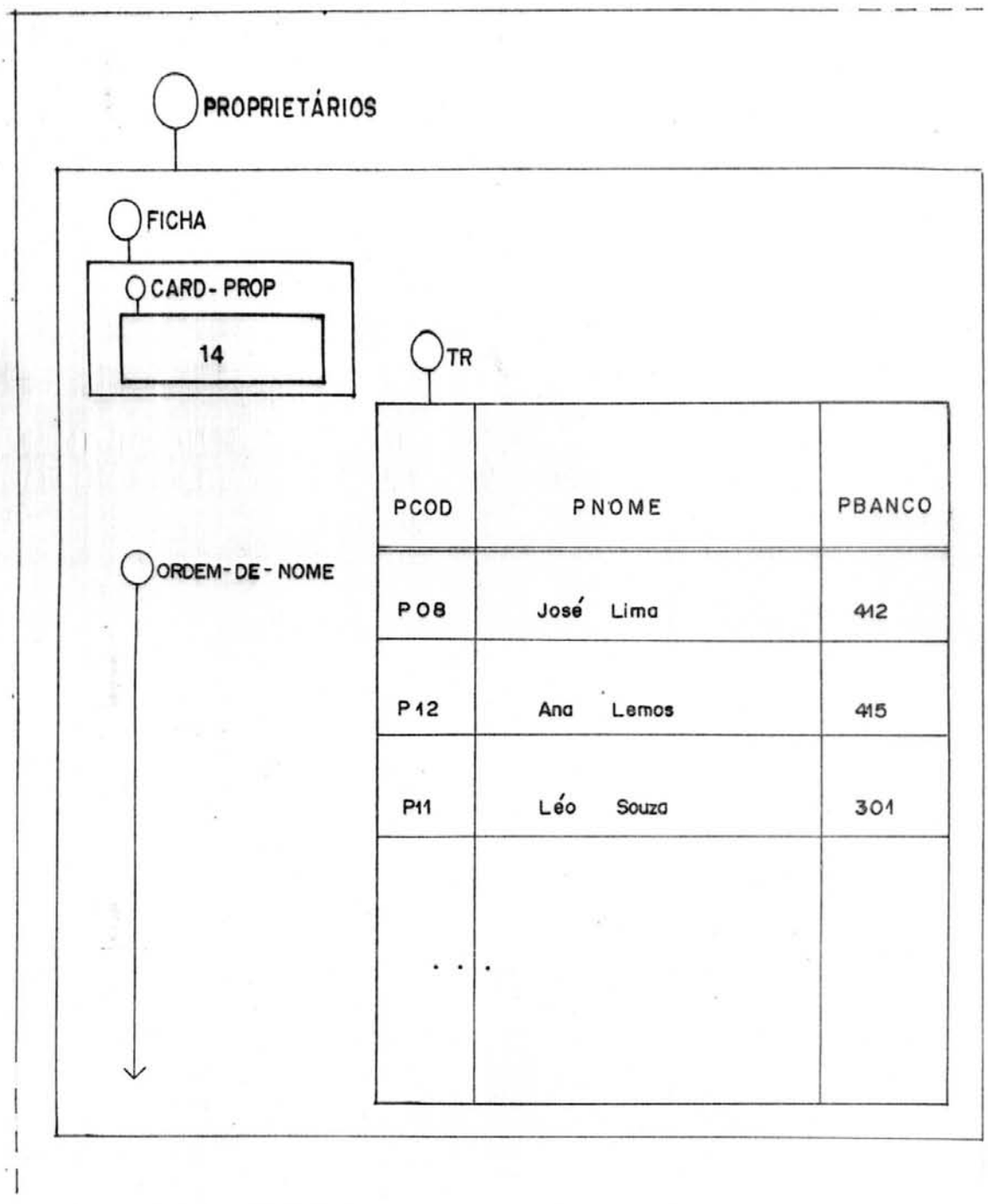


FIGURA 2.3: Arquivo relacional PROPRIETARIOS

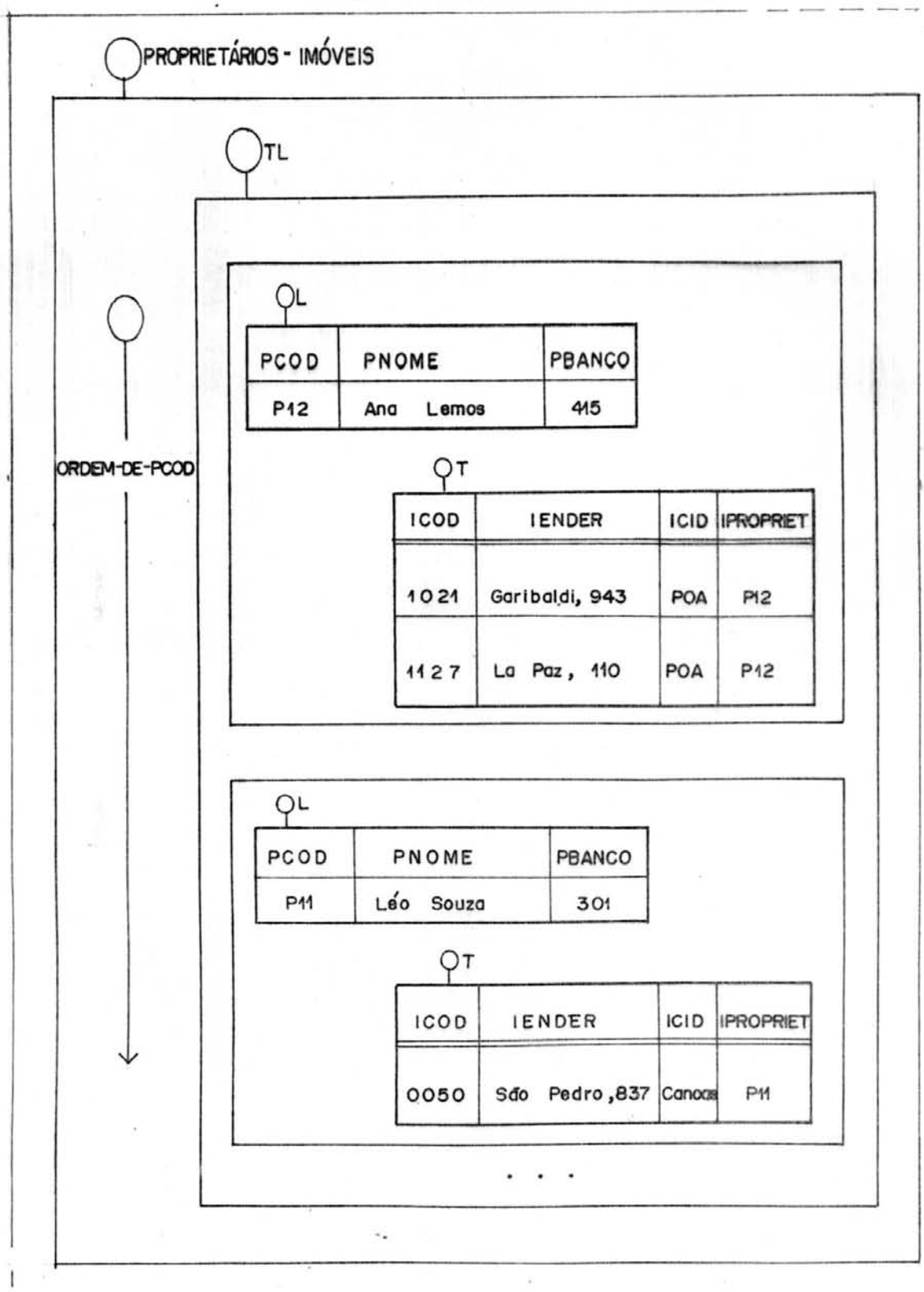


Figura 2.4: Arquivo ligacional PROPRIETARIOS-IMOVEIS

A tabela ligacional de um arquivo ligacional pode ser mantida automaticamente, pelo sistema, ou manualmente, pelo usuário. Se o usuário especifica uma tabela ligacional mantida manualmente, as operações de inclusão, exclusão e alteração de ligações ficam sob sua responsabilidade.

2.3 Referência às construções

Uma construção, em IMC, significa qualquer informação referenciável.

A referência às construções em LOBAN é feita de forma única conhecida por endereçamento de pontos /RIC 81, HEU 81/, através da qual uma construção é localizada em determinado contexto.

Um endereço de ponto, em LOBAN, tem a forma:

critério [. criério]...

onde cada critério serve para seleção de pontos em um nível de agregação de construções. Um critério pode ser, simplesmente, o nome da construção no ponto a selecionar (para selecionar o ponto da tabela relacional de um arquivo relacional, por exemplo), ou pode ser uma expressão booleana complexa, que deverá ser verdadeira para que o ponto seja selecionado.

Utilizando o arquivo relacional IMOVEIS da figura 2.2 como contexto de referência, o endereço

FICHA. CARD-IMOVEIS

identifica o ponto da construção componente da construção FICHA que tem o nome CARD-IMOVEIS, e o endereço

TR. (C IPROPRIET = 'P12')

identifica todas as tuplas de imóveis que possuem a cons-

trução sob nome IPROPRIET igual a 'P12'.

O endereçamento de pontos é utilizado da mesma forma para arquivos relacionais e ligacionais. Considerando como contexto de referência o arquivo ligacional da fig. 2.4 (PROPRIETARIOS-IMOVEIS), o endereço

TL. (C L. PBANCO = 415)

identifica todas as ligações da tabela ligacional (TL) correspondentes a proprietários cujo depósito da arrecadação é feito no banco 415, e a identificação de todos os imóveis da proprietária 'Ana Lemos' é feita por

TL. (C L. PNOME = 'Ana Lemos').T

A obtenção das construções endereçadas é feita através da aplicação de um operador sobre o endereço de ponto especificado /HEU 81/. Estes operadores são denominados operadores de obtenção de construções, e são vistos no próximo item deste capítulo.

2.4 Operadores LOBAN estudados

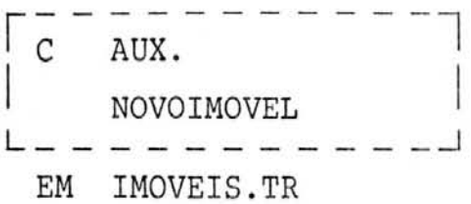
2.4.1 Características gerais das instruções LOBAN

As instruções LOBAN estão classificadas em instruções administrativas e instruções operativas. Entre as instruções administrativas estão as instruções para abrir e fechar arquivos, criar e abolir arquivos e as instruções para controle de fluxo: instrução FAZER PARA CADA, que permite a execução repetitiva de laços, e a instrução FAZER EM CASO, para desvios de fluxo condicionados.

As instruções operativas realizam a representação de construções no canal de saída (REPRESENTAR) e alteração no acervo setorial (INCLUIR, EXCLUIR, SUBSTITUIR).

Um exemplo de instrução operativa é dado por:

INCLUIR



EM IMOVEIS.TR

no qual a construção encontrada no ponto AUX.NOVOIMOVEL é obtida (através do operador C) e incluída na tabela relacional do arquivo IMOVEIS.

A expressão dentro do retângulo tracejado é uma expressão de obtenção de construção, que sempre aparece embutida em instruções operativas, para que a construção obtida seja "materializada" /HEU 81/. Pode ser inserida na base de dados, canal auxiliar ou representada na saída.

É nas expressões de obtenção de construções que reside a potencialidade das instruções de LOBAN, sendo utilizadas tanto na manipulação dos dados como em sua definição. Entre os operadores oferecidos em LOBAN, incluem-se operadores equivalentes aos da álgebra relacional, definida por Codd /COD 70/.

2.4.2 Descrição dos operadores estudados

Os operadores LOBAN aqui estudados são aqueles operadores de obtenção de construções que podem possuir tabelas inteiras como operandos, excetuando-se apenas os operadores CARD (que produz a cardinalidade de uma tabela relacional ou ligacional) e CONT (que conta quantos pontos satisfazem a um certo critério), e podem ser divididos em dois grupos: operadores que produzem como resultado uma tabela (relacional ou ligacional), e operadores que, apesar de usarem tabelas como operandos, produzem como resultado um valor booleano (Verdadeiro ou Falso). No primeiro caso,

estão também incluídos aqueles operadores que produzem como resultado apenas uma tupla ou uma ligação.

São denominadas operações relacionais aquelas operações que produzem, como resultado, tabelas relacionais, e operações ligacionais aquelas operações das quais resultam tabelas ligacionais.

A seguir são descritos brevemente os operadores LOBAN estudados (maiores esclarecimentos podem ser obtidos em /RIC 78, SAN 80, HEU 81/), acompanhados de exemplos referentes às figuras 2.2 a 2.4. Nas descrições, é utilizada a metalinguagem BNF com algumas extensões /GOL 80a/. As metavariáveis empregadas aparecem logo abaixo, com seu significado:

- <end.ponto>: é um endereço de ponto.

- <obter construção>: é uma operação cuja execução resulta na obtenção de uma construção na Zona Intermediária, onde são guardados os resultados intermediários de uma instrução operativa.

- <obter tarel>: equivale a um <obter construção> que produz, como resultado, uma tabela relacional.

- <obter talig>: equivale a um <obter construção> que produz, como resultado, uma tabela ligacional.

- <obter tupla>: equivale a um <obter construção> que produz, como resultado, uma tupla.

- <obter ligação>: equivale a um <obter construção> que produz, como resultado, uma ligação.

- <atributo>: é um atributo de uma tabela relacional.

A indicação $\alpha_{,,,}$ serve para representar a repetição de α, sendo as ocorrências separadas por vírgulas.

Operadores que produzem como resultado tabelas relacionais ou ligacionais:

1 - C <end.ponto>[EM <obter construção>]

É a operação que permite buscar construção da base de dados ou canal auxiliar. O operador C traz uma cópia da construção endereçada para a zona intermediária. Se o termo opcional EM <obter construção> não aparecer, <end.ponto> é um endereço relativo ao contexto; se for empregado, então <end.ponto> é um endereço relativo à construção obtida pela execução da instrução <obter construção>.

Exemplo:

```
C  IMOVEIS.TR.(C  IENDER = 'São Pedro,837')
C  PROPRIETARIOS-IMOVEIS.TL
```

2 - ESTREITAR<obter tarel> {DE|PARA}<atributo>,,,

Produz uma tabela relacional pela eliminação de uma ou mais colunas da tabela relacional obtida pela operação <obter tarel>. É a operação de projeção definida em /COD 70/. A opção DE <atributo>,,, indica quais os atributos que devem ser eliminados e a opção PARA <atributo>,,, indica os atributos que devem permanecer. Pode resultar uma tabela com menor cardinalidade que a de partida, quando for eliminado um atributo que faz parte da chave primária.

Exemplo:

```
ESTREITAR  C PROPRIETARIOS.TR  PARA  PNOME
```

3 - COLEC <end.ponto>[EM <obter construção>]

A execução da operação COLEC resulta em uma tabela relacional ou tabela ligacional, dependendo das construções que foram colecionadas nos pontos endereçados. O termo <end.ponto> deve endereçar tuplas ou ligações. Se for omitido o termo EM <obter construção>, <end.ponto> é um endereço

relativo ao contexto; do contrário, é um endereço relativo à construção obtida pela execução do <obter construção>.

O operador COLEC inclui as características do operador "select" da álgebra relacional.

Exemplo:

```
COLEC  IMOVEIS.TR.(C IPROPRIET = 'P12')
COLEC  PROPRIETARIOS-IMOVEIS.TL.
          (C  L.PBANCO = 415)
```

4 - JUNTAR <obter tarel> COM <obter tarel>
POR {C <atributo> {EXCL|INCL}=
= C <atributo>{EXCL|INCL}},,,

Produz uma tabela relacional a partir de duas tabelas relacionais (extensão do "equi-join" da abordagem relacional /COD 70/). Será incluída uma tupla na tabela resultado a cada vez que as tuplas das tabelas especificadas concordarem nos valores correspondentes a seus respectivos atributos. A tupla resultante da junção possui como componentes imediatos os componentes das duas tuplas, sob seus nomes originais.

O usuário sempre pode especificar se o atributo que foi utilizado para a junção deve ser incluído (INCL) na tabela resultado ou não (EXCL). Em cada comparação, no máximo um dos atributos pode ser excluído.

Exemplo:

```
JUNTAR
  ESTREITAR   C  PROPRIETARIOS.TR
  PARA        PCOD, PNOME
COM   C   IMOVEIS.TR
POR   C   PCOD  =  C  IPROPRIET  EXCL
```

5 - <obter tarel>{UNI|DIF|ISEC}<obter tarel>

É obtida uma tabela relacional pela união (UNI), diferença (DIF) ou intersecção (ISEC) de duas tabelas relacionais. As tabelas relacionais devem possuir tuplas com a mesma composição.

Exemplo:

```
COLEC  C  IMOVEIS.TR.(C  IPROPRIET = 'P12')
DIF
COLEC  C  IMOVEIS.TR.(C  ICOD > 1000)
```

6 - DESAGRUPAR <obter talig>

A execução desta operação resulta em uma tabela relacional construída a partir da tabela ligacional obtida pela execução da operação <obter talig>. O processo de desagrupar constitui-se da formação de conjuntos de tuplas a partir de cada ligação, juntando cada tupla ligada com seu ligante, e da união dos conjuntos formados, obtendo a tabela relacional especificada.

Exemplo:

```
DESAGRUPAR  C  PROPRIETARIOS-IMOVEIS.TL
```

7 - AGRUPAR <obter tarel> POR <atributo>,,,

Constrói uma tabela ligacional a partir da tabela relacional obtida pela execução da operação <obter tarel>.O processo de agrupar constitui-se da formação de conjuntos de tuplas que possuam os mesmos valores sob os atributos indicados, onde cada conjunto dá origem a uma ligação (cujo ligante é composto pelos valores comuns e a tabela de ligados é obtida pelas tuplas constituídas dos demais atributos); as ligações obtidas são colecionadas, formando a tabela ligacional especificada.

Exemplo:

```
AGRUPAR  C  PROPRIETARIOS.TR  POR  PBANCO
```

8 - LIGAR <obter tarel> COM <obter tarel>
POR {C <atributo> = C<atributo>{EXCL|INCL}},,,

Produz uma tabela ligacional a partir da ligação das duas tabelas relacionais obtidas pela execução das operações <obter tarel>. O processo de ligação envolve, como primeiro passo, a formação de uma ligação para cada tupla da tabela relacional obtida pela execução da primeira operação <obter tarel> indicada: o ligante é a tupla obtida e a tabela de ligados é composta por todas as tuplas da outra tabela relacional cujos valores sob os atributos especificados satisfizerem as igualdades indicadas na instrução. As ligações obtidas são colecionadas, formando a tabela ligacional especificada.

Os atributos de ligação sempre fazem parte do ligante, podendo ou não pertencer à tabela de ligados.

Exemplo:

```
LIGAR
  ESTREITAR  C  PROPRIETARIOS.TR
  PARA PCOD, PBANCO

COM
  COLEC  IMOVEIS.TR.(C PCOD > 'P1Ø')

POR   C   PCOD =   C  IPROPRIET  INCL
```

Operadores que produzem como resultado um valor booleano:

9 - <obter tarel>{SUB|SUB=}<obter tarel>

Devolve o valor booleano Verdadeiro se a tabela relacional obtida pela execução da primeira operação <obter

tarel> é subconjunto (SUB) ou subconjunto próprio (SUB=) da tabela relacional obtida pela execução da segunda operação <obter tarel>. Em caso contrário, devolve o valor booleano Falso.

10 - <obter tupla> ELEM <obter tarel>
<obter ligação> ELEM <obter talig>

No primeiro caso, devolve o valor booleano Verdadeiro se a tupla obtida pela execução da operação <obter tupla> é elemento da tabela relacional obtida pela execução da operação <obter tarel>, e o valor booleano Falso, em caso contrário. No segundo caso, devolve o valor booleano Verdadeiro se a ligação obtida pela execução da operação <obter ligação> é elemento da tabela ligacional obtida pela execução da operação <obter talig>, e o valor booleano Falso, em caso contrário.

. . .

O operador COLEC, descrito no item 5 entre os operadores estudados, permite que expressões bem mais complexas apareçam embutidas no <end.ponto> que utiliza. Por exemplo, temos o caso abaixo:

```
COLEC   PROPRIETARIOS-IMOVEIS.
        TL.
        (  (COLEC  T.
                   (C  ICID = 'POA')

           )
           SUB
           (COLEC  T.
                   (C  ICOD < 1000)

           )
        )
```

No exemplo apresentado, são colecionadas todas as ligações da tabela ligacional do arquivo PROPRIETARIOS-

IMOVEIS cujas tabelas de ligados satisfaçam à operação SUB. Assim, são colecionadas todas as ligações em que o conjunto de ligados que representam imóveis situados em 'POA' (ICID= 'POA') seja um subconjunto do conjunto de ligados representando imóveis com código inferior a 1000 (ICOD < 1000).

Devido à possibilidade de ocorrência de operações LOBAN dentro de um <end.ponto>, foram estudados à parte os operadores ESTREITAR, AGRUPAR e COLEC sobre tabelas de ligados, já que estes operadores são os únicos que podem ser aplicados diretamente sobre tabelas de ligados. Os demais operadores LOBAN, quando ocorrentes dentro de um <end.ponto>, usam como operandos tabelas relacionais ou ligacionais, resultantes de operações efetuadas sobre a própria tabela de ligados ou sobre outras operações efetuadas sobre esta tabela, comportando-se da mesma forma como já descritos.

3. O SISTEMA L

3.1 Introdução

O sistema L /HEU 81, GOL 81a, GOL 81b/ implementa a linguagem LOBAN, apresentada no capítulo anterior.

A máquina L é, basicamente, uma máquina de pilha. Esta arquitetura foi escolhida /GOL 81a/ por ser a mais adequada a uma linguagem como LOBAN, onde não existem variáveis declaradas explicitamente, e onde a manipulação de construções temporárias é feita em expressões embutidas umas dentro das outras.

O principal repositório de operandos da máquina L é a pilha. Os operandos são sempre colocados e retirados do topo da pilha. As consultas normalmente são feitas no topo da pilha, mas há situações em que é necessário utilizar operandos que estão mais abaixo.

A grande diferença da máquina L para outras máquinas é a existência de operadores que trabalham diretamente sobre a base de dados interna, ao invés de simples primitivas de Entrada/Saída. Estes operadores certamente serão executados com freqüência e, sendo primitivas, podem ser implementados de forma mais eficiente.

3.2 Arquitetura global do sistema L

A figura 3.1 mostra a arquitetura global do sistema L /HEU 81/, permitindo observar-se sua subdivisão em duas grandes estações: Reconhecedor e Interpretador de pacotes.

Na estação denominada Reconhecedor é feito todo o processamento sobre o texto fonte que independa do conteúdo dos demais canais. Todo o programa LOBAN é processado de

uma só vez, sendo transformado em um programa em código intermediário 2 (no canal CI2)/HEU 81/.

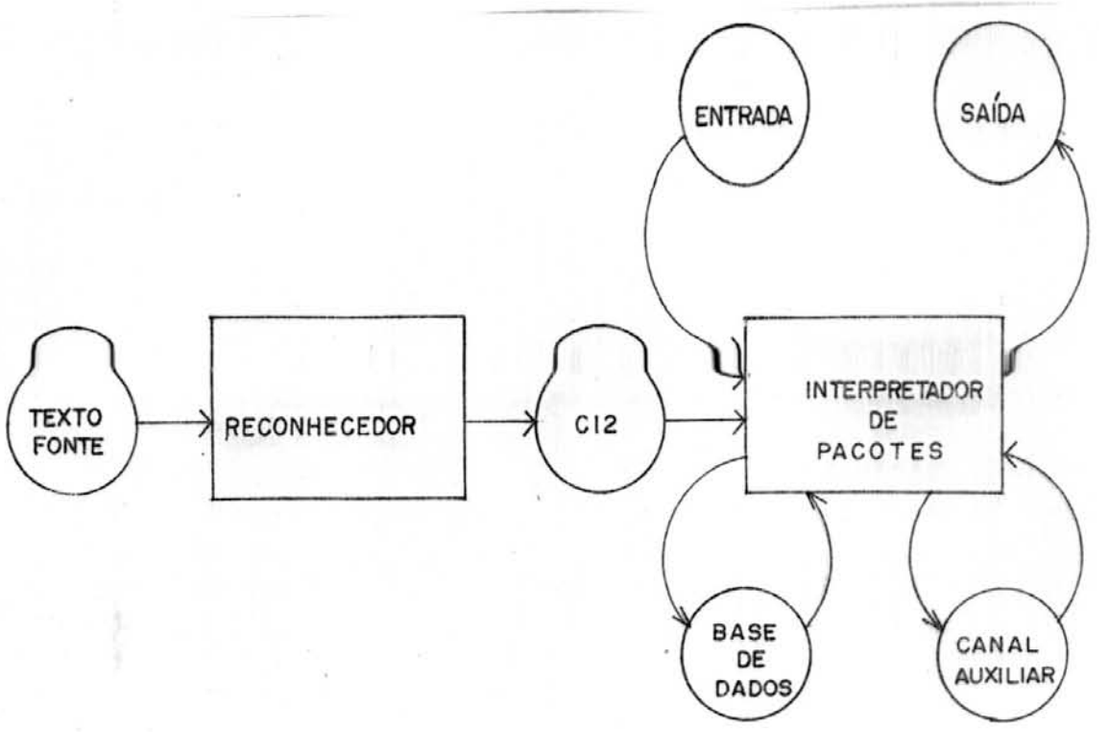


FIGURA 3.1: Arquitetura global do sistema L

Um programa em código intermediário 2 apresenta as seguintes características:

- é constituído de uma lista de unidades sintáticas, que contém referências a uma tabela de literais;
- está em notação polonesa, para facilitar o processamento pela estação seguinte;
- está dividido em "pacotes", que são conjuntos de instruções que podem ser compilados e executados de uma só vez, pois nenhuma instrução do pacote (exceto a última) especifica alteração da estrutura dos canais.

A estação denominada Interpretador de pacotes

é encarregada de retirar os pacotes do canal CI2, analisar cada pacote, levando em conta a descrição da estrutura de dados de cada canal, e executar as instruções contidas no pacote.

A arquitetura do reconhecedor é mostrada na figura 3.2. Maiores detalhes podem ser obtidos em /HEU 81/.

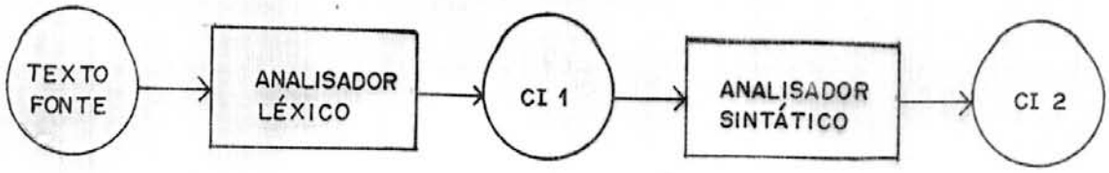


FIGURA 3.2: Arquitetura do Reconhecedor

A figura 3.3 mostra a arquitetura do Interpretador de pacotes.

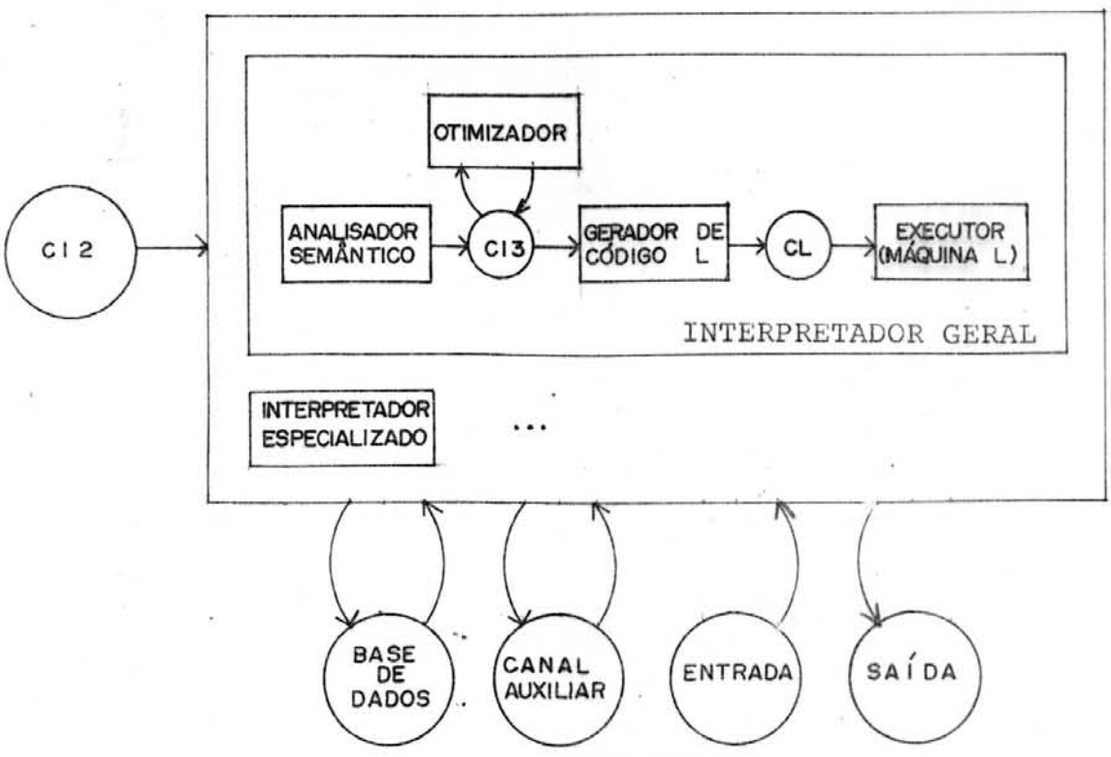


FIGURA 3.3: Arquitetura do Interpretador de pacotes

Os interpretadores especializados têm por função executar as instruções de definição de dados, criação/eliminação de partes da base de dados, arquivamento de cópias ou reconstrução da base de dados. Já o interpretador geral executa as instruções de manipulação de dados, controle de fluxo, definição de transações e de proteção.

O analisador semântico, a primeira estação do Interpretador geral, tem por funções realizar a resolução de nomes e a verificação de compatibilidade entre operador e operandos, e produz como saída o código intermediário 3 (CI3). No código intermediário 3, uma instrução operativa aparece em forma de árvore, e as referências a construções já são feitas através de identificadores internos (o item 3.4 descreve em maior detalhe o CI3).

A próxima estação, o otimizador, visa diminuir o tempo de processamento e o espaço de armazenamento necessários à execução das instruções, através da alteração da seqüência de execução dos operadores e da combinação dos operadores, utilizando técnicas descritas em detalhe em /GOL 80a, GOL 80b/.

O código L (que será executado) é o código objeto de uma máquina abstrata, implementada pelo executor, que recebeu o nome de máquina L. A máquina L é, basicamente,uma máquina de pilha, mais adequada a uma linguagem como LOBAN, onde não existem variáveis declaradas explicitamente e a manipulação de construções temporárias é feita através do embutimento de expressões.

As primitivas de banco de dados /GOL 81b/ são os operadores que trabalham diretamente sobre a base de dados interna, e que diferenciam a máquina L das outras máquinas de pilha. O modelo de dados da base de dados interna é descrito detalhadamente no próximo item deste capítulo.

O gerador de código, usando como entrada o código intermediário 3, tem por função gerar o código L, e se

constitui no tema deste estudo. A geração de código é feita analisando-se cada operação de CI3 quanto à disponibilidade de ordens de classificação e estruturas de acesso auxiliares, buscando utilizar algoritmos que tirem proveito da situação específica em que esta se encontra. Ainda é feita uma coordenação das ordens de classificação dos resultados intermediários, de modo a diminuir o número de classificações e, conseqüentemente, o número de arquivos intermediários gerados.

3.3 Modelo de dados da base de dados interna

3.3.1 Características gerais

O modelo de dados da base de dados interna será descrito com auxílio do conjunto de conceitos IMC.

O conteúdo total da base de dados interna é uma nominação, cujos componentes são áreas, identificadas por números naturais. Alguns dos tipos de áreas existentes são: área de sistema, usada pelo Sistema de Gerência de Banco de Dados - SGBD - para representar as informações sobre o acervo e sobre o mapeamento acervo→conteúdo da base de dados interna, área de dados, para representar os dados armazenados no acervo, e área de trabalho, para representar, temporariamente, os dados armazenados no canal auxiliar ou na zona intermediária, e as informações sobre estes dados.

A área é uma nominação cujos componentes são dos tipos: arquivo, lista, cardinalidade e registro de dados, sendo que os componentes do tipo lista só aparecem em áreas de trabalho.

3.3.2 Arquivos

Um arquivo pode pertencer a um dos seguintes tipos: arquivo de dados, arquivo de inversões, arquivo de as-

sociações por item e arquivo de associações por idreg.

Um arquivo de dados (figura 3.4) é uma nominação em que os componentes são registros de dados e os nomes são números naturais chamados identificadores de registro ou, abreviadamente, idregs.

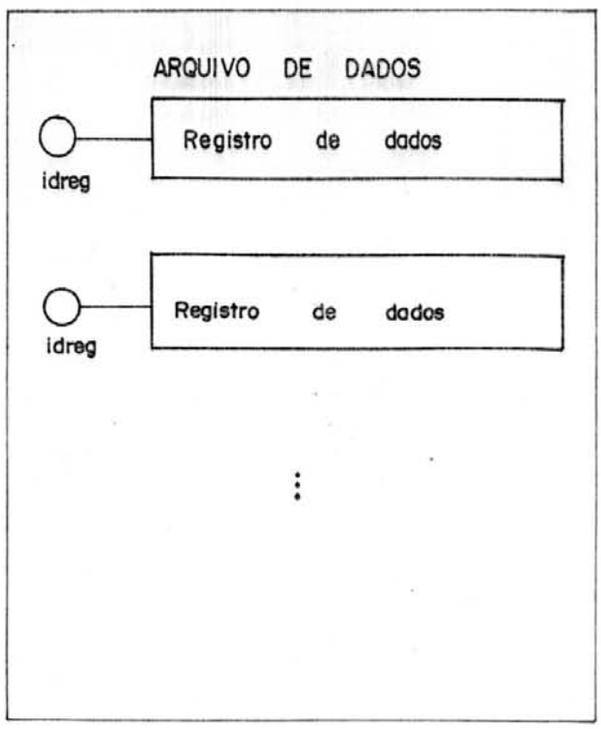


FIGURA 3.4: Arquivo de dados

Um arquivo de dados é utilizado para representar as tabelas relacionais definidas pelo usuário, além de outras estruturas.

Um registro de dados é uma nominação em que os componentes são dos tipos: item de dados ou idreg, e os nomes são números naturais, que representam a posição do componente, em termos de deslocamento, em relação ao início do registro. Os registros de dados em um arquivo não estão classificados.

Os arquivos de inversões (figura 3.5) são cadeias em que os componentes são registros de inversão. Es-

tes arquivos existem sempre conectados a um arquivo de dados.

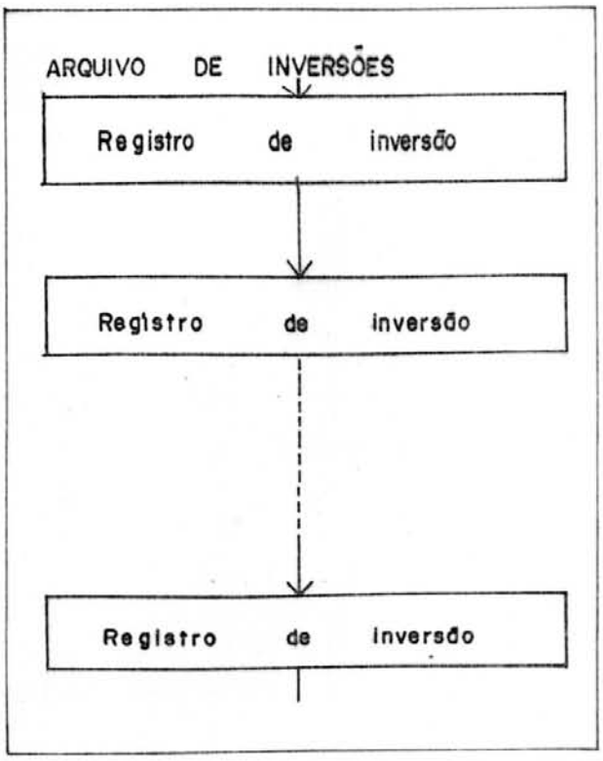


FIGURA 3.5: Arquivo de inversões

Um registro de inversão (figura 3.6) é uma nominação, cujos componentes são dos tipos: registro de dados, cardinalidade e lista de idregs.

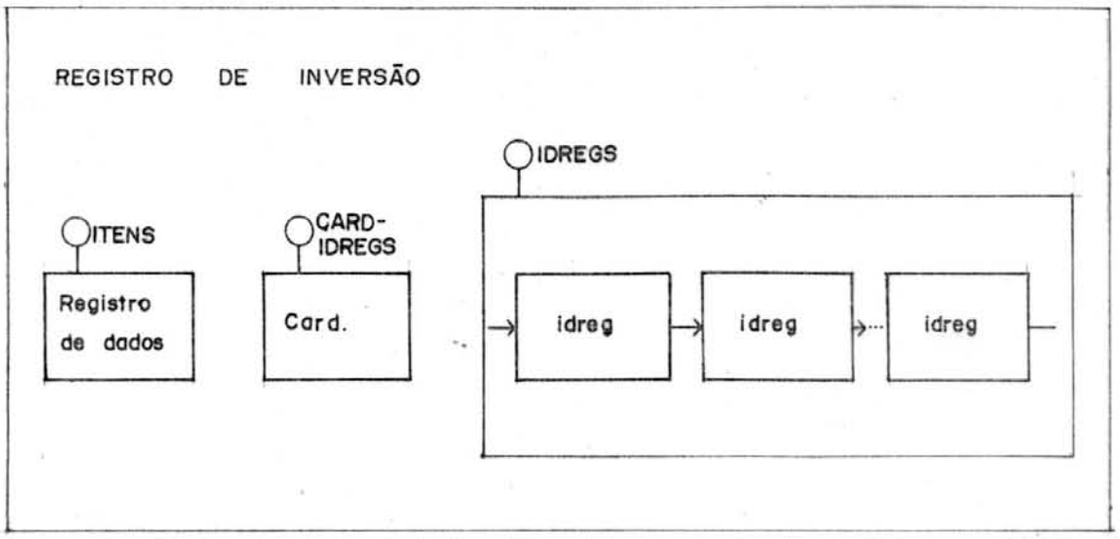


FIGURA 3.6: Registro de inversão

O registro de dados (sob nome ITENS) contém os itens sobre os quais é mantida a inversão, e constitui a chave primária do arquivo de inversões (ou, simplesmente, chave de inversão). O componente sob nome IDREGS contém os idregs de registros(do arquivo de dados associado) que possuem como componentes os valores constantes em ITENS. Sob o nome CARD-IDREGS, é mantida a cardinalidade da lista de idregs. Os registros de inversão são mantidos em ordem pela chave primária do arquivo de inversões.

Os arquivos de inversões são mantidos pelo SGBD, podendo existir, no máximo, dois arquivos de inversões associados a um arquivo de dados: um deles mantido pela chave primária do arquivo de dados (neste caso, CARD-IDREGS terá sempre valor 1) e o outro para representar a relação de ordem definida pelo usuário (ver capítulo 2, item 2.1).

Os arquivos de associações podem ser de dois tipos: arquivo de associações por item e arquivo de associações por idreg. São utilizados para representar as tabelas ligacionais especificadas pelo usuário.

Os arquivos de associações por item/por idreg são cadeias em que os componentes são registros de associação por item/por idreg. Os registros de associação são mantidos em ordem pela chave primária do arquivo de associações.

Um arquivo de associações por item representa uma tabela ligacional resultante de uma conexão fornecida pelo usuário, mantida automaticamente pelo SGBD.

O arquivo de associações por item associa dois arquivos de dados, um cujos registros representam ligantes e outro cujos registros representam ligados.

Um registro de associação por item(figura 3.7) re

presenta a ligação entre um registro de dados (cujo idreg aparece sob o nome LIGANTE) e vários outros (LIGADOS), que possuem valores comuns para os componentes especificados. Estes valores comuns (valores de ligação) aparecem sob o nome ITENS, e na construção sob o nome CARD-LIGADOS é mantido o número de registros ligados.

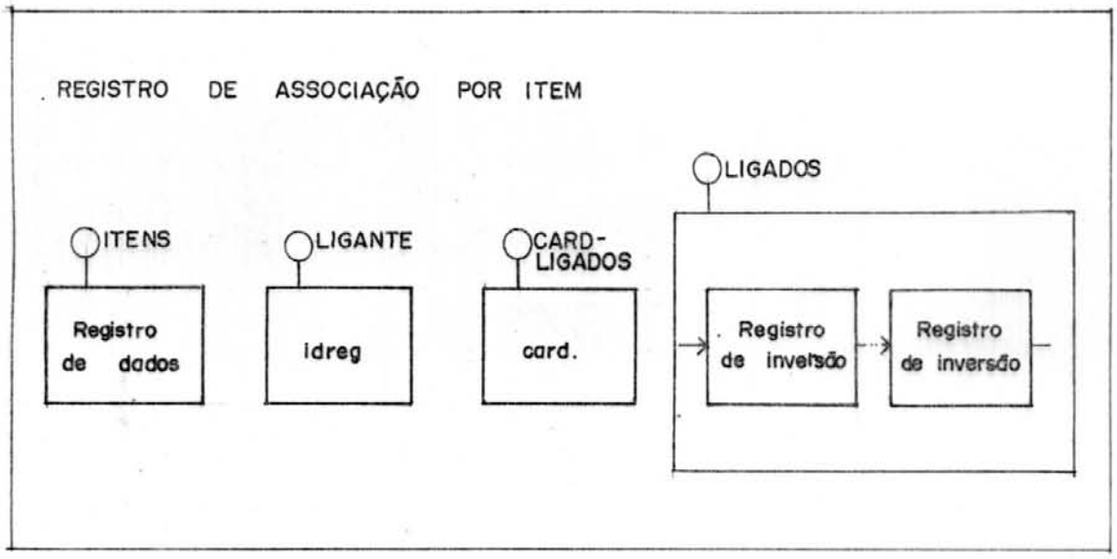


FIGURA 3.7: Registro de Associação por Item

O componente sob o nome LIGADOS é mantido como um arquivo de inversões para o arquivo de dados que fornece os ligados, sendo que os itens de inversão são os que compõem a chave primária do arquivo de dados.

A chave primária do arquivo de associações por item é ITENS.

Um arquivo de associações por idreg é utilizado pelo SGBD para representar uma conexão manual entre dois arquivos de dados (controlada pelo usuário). Neste caso, não existem valores de ligação entre ligantes e ligados: uma ligação é criada pelo usuário através do fornecimento de um ligante com os ligados correspondentes.

Os ligantes e os ligados são mantidos pelo SGBD

em arquivos de dados distintos. O arquivo de dados contendo ligantes possui, em cada registro de dados, apenas os componentes do próprio ligante. O arquivo de dados contendo ligados possui um componente adicional, em cada registro de dados, que informa a qual ligante este registro está associado: este componente é o idreg do ligante associado (figura 3.8).

A chave primária do arquivo de dados contendo ligados de um arquivo de associações por idreg é a concatenação do idreg do ligante com a chave primária da tabela de ligados, definida pelo usuário.

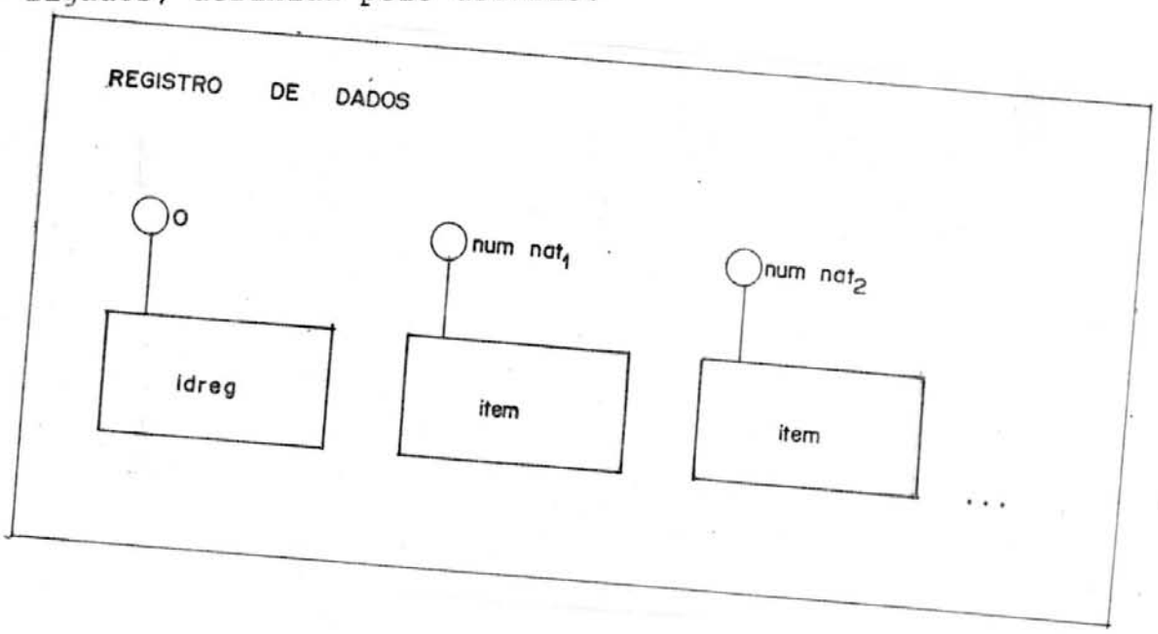


FIGURA 3.8: Ligado de um arquivo de associações por idreg

Um registro de associação por idreg (figura 3.9), assim como um registro de associação por item, representa a associação entre registros de dois arquivos de dados. Neste caso, porém, não existem valores comuns que fazem a ligação. O componente LIGANTE contém o idreg do registro ligante, e CARD-LIGADOS contém o número de ligados correspondentes a esse ligante. O componente LIGADOS é mantido como um arquivo de inversões para o arquivo de dados que contém os ligados, pelos itens que formam a chave primária da tabela de ligados de uma ligação, conforme especificado pelo usuá

rio.

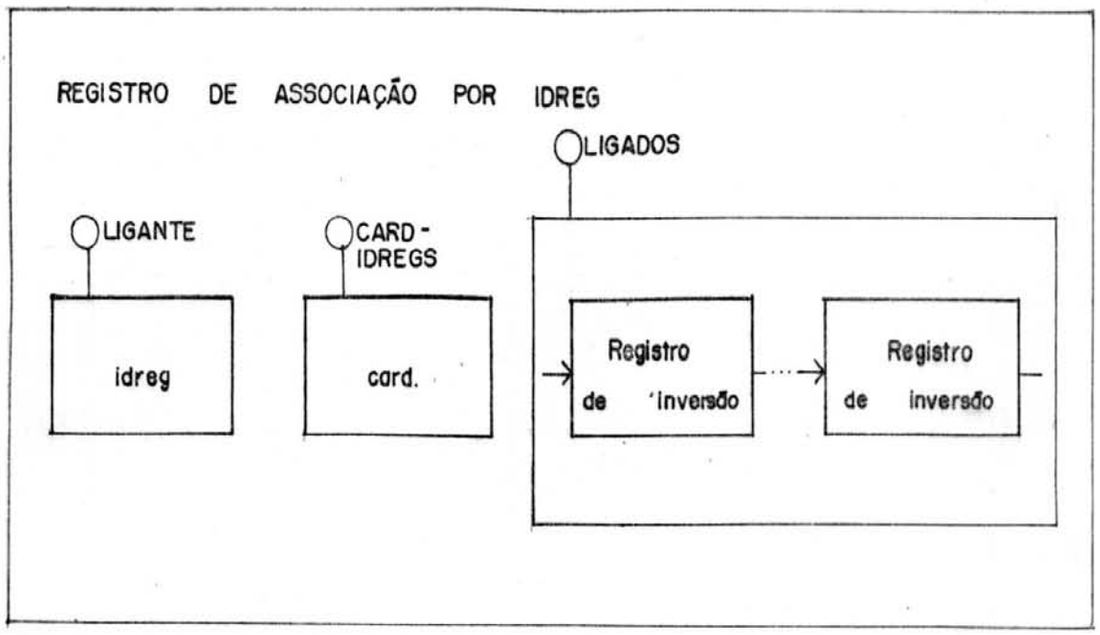


FIGURA 3.9: Registro de associação por idreg

A chave primária de um arquivo de associações por idreg é LIGANTE.

3.3.3 Listas

Uma lista é uma cadeia, em que os componentes são registros de dados.

As listas de dados são utilizadas para representar as tabelas relacionais ou ligacionais mantidas na zona intermediária ou no canal auxiliar. Os registros de dados estão em ordem cronológica na lista, isto é, o primeiro registro da cadeia foi o primeiro a ser inserido, e, as inserções são sempre realizadas no fim da cadeia.

A representação das tabelas ligacionais mantidas na zona intermediária ou no canal auxiliar é feita através de duas listas de dados: uma lista de ligantes e uma lista de ligados.

Cada registro de dados da lista de ligantes con-

têm, além dos itens de dados do ligante, dois itens referentes às posições onde se encontram o primeiro e o último ligados dessa ligação, respectivamente, na lista de ligados. A indicação de tabela de ligados vazia é feita através de valores inválidos para as posições de primeiro e último ligados da ligação, junto ao ligante.

3.4 Código intermediário 3

Sendo o CI3 o código de entrada para o gerador de código L, é interessante que seja mostrado em maior detalhe.

São aqui descritos os operadores de CI3 para os quais foi estudada a geração de código L, sendo apresentados em notação polonesa pós-fixada.

As metavariáveis utilizadas na descrição do CI3, com seu significado, são, respectivamente:

- <id arq dados>: identificador de arquivo de dados
- <id arq assoc>: identificador de arquivo de associações
- <expr sel reg>: expressão de seleção de registro
- <expr sel reg ligados>: expressão de seleção de ligados de uma associação
- <id reg dados>: identificador de registro de dados
- <id reg assoc>: identificador de registro de associação
- <id item>: identificador de um item, dentro de um registro de dados
- <num de itens>: número de itens

Operadores de CI3 que produzem como resultado listas de dados na zona intermediária:

1 - COLR: <id arq dados><expr sel reg> COLR

Coleciona registros de um arquivo/lista de dados que atendem a uma expressão de seleção, produzindo uma lis-

ta de dados contendo os registros colecionados.

2 - COLRU: <id arq dados><expr sel reg> COLRU

Resulta de uma operação C <end.ponto> (ver capítulo 2, item 2.3.2) cujo <end.ponto> pode ser satisfeito por, no máximo, um registro de dados. Coleciona (obtém) nenhum ou um único registro de dados de um arquivo/lista de dados, através de uma expressão de seleção que envolve critérios de igualdade sobre os itens da chave primária, todos ligados pelo operador lógico ET (conjunção); podem existir ainda outras subexpressões, envolvendo outros itens, todas elas ligadas por ET ao critério de seleção sobre a chave primária. O registro colecionado é colocado no topo da pilha.

3 - ESTR: <id arq dados><id item>,,, <num de itens> ESTR

Obtém uma lista de dados através da eliminação de itens dos registros de dados de um arquivo/lista de dados (é utilizado quando a operação de estreitamento elimina itens da chave primária do arquivo/lista de dados).

4 - ESTRND: <id arq dados><id item>,,, <num de itens> ESTRND

Obtém uma lista de dados através da eliminação de itens dos registros de dados de um arquivo/lista de dados (é utilizado quando não é exigida eliminação de duplicatas).

5 - COLA: <id arq assoc><expr sel reg> COLA

Coleciona registros de associação de um arquivo de associações ou coleciona associações a partir de uma lista de ligantes + lista de ligados, representando uma tabela ligacional mantida na zona intermediária ou no canal auxi-

liar. A expressão de seleção pode ser bastante complexa, inclusive contendo outras operações LOBAN (em CI3).

6 - COLAU: <id arq assoc><expr sel reg> COLAU

Resulta de uma operação C cujo <end.ponto> pode ser satisfeito por, no máximo, uma associação. Coleciona (obtém) nenhum ou um único registro de associação de um arquivo de associações ou coleciona nenhuma ou única associação a partir de uma lista de ligantes + lista de ligados, representando uma tabela ligacional mantida na zona intermediária ou no canal auxiliar. A expressão de seleção envolve critérios de igualdade sobre os itens da chave primária do arquivo de associações, todos ligados pelo operador lógico ET, podendo existir ainda outras subexpressões, todas elas ligadas por ET ao critério de seleção sobre a chave primária. Outras operações LOBAN também podem ser necessárias para resolver um critério de seleção.

7 - COLL: <id arq assoc><expr sel reg>
<expr sel reg ligados> COLL

Coleciona registros ligados a partir de um arquivo de associações, ou a partir de uma lista de ligantes + lista de ligados mantida na zona intermediária ou canal auxiliar. Utiliza duas expressões de seleção: uma delas selecionando os registros de associação, e a outra selecionando ligados dentro de um registro de associação.

8 - AGRUPAR: <id arq dados><id item>,,,
<num de itens> AGRUPAR

Executa a função do operador AGRUPAR. Gera uma lista de ligantes + lista de ligados, representando uma tabela ligacional na zona intermediária, a partir de um arquivo/lista de dados.

9 - DESAGRUPAR: <id arq assoc> DESAGRUPAR

Executa a função do operador DESAGRUPAR, produzindo uma lista de dados a partir de um arquivo de associações ou de uma lista de ligantes + lista de ligados.

10 - UNI:<id arq $dados_1$><id arq $dados_2$> UN

Executa a função do operador UNI, produzindo uma lista de dados a partir da união de dois arquivos / listas de dados.

11 - DIF: <id arq $dados_1$><id arq $dados_2$> DIF

Executa a função do operador DIF, produzindo uma lista de dados a partir da diferença entre dois arquivos/listas de dados.

12 - ISEC: <id arq $dados_1$><id arq $dados_2$> ISEC

Executa a função do operador ISEC, produzindo uma lista de dados a partir da interseção entre dois arquivos/listas de dados.

13 - JUNT: <id arq $dados_1$><id arq $dados_2$>
{<id item arq_1>{$\frac{\text{EXCL}}{\text{INCL}}$}<id item arq_2>{$\frac{\text{EXCL}}{\text{INCL}}$}}$_{,,,}$
<num de itens> JUNT

Executa a função do operador JUNTAR, produzindo uma lista de dados a partir da junção de dois arquivos/listas de dados.

14 - LIG: <id arq $dados_1$><id arq $dados_2$>
{<id item arq_1><id item arq_2>{$\frac{\text{EXCL}}{\text{INCL}}$}}$_{,,,}$
<num de itens> LIG

Executa a função do operador LIGAR, produzindo uma lista de ligantes + lista de ligados a partir de

dois arquivos/listas de dados.

Operadores de CI3 que produzem como resultado um valor booleano:

15 - SUB,SUB=: <id arq dados$_1$><id arq dados$_2$>{SUB= / SUB}

Executam as funções dos operadores SUB e SUB=, (conforme descritos no capítulo 2), verificando se um arquivo/lista de dados é subconjunto (SUB) ou subconjunto próprio (SUB=) de outro arquivo/lista de dados, e devolvendo o valor booleano Verdadeiro ou Falso.

16 - ELEM: <id reg dados><id arq dados> ELEM |
<id reg assoc><id arq assoc> ELEM

Executa a função do operador ELEM, verificando se um registro de dados é elemento de um arquivo/lista de dados, ou se uma associação é elemento de um arquivo de associações ou de uma lista de ligantes + lista de ligados mantidos na zona intermediária ou canal auxiliar, devolvendo o valor booleano Verdadeiro ou Falso.

No código intermediário 3, as informações sobre uma instrução operativa de LOBAN a ser executada estão representadas através de uma árvore, que reflete a seqüência de operações fornecidas pelo usuário. Esta árvore passará aqui a denominar-se árvore de operações, e será percorrida com o objetivo de determinar os algoritmos a serem aplicados e as ordens de classificação mais interessantes a cada operação.

A figura 3.10 mostra a árvore de operações, em CI3, para a seqüência de operações LOBAN dadas a seguir (ver figura 2.2).

```
ESTREITAR

    JUNTAR
      (ESTREITAR  C  IMOVEIS.TR  PARA ICOD,IPROPRIET)
    COM
      (COLEC  PROPRIETARIOS.TR.(C  PBANCO = 415))
    POR    C  ICOD =  C  PCOD  EXCL

PARA     ICOD, PNOME
```

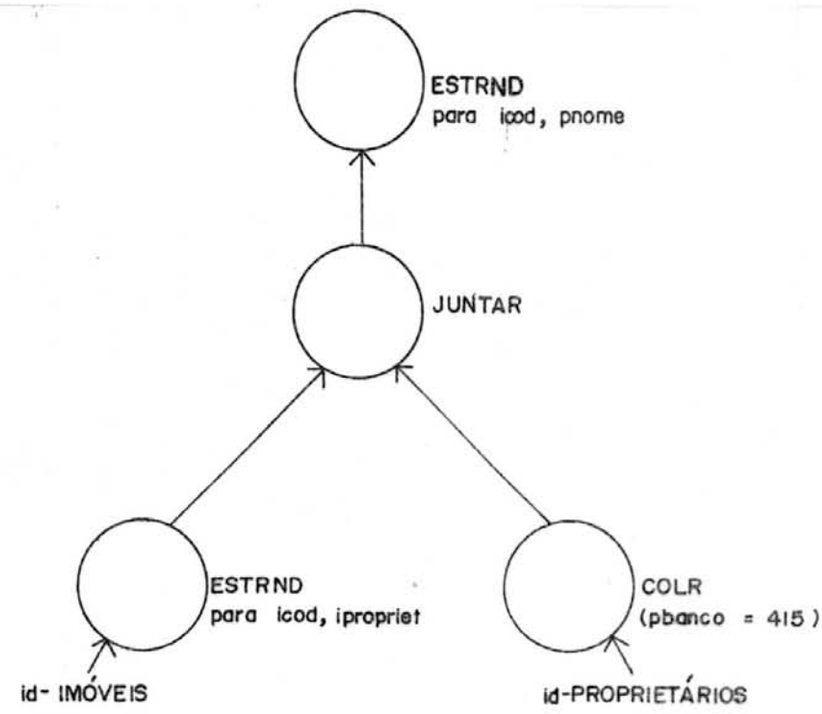


FIGURA 3.10: Esboço de uma árvore de operações em CI3

Na seqüência de operações de CI3 geradas, supõe-se que a chave primária da tabela relacional do arquivo IMOVEIS é constituída pelo item ICOD (dando razão à operação CI3 ESTRND, já que este estreitamento não necessita eliminar duplicatas). Já a chave primária de uma tabela resultante da operação JUNTAR é formada pela concatenação da chave primária das duas tabelas juntadas. Supondo que a chave primária da tabela relacional do arquivo PROPRIETA-

RIOS seja PCOD, então o resultado do último estreitamento mantém, também, a chave primária da tabela resultante da junção (gerando-se, novamente, a operação ESTRND).

3.5 Primitivas de Banco de Dados

Os operadores de código L são denominados primitivas de banco de dados, e implementam uma interface navegacional sobre a base de dados, através da qual são transmitidos registros.

As primitivas a serem utilizadas foram divididas em grupos, como segue:

1 - Primitivas sobre arquivos
2 - Primitivas sobre cursores
3 - Primitivas para recuperação de itens/registros
4 - Primitivas para inclusão/exclusão e alteração de registros.

. . .

1 - Primitivas sobre arquivos

- Abrir arquivo
- Fechar arquivo

2 - Primitivas sobre cursores

Os cursores são identificadores que podem ser associados a um arquivo (de dados, de inversões ou de associações) ou lista de dados, possibilitando que seus registros sejam recuperados um a um. Nos arquivos e listas de dados, o posicionamento inicial do cursor é no primeiro registro. Nos arquivos de inversões e associações o cursor pode ser colocado em um registro específico (dado um valor para o componente que é chave primária).

- Estabelecer cursor sobre um arquivo de dados.

- Estabelecer cursor sobre uma lista de dados.

- Estabelecer cursor sobre um arquivo de inversões, sendo fornecidos os valores para o componente sob nome ITENS.

- Estabelecer cursor sobre o arquivo de inversões que representa os ligados de um registro de associação.

- Estabelecer cursor sobre um arquivo de associações, sendo fornecidos os valores para o componente que é chave primária (ITENS, quando for um arquivo de associações por item, ou LIGANTE, quando for um arquivo de associações por idreg).

- Mover cursor.

- Abandonar cursor.

3 - Primitivas para recuperação de itens/registros

Registros de dados:
- Buscar registro de dados com idreg fornecido.
- Buscar registro de dados correspondente à ficha de um arquivo.
- Buscar registro de dados conforme indicado por um cursor.
- Buscar registro de dados de uma lista de dados, com posição fornecida.

Registros de inversão:
- Buscar registro de inversão indicado por um cursor.
- Buscar registro de inversão que possua os valores de itens fornecidos (componente sob nome

ITENS).

Registro de associação:

- Buscar registro de associação por item conforme indicado por um cursor.
- Buscar registro de associação por idreg conforme indicado por um cursor.
- Buscar registro de associação por item de acordo com os valores de itens fornecidos (componente sob nome ITENS).
- Buscar registro de associação por idreg, sendo fornecido o idreg do ligante.

4 - Primitivas para inclusão/exclusão e alteração de registros

- Incluir registro de dados em um arquivo de dados.
- Incluir registro de dados em uma lista de dados.
- Incluir o idreg de um registro de dados em um arquivo de inversões.
- Incluir registro ligado em um registro de associação, sendo fornecidos os itens que compõem a chave primária do arquivo de associações.
- Excluir registro de dados de um arquivo de dados.
- Excluir idreg (de um registro de dados) de um registro de inversão.
- Excluir registro ligado de um registro de associação, sendo fornecido o ligante.
- Substituir registro de dados em um arquivo de dados.

Além destas primitivas existem outras, como uma primitiva de classificação e primitivas para reconstrução, transações e proteção.

4. RECUPERAÇÃO DE REGISTROS DE DADOS A PARTIR DE LISTAS DE IDREGS CLASSIFICADAS

4.1 Introdução

Em vários dos algoritmos apresentados, são criadas listas temporárias de idregs, onde cada idreg reflete a página em que se encontra determinado registro. Estas listas são provenientes, em sua maioria, de pesquisas ou comparações em arquivos de inversões ou associações, com o objetivo de relacionar um conjunto de registros (tuplas) resultantes de uma certa operação.

A recuperação dos registros correspondentes, a partir de uma lista temporária de idregs, é denominada materialização dessa lista de idregs.

4.2 Classificação de listas de idregs

4.2.1 Materialização por idregs não classificados

Considerando que a busca aleatória de um registro implica, geralmente, um acesso a disco, o número de acessos necessários para recuperar todos os registros de uma lista de idregs, se esta não estiver classificada, poderá ser igual ao número de idregs da lista (pior caso). Para este caso, pode-se ver, pela tabela 4.1, que o fator de bloco não tem influência sobre o número de acessos necessários para recuperação de registros, já que de cada bloco trazido para a memória só um registro é utilizado.

O arquivo de dados idealizado para confecção das tabelas aqui apresentadas contém R = 15 000* registros. I

---

* Analisando-se as características do equipamento disponível para implementação e considerando-se registros de dados de, no mínimo, 50 bytes, o maior tamanho permitido para um arquivo fica em torno de 15 000 registros.

representa o número de idregs da lista temporária. O percentual de registros a serem buscados é representado por I/R. O número total de páginas (blocos) ocupados pelo arquivo de dados é representado por B. É calculado o número de acessos necessários para materialização de listas de idregs considerando três blocagens diferentes do arquivo de dados.

Deve-se observar que o melhor caso na recuperação de registros ocorre quando, apesar de a lista de idregs não ter sido classificada, os registros de dados a serem buscados se encontram fisicamente próximos, de tal maneira que todos os registros trazidos em um bloco são utilizados. Neste caso, sendo f_b o fator de bloco do arquivo de dados, o número de acessos necessários é dado por I/f_b, que representa o número mínimo de páginas contendo os registros a serem buscados.

Entretanto, este caso é muito difícil de acontecer, na prática. Considerando um aproveitamento de 50% de cada página trazida, o número de acessos necessários passa a ser dado por $I/(f_b \times 0.50)$, conforme mostrado na tabela 4.2, que constitui uma situação um pouco mais próxima da realidade.

O gráfico da figura 4.1 mostra o número de acessos estimado conforme as tabelas 4.1 e 4.2.

TABELA 4.1: Número de acessos para recuperação por idregs não classificados (pior caso)

f_b \ I/R	I= 1500 0,10	I= 4500 0,30	I= 7500 0,50	I=12000 0,80	I=15000 1,00
$B = 7500^2$	1500	4500	7500	12000	15000
$B = 3000^5$	1500	4500	7500	12000	15000
$B = 1500^{10}$	1500	4500	7500	12000	15000

TABELA 4.2: Número de acessos para recuperação por idregs não classificados (aproveitamento de 50% de cada bloco buscado)

f_b \ I/R	I= 1500 0,10	I= 4500 0,30	I= 7500 0,50	I=12000 0,80	I=15000 1,00
$B = 7500^2$	1500	4500	7500	12000	15000
$B = 3000^5$	600	1800	3000	4800	6000
$B = 1500^{10}$	300	900	1500	2400	3000

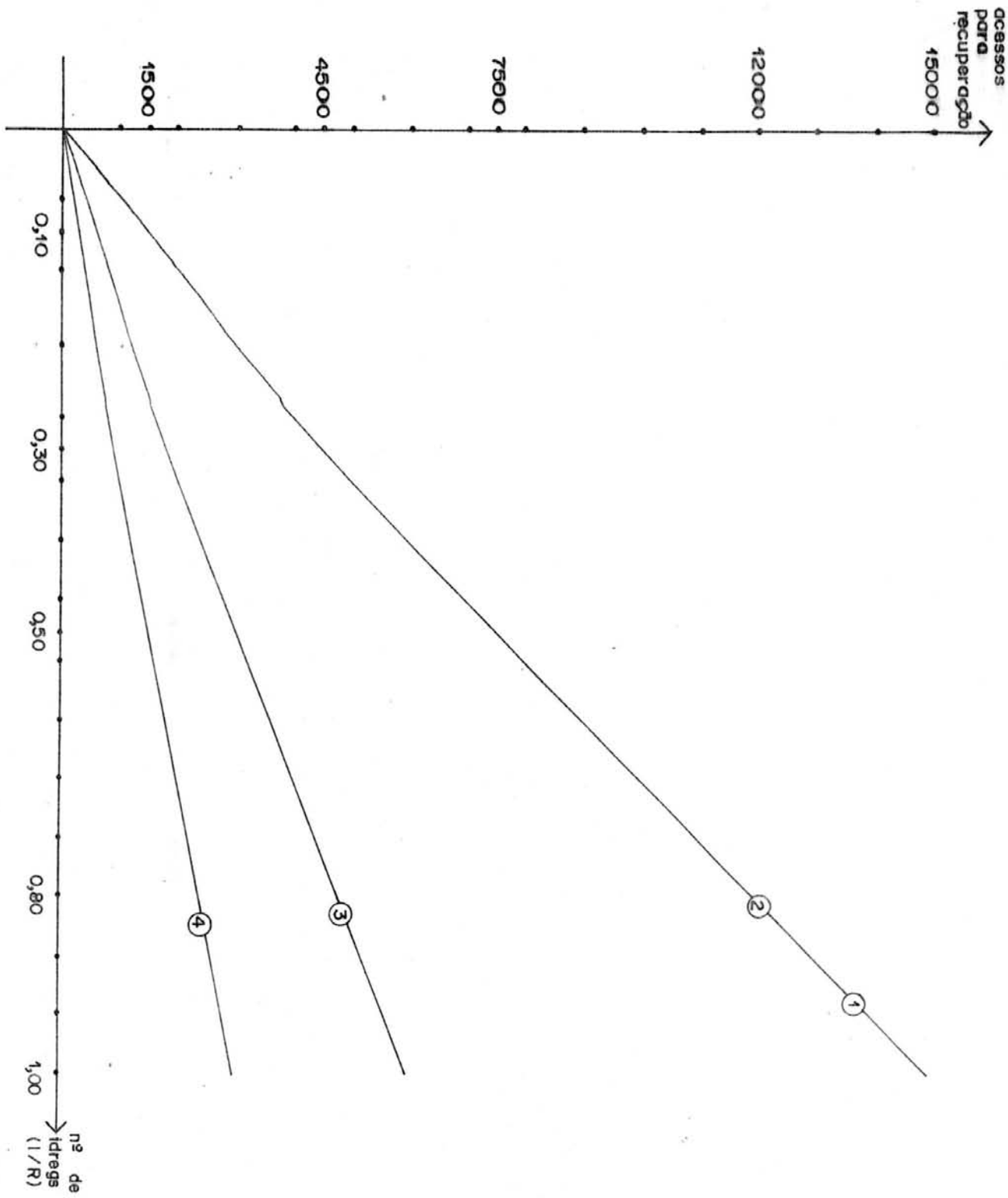


FIGURA 4.1: Recuperação por idregs não classificados

4.2.2 Número de acessos para classificação de idregs

Usando-se um algoritmo de classificação/intercalação (SORT/MERGE do tipo "8-way") para classificação de uma lista de registros contendo idregs, o número de acessos será:

$$2M + 2M \times \text{Máx} \{ \lceil \log_8 M/Q \rceil , 1 \}$$

onde:

M - é o número total de páginas que constituem esta lista temporária de idregs.

Q - é o número de páginas transferidas pelo algoritmo de classificação/intercalação, a cada vez, entre a memória principal e a memória secundária.

A primeira parcela (2M) do número de acessos é referente à leitura inicial dos registros da lista a classificar e gravação final da lista classificada.

A segunda parcela ($2M \times \text{Máx} \{ \lceil \log_8 M/Q \rceil , 1 \}$) se refere aos acessos feitos especificamente pelo algoritmo de classificação, e foi adaptada das conclusões de /BLA 77/ para atender também à classificação de listas com pequeno número de registros. M/Q dá o número de cadeias de Q páginas a serem classificadas isoladamente; $\log_8 M/Q$ dá o número de níveis da árvore de intercalação, considerando que, no máximo, 8 cadeias podem ser intercaladas de cada vez, para produzir uma nova cadeia classificada (sendo este um "8-way merge"). Então até 8 cadeias iniciais podem ser intercaladas de uma única vez. Com até 64 cadeias iniciais pode ser produzido o resultado final com dois níveis de intercalação, e assim por diante.

A função $\text{Máx}\{ \lceil \log_8 M/Q \rceil , 1 \}$ é utilizada para prever os casos de arquivos com menos de 8 cadeias iniciais a classificar ($M/Q < 8$), onde a função logaritmo produziria valores inferiores a 1.

A necessidade de ler e gravar todo o arquivo, a cada nível de intercalação, faz aparecer o fator 2M multiplicando o número de níveis de intercalação.

Além dos acessos gastos pelo algoritmo de classificação/intercalação, é necessário computar o número de acessos exigidos para transferência deste programa para a memória (o que fica em torno de 70 acessos).

Dependendo do tamanho da lista de idregs a ser classificada, pode ser usado um algoritmo de classificação interna, que realiza toda a classificação/intercalação na própria memória principal. Na implementação em vigor, está disponível uma área que permite a classificação interna de até 4000 bytes, sem a necessidade de acesso a meio externo. Recomenda-se, assim, o uso de primitivas específicas para classificação de listas de idregs, que mantenham na memória principal listas com menos de 4000 bytes, e as transfiram para memória secundária, quando superarem este tamanho.

No caso de uma classificação interna, não há necessidade de acessos adicionais: bastam os acessos para a sobreposição do programa de classificação.

Sendo considerada a disponibilidade de memória na implementação do algoritmo de classificação/intercalação /CAR 81/, ficou estabelecido um total de 11 páginas transferidas, a cada vez, entre a memória principal e a secundária (logo, Q = 11).

Por último, deve-se observar que, por características da implementação, uma página tem sempre tamanho de 512 bytes (equivalente ao tamanho de um bloco, na memória), e um idreg ocupa 2 palavras para ser armazenado (no computador LABO 8Ø34, o equivalente a 4 bytes), o que permite classificação interna para até 1000 idregs.

Feitas essas observações, é mostrado na tabela 4.3 o número de acessos para classificação de listas de

idregs.

TABELA 4.3: Número de acessos para classificação, conforme a cardinalidade da lista de idregs

TAMANHO (em nº de idregs)	M (nº de páginas)	ACESSOS
< 1000	-	70
1000	8	102
1500	12	118
2000	16	134
3000	24	166
4000	32	198
4500	36	214
5000	40	230
6000	47	258
7000	55	290
7500	59	306
8000	63	322
9000	71	354
10000	79	386
11000	86	414
12000	94	634
13000	102	680
14000	110	730
15000	118	778

A figura 4.2 mostra o gráfico obtido com os dados da tabela 4.3.

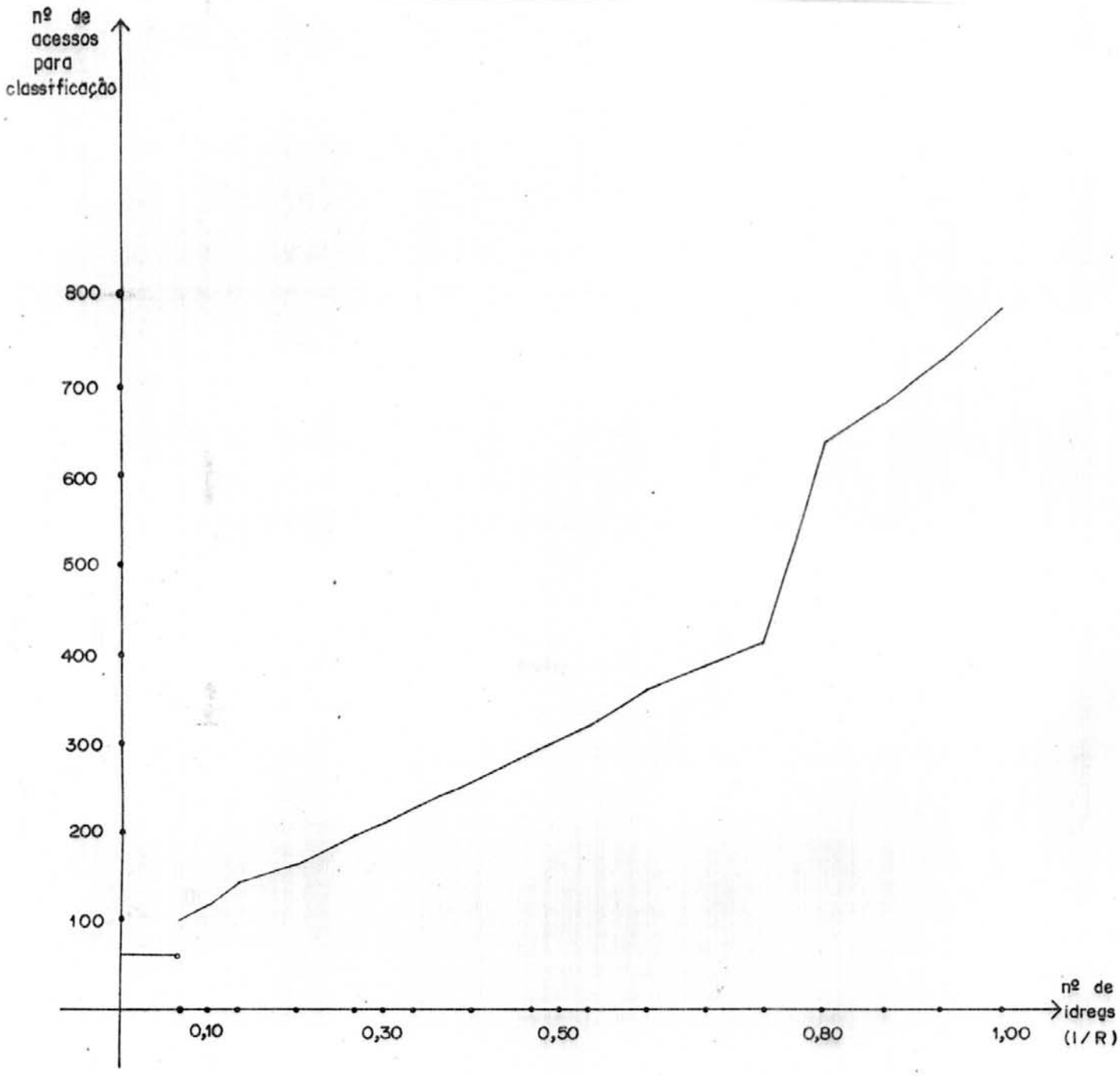


FIGURA 4.2: Número de acessos para classificação de listas de idregs

4.2.3 Comparação das duas alternativas

A tabela 4.4 mostra o número de acessos para recuperação de registros a partir de uma lista de idregs classificados, no pior caso. Este é o caso em que, apesar de classificados, os idregs estão distribuídos de tal forma que o maior número de páginas deve ser acessado.

O melhor caso de recuperação por idregs classificados é equivalente ao melhor caso de recuperação por idregs não classificados, e é apresentado na tabela 4.5.

TABELA 4.4: Número de acessos para recuperação por idregs classificados (pior caso)

f_b \ I/R	I= 1500 0,10	I= 4500 0,30	I= 7500 0,50	I=12000 0,80	I=15000 1,00
$B = 7500^{2}$	1500	4500	7500	7500	7500
$B = 3000^{5}$	1500	3000	3000	3000	3000
$B = 1500^{10}$	1500	1500	1500	1500	1500

TABELA 4.5: Número de acessos para recuperação por idregs classificados (melhor caso)

f_b \ I/R	I= 1500 0,10	I= 4500 0,30	I= 7500 0,50	I=12000 0,80	I=15000 1,00
$B = 7500^{2}$	750	2250	3250	6000	7500
$B = 3000^{5}$	300	900	1500	2400	3000
$B = 1500^{10}$	150	450	750	1200	1500

O número total de acessos, fazendo-se recuperação dos registros via idregs classificados, deve ser acrescido, ainda, dos acessos gastos para classificação da lista de idregs (tabela 4.3).

Os gráficos das figuras 4.3, 4.4 e 4.5 comparam as alternativas de recuperação via idregs classificados X idregs não classificados, mostrando as curvas contendo o número total de acessos necessários (já com os acessos gastos para classificação), respectivamente, considerando-se os fatores de bloco 2, 5 e 10. Cada figura inclui 4 curvas, representando as seguintes situações:

(1) - Recuperação por idregs não classificados (pior caso);
(2) - Recuperação por idregs não classificados (50% de aproveitamento de cada bloco trazido);
(3) - Recuperação por idregs classificados (pior caso);
(4) - Recuperação por idregs classificados (melhor caso).

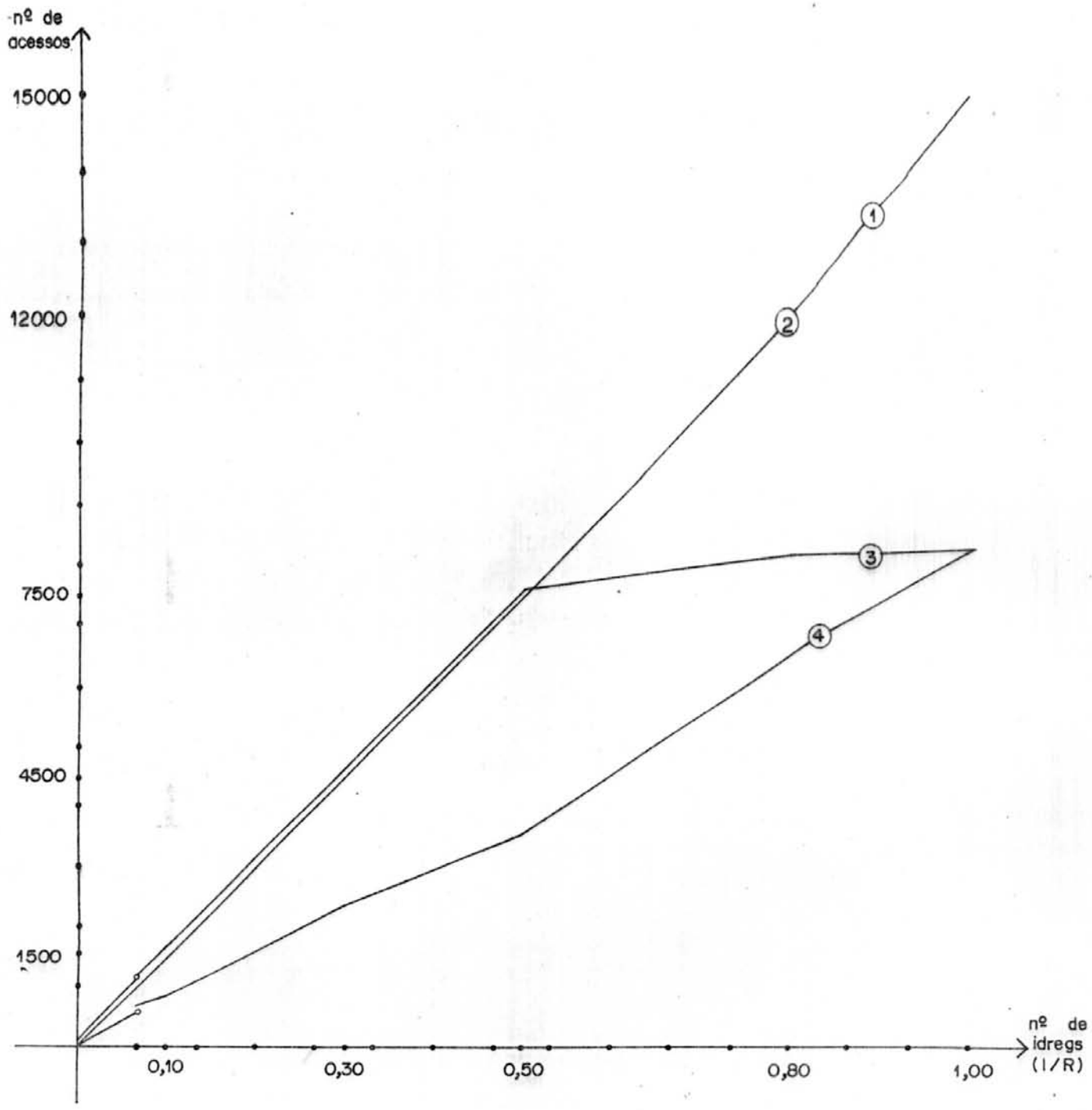


FIGURA 4.3: Número total de acessos para recuperação por idregs classificados X não classificados - $f_b = 2$

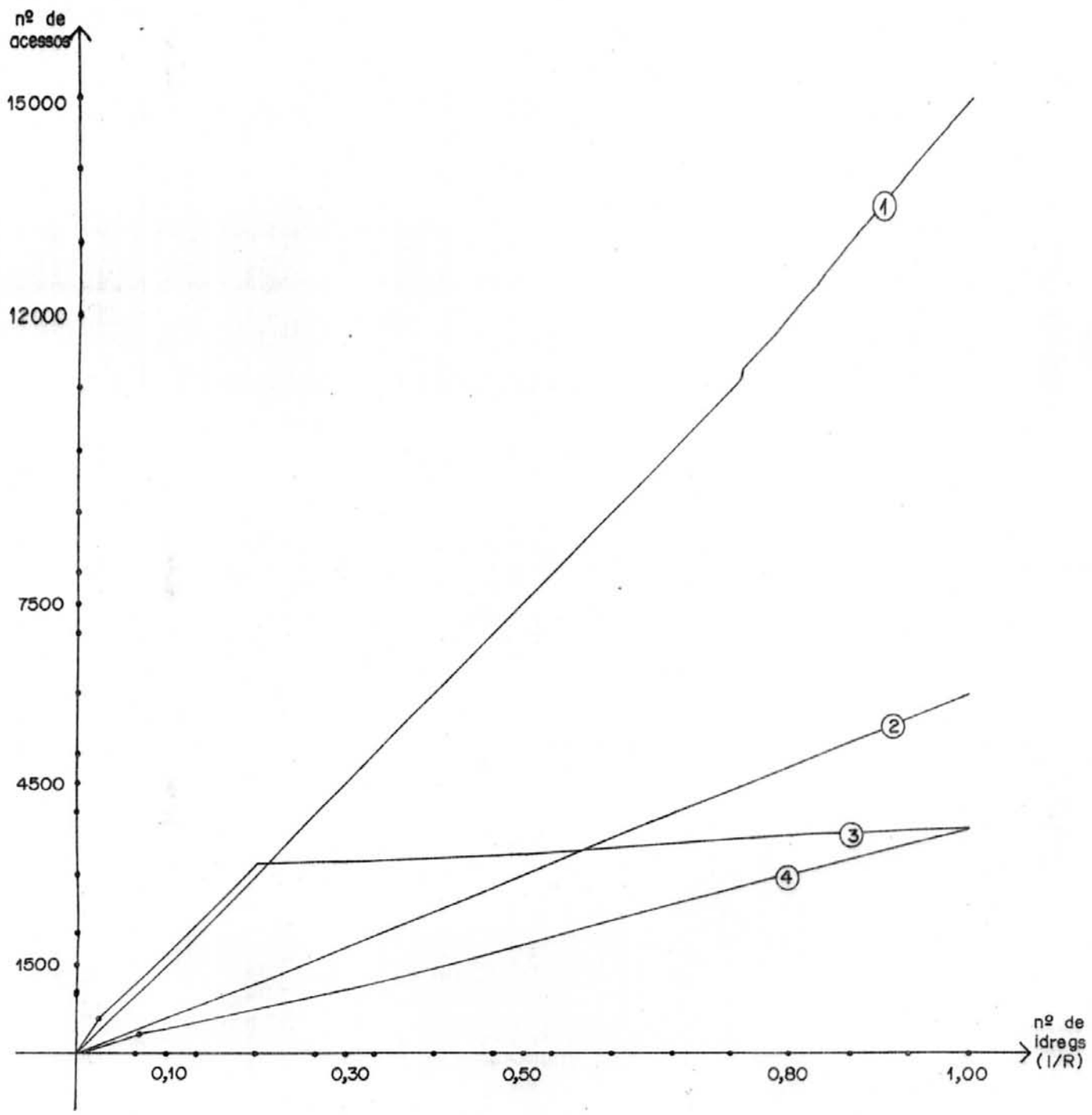


FIGURA 4.4: Número total de acessos para recuperação por idregs classificados X não classificados - $f_b = 5$

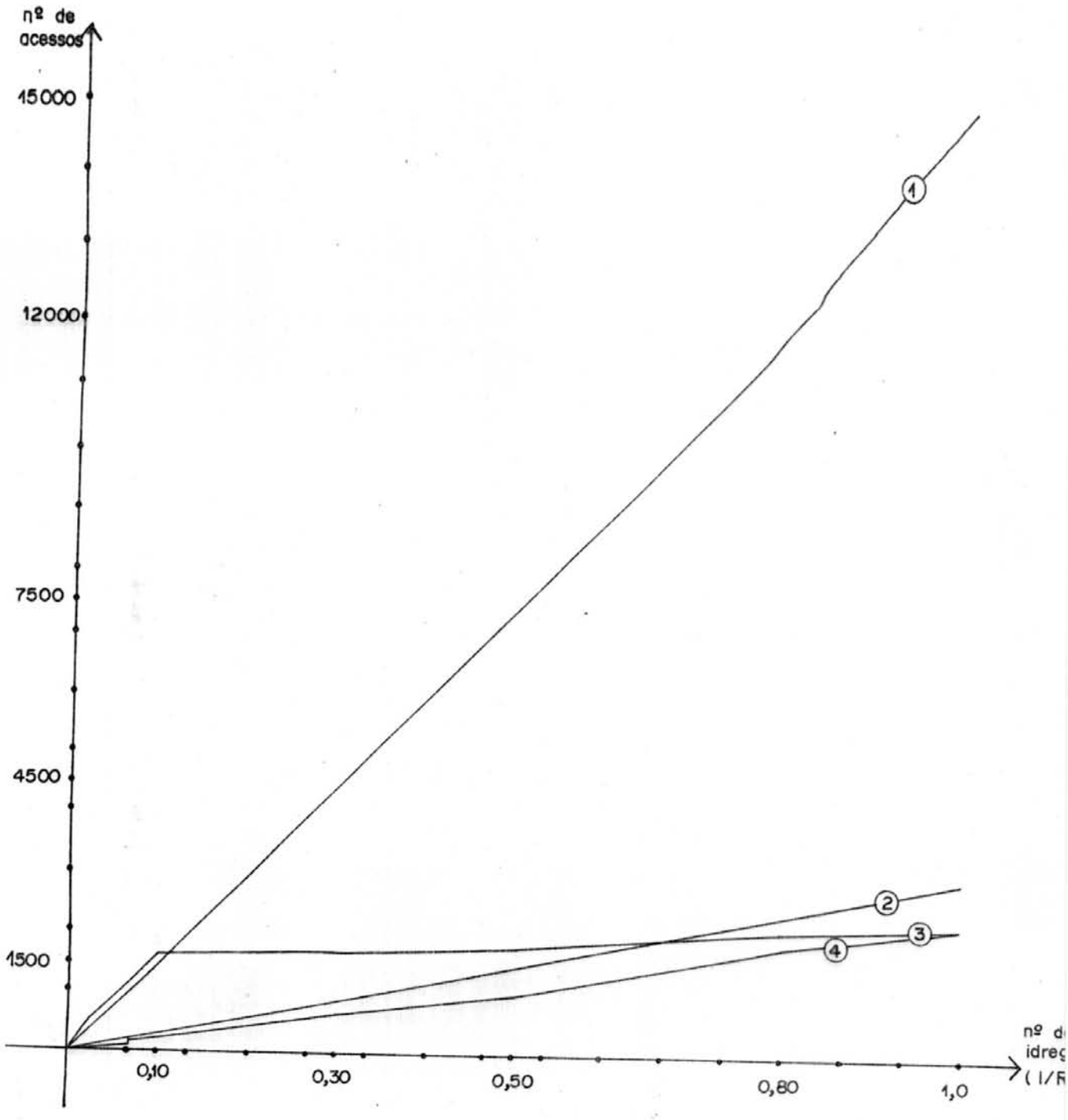


FIGURA 4.5: Número total de acessos para recuperação por idregs classificados X não classificados - f_b = 10

4.3 Classificação de listas de pares de idregs

Algumas das operações estudadas podem produzir listas de pares de idregs, a partir dos quais é feita a recuperação de pares de registros que serão combinados gerando um único registro resultante.

Assim como as listas de idregs, as listas de pares de idregs também podem ser classificadas, com a finalidade de diminuir-se o número total de acessos para recuperação dos registros de dados.

A classificação proposta, neste caso, é feita segundo o primeiro idreg de cada par. A alternativa de classificar, posteriormente, os pares "registro de dados - idreg" provenientes da primeira recuperação por idregs classificados foi omitida devido ao considerável aumento no tamanho destes novos registros.

Considerando-se a área de 4000 bytes para classificação interna, listas de até 5000 pares de idregs podem ser classificadas internamente, já que cada par de idregs ocupará o equivalente a 8 bytes.

A tabela 4.6 mostra o número de acessos para classificação de uma lista de pares de idregs segundo o primeiro idreg do par, conforme sua cardinalidade.

O número de acessos para recuperação dos registros via seus idregs (classificados ou não) é o mesmo visto nas tabelas 4.1 a 4.5, já que está em questão apenas o primeiro idreg de cada par.

A figura 4.6 mostra o gráfico obtido com os dados da tabela 4.6. PI representa o número de pares de idregs da lista envolvida. R é total de registros do arquivo de dados cujos idregs são os primeiros de cada par.

As figuras 4.7 a 4.9 mostram, para arquivos com

fatores de bloco 2, 5 e 10, respectivamente, as curvas comparativas das seguintes situações:

(1) - Recuperação por pares de idregs não classificados (pior caso);

(2) - Recuperação por pares de idregs não classificados (50% de aproveitamento de cada bloco trazido);

(3) - Recuperação por pares de idregs classificados (pior caso);

(4) - Recuperação por pares de idregs classificados (melhor caso).

O número de acessos estimado para cada caso diz respeito, sempre, à recuperação pelo primeiro idreg de cada par, já que não é abordada a classificação de ambos os idregs do par.

TABELA 4.6: Número de acessos para classificação conforme a cardinalidade da lista de pares de idregs

TAMANHO (em pares de idregs)	M (nº de páginas)	ACESSOS
< 500	-	70
500	8	102
1000	16	134
1500	24	166
2000	32	198
3000	47	258
4000	63	322
4500	71	354
5000	79	386
6000	94	634
7000	110	730
7500	118	778
8000	125	820
9000	141	916
10000	157	1012
11000	172	1102
12000	188	1198
13000	204	1294
14000	219	1384
15000	235	1480

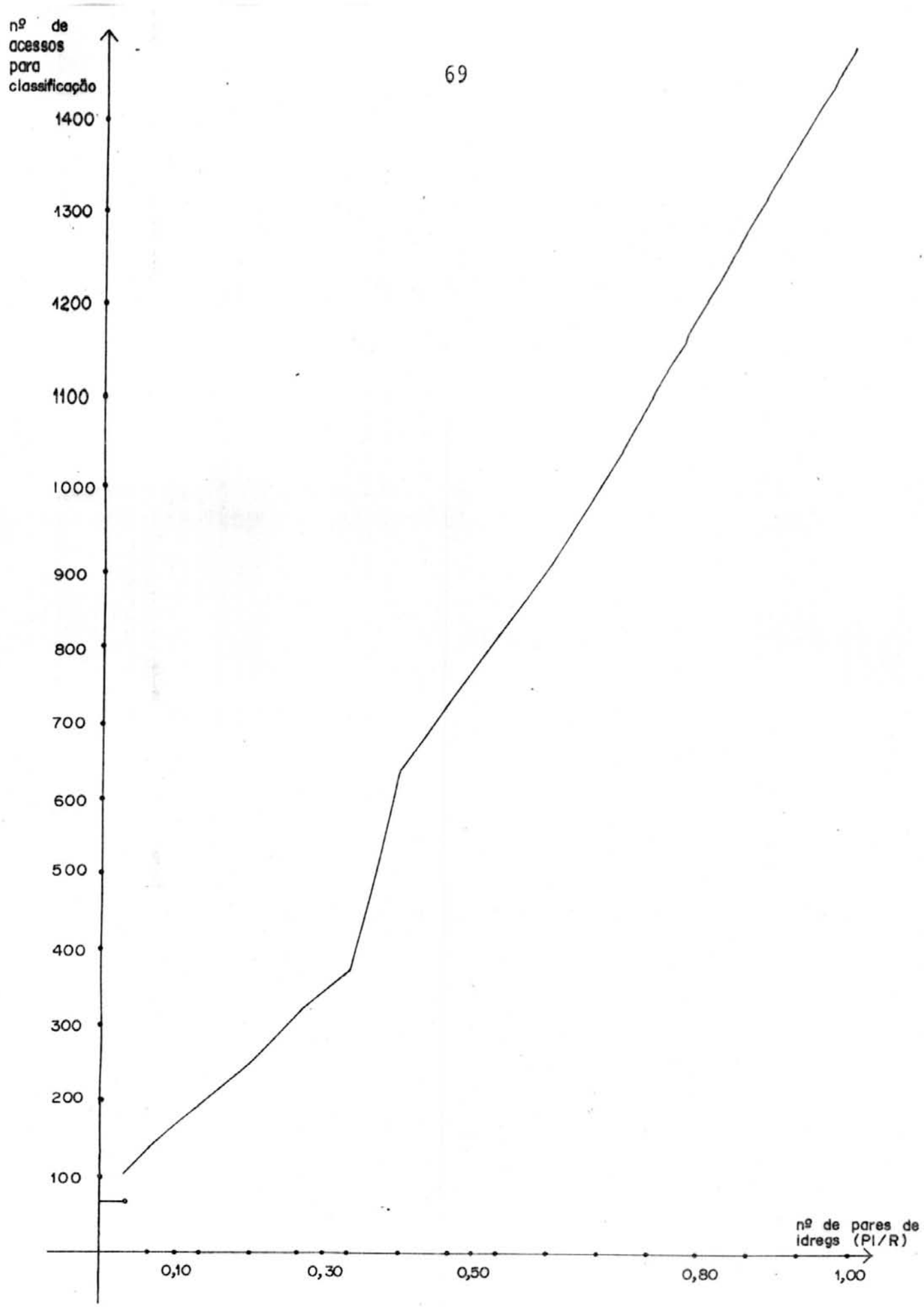


FIGURA 4.6: Número de acessos para classificação de listas de pares de idregs

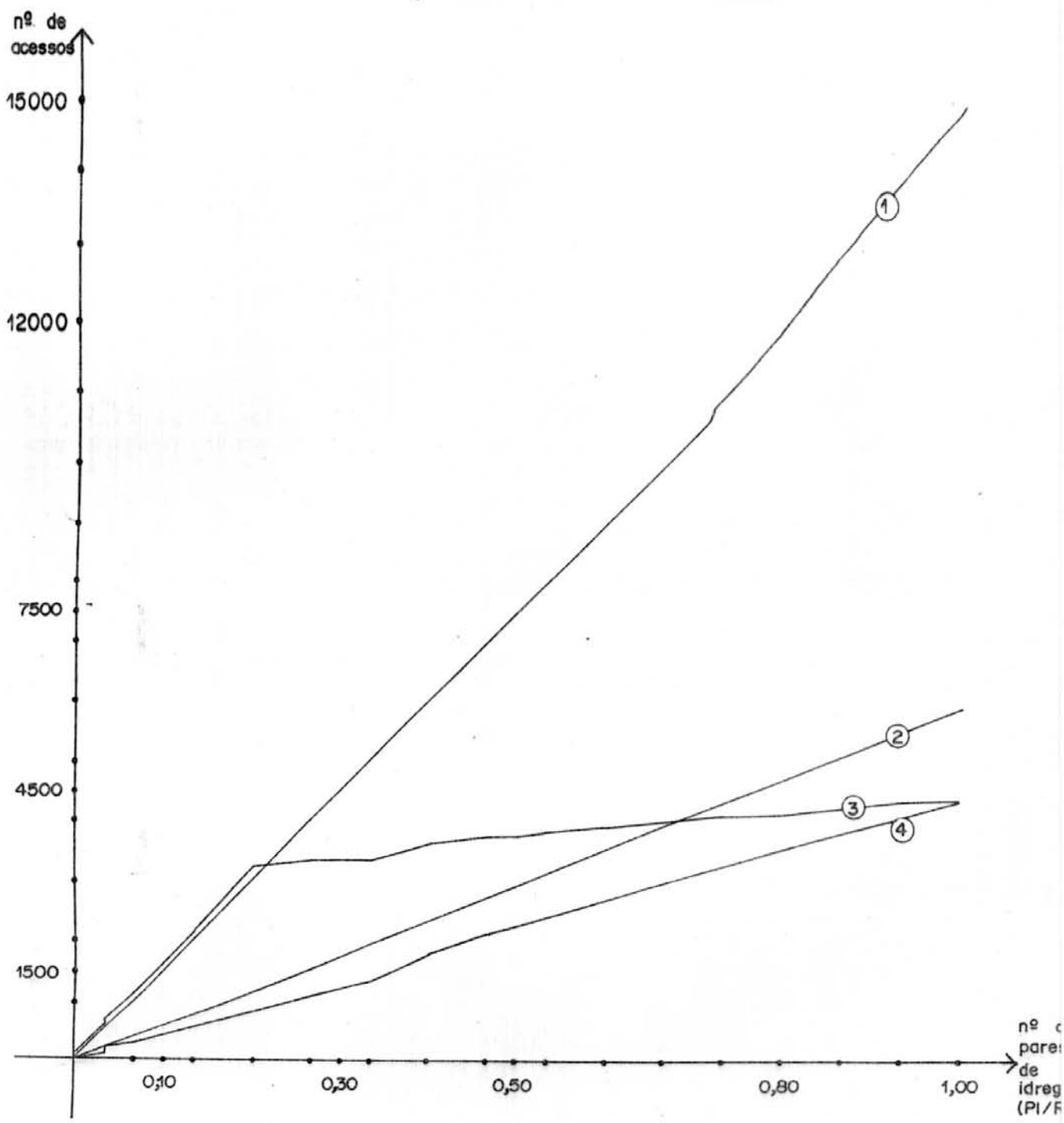


FIGURA 4.8: Número total de acessos para recuperação via pares de idregs classificados X não classificados - $f_b = 5$

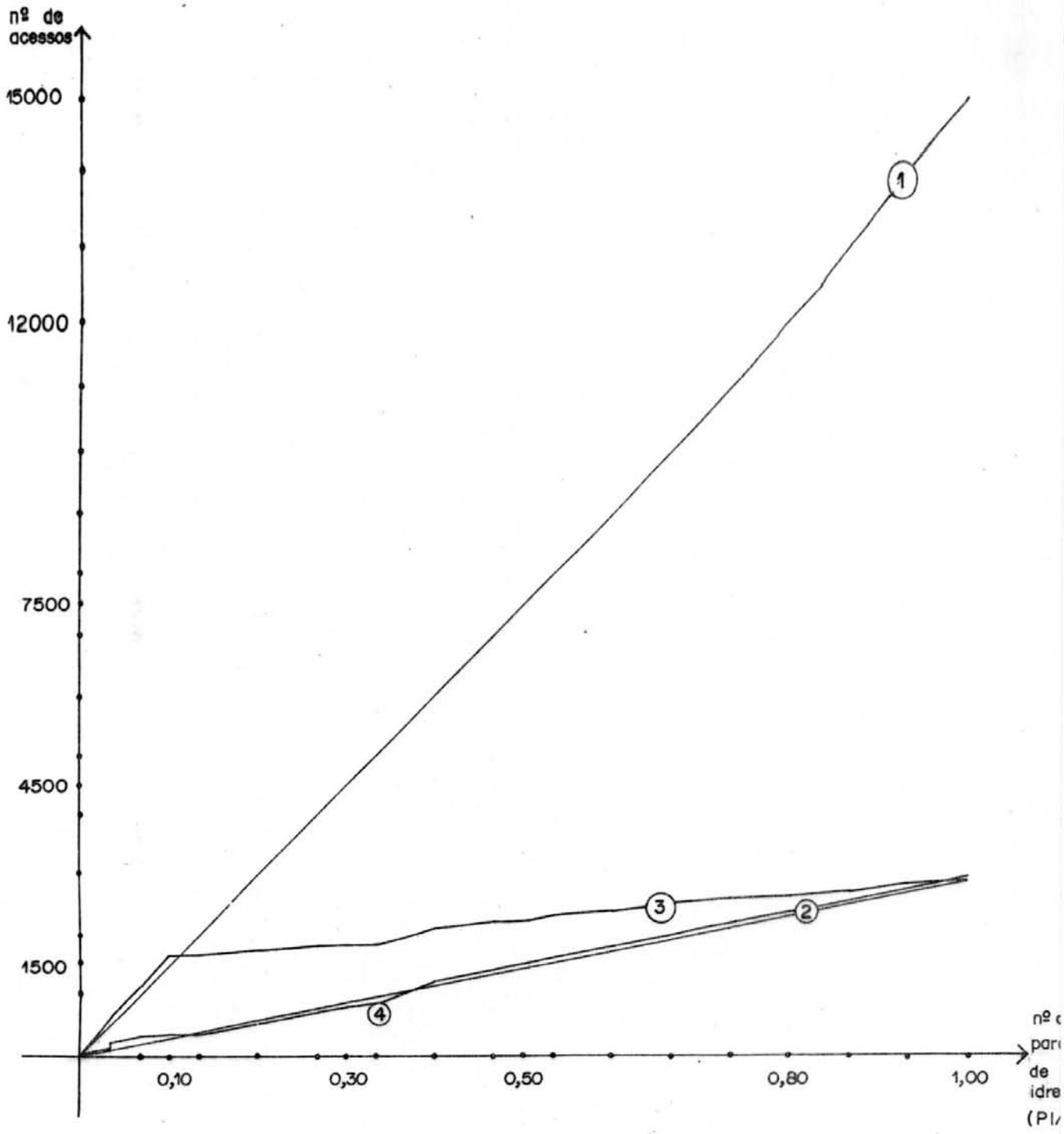


FIGURA 4.9: Número total de acessos para recuperação via pares de idregs classificados X não classificados - f_b = 10

4.4 Conclusão

Através das figuras 4.3, 4.4 e 4.5 é possível observar-se que, quanto maior o fator de bloco, mais cedo a curva (3) intercepta a curva (1). Logo, quanto maior o fator de bloco, menor é o número mínimo de idregs necessários para que seja interessante a classificação de uma lista de idregss, em relação ao pior caso da recuperação por idregs não classificados X pior caso da recuperação por idregs classificados.

Considerando as condições de implementação de registros de dados no computador LABO 8034, com blocos de 512 bytes, vê-se que o fator de bloco 5 parece ser bastante provável de ocorrer. Assim, conforme as curvas da figura 4.4, é possível observar que mesmo o pior caso da classificação, com aproximadamente pouco mais de 20% de idregs, já apresenta número de acessos inferior ao pior caso da recuperação via idregs não classificados.

Considerando-se o aproveitamento de 50% de cada bloco trazido (linhas (2)), é sensível que esse comportamento, na prática, não ocorrerá na forma de uma reta. Quando o número de registros da lista de idregs/pares de idregs é pequeno, a probabilidade de ocorrer um aproveitamento de 50% de cada bloco trazido parece ser menor, considerando-se uma distribuição uniforme desses idregs, o que sugere que as linhas (2) possam ser melhor expressas, talvez, por uma curva como a esboçada na figura 4.10.

Mesmo assim, ainda na figura 4.4, comparando-se as curvas (2) e (4), observa-se que a recuperação via idregs classificados nunca exige número de acessos superior ao exigido pela recuperação via idregs não classificados, com 50% de aproveitamento de cada bloco trazido.

Levando-se em conta a possibilidade de classificação interna, a classificação de listas com até 1000 idregs

ou 500 pares de idregs requer um número de acessos muito pequeno, correspondente, apenas, à sobreposição do programa de classificação.

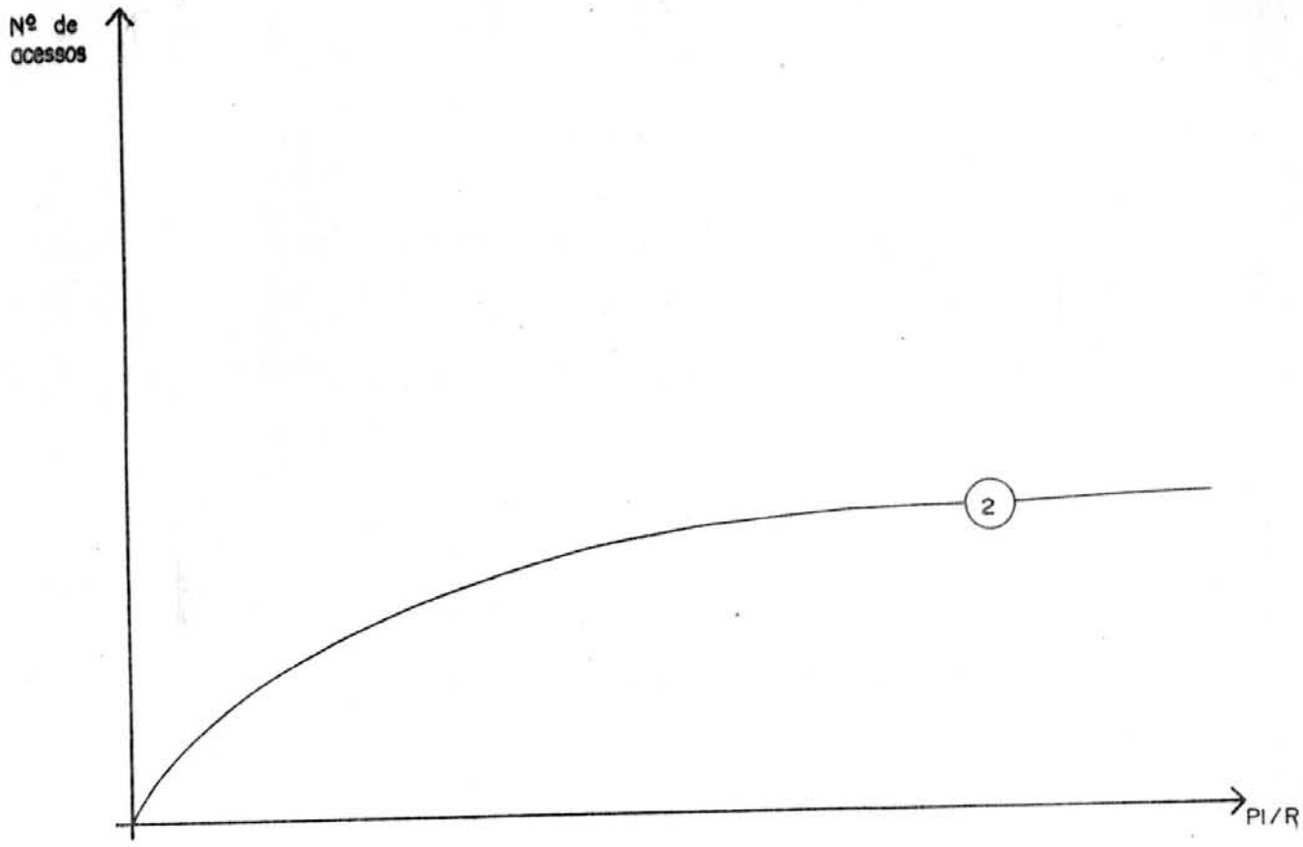


FIGURA 4.10: Esboço de uma linha (2) corrigida

No estudo das figuras 4.7, 4.8 e 4.9, deve-se atentar para o fato de que o pior caso da recuperação via pares de idregs classificados é muito difícil de ocorrer, na prática, já que as listas de pares de idregs são provenientes de uma das seguintes situações:

a) Varredura seqüencial de dois arquivos de inversões;
b) Varredura seqüencial de um arquivo de associações.

No caso (a), quando encontrados dois registros de inversão que satisfazem determinado critério, cada idreg mantido no primeiro registro é concatenado a todos os idregs mantidos no segundo registro.

No caso (b), a cada registro de associação selecionado, o idreg de seu ligante é concatenado aos idregs de todos os seus ligados.

Em ambos os casos, pode-se observar que serão gerados vários pares com o primeiro componente repetido, o que representa uma maior concentração, e um menor número de recuperações exigidas, no pior caso da recuperação via pares de idregs classificados.

Desta forma, decidiu-se optar pela classificação de listas de idregs/pares de idregs para recuperação dos respectivos registros de dados, em casos em que a ordem de classificação do resultado de uma certa operação é indiferente à operação seguinte, na árvore de operações.

5. ALGORITMOS PARA RESOLUÇÃO DAS OPERAÇÕES ESTUDADAS

5.1 Introdução

Conforme a descrição contida no capítulo 2, item 2.4.2, o presente estudo fixou-se na geração de código para operações que possuem tabelas inteiras, relacionais ou ligacionais, como operandos.

Na resolução de tais operações de CI3, são indispensáveis certos cuidados, visando diminuir o número de operações de entrada e saída (que consomem a maior parte do tempo de execução) e o espaço ocupado para armazenamento.

Desta forma, é necessário que o SGBD possua, na fase de geração de código L, características permitindo tirar proveito do contexto apresentado na árvore de operações.

No estudo desenvolvido, foram incluídas, com este objetivo, as seguintes características:

- linhas gerais dos algoritmos desenvolvidos, sendo apresentados vários algoritmos para resolução de cada uma das operações, sempre voltados a situações específicas das quais se pode tirar proveito;

- seleção do algoritmo mais adequado a cada operação, analisando-se as condições encontradas nos respectivos nodos da árvore de operações;

- estudo das ordens de classificação envolvidas, considerando-se que alguns dos algoritmos podem fornecer seu resultado sob diferentes ordens de classificação.

5.2 Características gerais

5.2.1 Aproveitamento do contexto na árvore de operações

A idéia básica, entre os algoritmos desenvolvidos, é a adequação ao contexto na árvore de operações, aproveitando ordens de classificação e estruturas de dados disponíveis sobre os operandos envolvidos.

Os exemplos a seguir mostram duas seqüências de operações LOBAN.

Exemplo 1 -

```
JUNTAR    C   PROPRIETARIOS.TR
COM       C   IMOVEIS.TR
POR       C   PCOD =  C  IPROPRIET  EXCL
```

Exemplo 2 -

```
JUNTAR
   ESTREITAR   C   PROPRIETARIOS.TR
   PARA        PCOD, PNOME
COM   C   IMOVEIS.TR
POR   C   PCOD = C  IPROPRIET  EXCL
```

Analisando-se o exemplo 1, vemos que o operador JUNTAR manipula duas folhas da árvore de operações (já que não manipula resultados de outras operações). Deste modo, seus operandos são arquivos de dados mantidos na própria base de dados, aos quais poderão estar associadas outras estruturas de dados, como arquivos de inversões ou arquivos de associações por item (no caso de existir uma conexão automática definida entre esses dois arquivos de dados).

Assim, algoritmos apropriados para resolução do exemplo 1 podem ser:

a) um algoritmo tentando usar arquivos de inver-

sões sobre os atributos de junção;

b) um algoritmo que faça uso de um arquivo de associações por item definido sobre uma chave da qual participe o atributo de junção pertencente ao primeiro operando;

c) um algoritmo que classifique os dois arquivos de dados envolvidos, para então realizar a junção.

Pode-se observar que, nos itens (a) e (b), a recuperação dos registros candidatos à junção é feita via outra estrutura já disponível, enquanto que no item (c) é necessária a classificação dos dois arquivos de dados. Este fato, sempre que possível, deve ser evitado, devido ao alto número de operações de entrada e saída que envolve.

Considerando ainda o exemplo 1, caso exista um arquivo de associações por item cuja chave primária seja formada pelo atributo de junção pertencente ao primeiro operando, esta operação JUNTAR pode ser transformada em um DESAGRUPAR aplicado sobre o respectivo arquivo de associações.

Quando há disponibilidade de mais de um arquivo de inversões apto a ser utilizado, a escolha do arquivo de inversões mais adequado depende da operação a executar. Este estudo deve ser feito numa etapa anterior à escolha do algoritmo a ser utilizado, associado a cada nodo da árvore de operações.

A operação JUNTAR no exemplo 2, por sua vez, já utiliza um resultado intermediário (da operação ESTREITAR) como operando.

Neste caso, o primeiro operando da operação JUNTAR será uma lista de dados na zona intermediária, podendo estar ou não classificado segundo o atributo de junção, enquanto que o segundo operando será um arquivo de dados (po-

dendo possuir arquivos de inversões envolvendo o atributo de junção). Uma forma de resolução dessa operação JUNTAR poderá ser com o uso de um arquivo de inversões, onde sejam procuradas chaves de inversão com os diferentes valores do atributo de junção ocorrentes na lista de dados. A melhor situação, nessa forma de resolução, ocorre quando a lista de dados se encontra classificada segundo a mesma ordem que o arquivo de inversões disponível.

Uma última opção para o exemplo 2 poderia ser a resolução percorrendo, paralelamente, a lista e o arquivo de dados, classificando-os de antemão, se necessário (a lista de dados já poderá estar classificada, conforme o algoritmo usado para resolução da operação ESTREITAR).

Tendo em vista a ordem de classificação de uma lista de dados qualquer, é interessante lembrar que, se a lista se encontra classificada segundo a ordem c, então está também classificada segundo qualquer ordem p, representada pelos "primeiros" componentes da ordem c. Por exemplo, supondo a existência de uma lista de dados cujos registros têm mesma configuração que as tuplas da tabela relacional componente do arquivo IMOVEIS (figura 2.2), e estão classificados segundo os atributos ICOD-IENDER-ICID, nesta ordem, é possível afirmar que esta lista de dados também se encontra classificada sob as ordens ICOD ou ICOD-IENDER. Essa característica deve ser sempre levada em conta, na aplicabilidade dos algoritmos propostos.

5.2.2 Uso da primitiva de classificação

Apesar de ter sido desenvolvida com características que a tornam bastante eficiente, a primitiva de classificação /CAR 81/ foi utilizada com certo resguardo, nos algoritmos apresentados.

Apenas quando não estão disponíveis outras estru-

turas de acesso, na base de dados interna, permitindo uma recuperação adequada, é feita a classificação de arquivos/ listas de dados.

O alto custo de uma classificação, em número de operações de entrada e saída que representa, já é mencionado em /SMI 75/, que também evita esta operação.

Buscando uma melhor apresentação dos passos realizados, foi incluída no texto dos algoritmos a operação de classificação, sempre que necessário. Considerando-se, porém, o número de páginas ocupadas pela rotina que implementa a primitiva de classificação, é interessante que as chamadas desta primitiva sejam feitas fora do código gerado para os algoritmos, já que alguns deles podem vir a receber arquivos/listas de dados já classificados, não necessitando, então, realizar a operação de classificação.

5.3 Algoritmos desenvolvidos

5.3.1 Apresentação

Os algoritmos para resolução das operações estudadas estão agrupados por operadores de CI3, onde cada grupo inclui:

- uma descrição sucinta dos métodos utilizados nos algoritmos desenvolvidos para resolução desta operação de CI3, envolvendo o tipo de estruturas de dados utilizadas, e a forma de acesso a essas estruturas;
- o conjunto de algoritmos para resolução desta operação.

São ainda incluídos os algoritmos para resolução das operações sobre tabela de ligados, permitidas na expressão seletora associada ao operador COLA.

A fim de simplificar-se o tratamento de tais operações, considera-se que, após a geração do CI3, as mesmas estarão indicadas, respectivamente, pelos operadores:

- COLECT : realiza uma operação de seleção (COLEC) sobre uma tabela de ligados.

- ESTREITT: realiza uma operação de projeção (ESTREITAR) sobre uma tabela de ligados.

- AGRUPART: realiza uma operação de agrupamento (AGRUPAR) sobre uma tabela de ligados.

A metodologia utilizada na descrição dos algoritmos consta sempre de três etapas, denominadas, respectivamente, condição, resultado e método.

Na etapa condição é descrito o conjunto de condições que devem existir para que um determinado algoritmo possa ser aplicado.

Em resultado é descrito o resultado produzido por este algoritmo, incluindo sua ordem de classificação. Nos algoritmos que produzem como resultado tabelas ligacionais na zona intermediária, como lista de ligantes + lista de ligados, são descritas a ordem de classificação entre os registros da lista de ligantes, e a ordem de classificação entre os ligados de um grupo de ligados pertencentes a uma mesma ligação.

Em método é descrito o algoritmo propriamente dito, fazendo uso de duas sub-etapas: estruturas de dados utilizadas e passos, a primeira enumerando as estruturas de dados da base de dados interna utilizadas por este algoritmo e as estruturas auxiliares criadas pelo mesmo na zona intermediária, e a segunda, listando os passos do algoritmo.

Nos algoritmos que não utilizam estruturas de dados adicionais, é omitida em método a sub-etapa estruturas de dados utilizadas.

5.3.2 Descrição em detalhe

São mostrados a seguir os algoritmos desenvolvidos para as operações de CI3, tendo em vista as primitivas de banco de dados disponíveis no sistema L (ver capítulo 3, item 3.5).

1 - COLR, COLRU

Algoritmos apresentados:

- COLR1: percorre seqüencialmente um arquivo/lista de dados (se o operando estiver classificado, resolve a operação COLRU com pesquisa binária).

- COLR2: parte de um arquivo de inversões buscando os registros de dados candidatos à seleção e testando as demais condições.

- Observações:

I) Alguns dos algoritmos propostos (no caso, o algoritmo COLR2) apresentam, na etapa condição, referência a subexpressões de primeiro nível, incluídas na expressão de seleção. Seja o seguinte endereço de ponto:

```
IMOVEIS.
TR.
(C  ICOD =  0050  ET
(C  IPROPRIET =  'P12'  OU
 C  ICID = 'CANOAS'
)
)
```

São consideradas subexpressões de primeiro nível aquelas subexpressões ligadas por ET às demais, constituindo condição necessária para que a expressão de seleção seja satisfeita. No endereço de ponto mostrado, o trecho C ICOD = 0050 e o trecho C IPROPRIET = 'P12' OU C ICID = 'CANOAS' constituem subexpressões de primeiro nível.

II) Ainda no algoritmo COLR2, aparece referência à subexpressão C A $\begin{matrix}>\\=\\<\end{matrix}$ valor, componente da expressão de seleção. Numa subexpressão desse tipo, valor pode representar:

- um item componente de um registro de dados mantido na base de dados ou no canal auxi-

liar (por exemplo, um item componente de um registro de dados representando uma FICHA) ou

- um valor constante qualquer.

III) A resolução de uma operação COLRU é feita através dos próprios algoritmos de resolução da operação COLR, com o uso de um parâmetro adicional (U), que determina se deve ser colecionado apenas um registro de dados (U=1) ou vários (U≠1). Esta medida visa eliminar algoritmos adicionais, cuja descrição é muito semelhante àquelas já apresentadas para a operação COLR. Na geração de código, porém, deve ser feita distinção entre esses dois operadores, com o objetivo de não gerar código desnecessário, quando ocorre uma operação COLRU.

COLR1 (ARQ,U)

- Condição:

ARQ é um arquivo/lista de dados.

- Resultado:

Se U=1 resulta um ou nenhum registro de dados no topo da pilha.

Do contrário, resulta uma lista de dados classificada segundo a mesma ordem que ARQ.

- Método:

Passos:

1 - Se U = 1 então:
1.1 - Se ARQ está classificado segundo sua chave primária, então:
1.1.1 - Buscar em ARQ um registro que satisfaça às condições dadas na expressão seletora (pesquisa binária).
1.1.2 - Se encontrado então:
1.1.2.1 - Colocar este registro no topo da pilha.
1.1.2.2 - Finalizar.

2 - Para cada registro de ARQ, fazer:
2.1 - Se este registro de ARQ satisfaz a expressão seletora, então:
2.1.1 - Se U = 1 então:
2.1.1.1 - Colocar este registro no topo da pilha.
2.1.1.2 - Finalizar.
2.1.2 - Senão:
2.1.2.1 - Adicionar este registro à lista resultado.
2.1.2.2 - Seguir.

COLR2 (ARQ,ORD,U)

- Condição:

ARQ é um arquivo de dados. A expressão de seleção pode ser composta de subexpressões. Pelo menos uma das subexpressões envolve um atributo sobre o qual existe definido um arquivo de inversões, e é da forma C A $\begin{matrix}>\\=\\<\end{matrix}$ valor (onde A representa uma chave de inversão sobre ARQ). As subexpressões devem estar todas ligadas por ET (em primeiro nível).

- Resultado:

Se U=1 então resulta nenhum ou um único registro de dados no topo da pilha. Do contrário:

- Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo o arquivo de inversões sobre ARQ.

- Se ORD não for uma ordem válida, resulta uma lista de dados não classificada.

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ.
(Se U=1 então INV é o arquivo de inversões sobre a chave primária de ARQ).
LI - Lista de idregs (de registros de ARQ).
LC - Lista LI classificada.

Passos:

1 - Subdividir a expressão C A $\begin{matrix}<\\=\\>\end{matrix}$ valor nos trechos:
t_1: C A = valor
t_2: C A > valor (ou C A < valor)

2 - Buscar o registro de INV que satisfaz a subex-

pressão C A = valor (da subexpressão C A $\begin{matrix}>\\=\\<\end{matrix}$ valor).

3 - Se encontrado, fazer:

3.1 - Se o trecho t_1 não é vazio, então:

3.1.1 - Se U=1 então:

3.1.1.1 - Buscar o registro de ARQ apontado pelo idreg mantido nesse registro de INV.

3.1.1.2 - Se este registro de ARQ satisfaz a toda a expressão seletora, então:

3.1.1.2.1 - Colocar este registro no topo da pilha.

3.1.1.2.2 - Finalizar.

3.1.2 - Do contrário: /* U ≠1 */

3.1.2.1 - Para cada idreg mantido nesse registro de inversão, fazer:

3.1.2.1.1 - Se o trecho t_2 é vazio então:
- Buscar o registro de ARQ apontado por este idreg.
- Adicionar o mesmo à lista resultado.

3.1.2.1.2 - Do contrário:
- Adicionar este idreg à lista LI.

/* Se não foi encontrado um registro de inversão nessas condições, supõe-se que a busca pára no mais próximo, ou seja, no primeiro registro de inversão cuja chave primária seja superior a valor */

4 - Se o trecho t_2 não é vazio, então:

4.1 - Para cada registro de INV a partir do registro corrente (em ordem ascendente ou descendente, conforme o trecho t_2), fazer:

4.1.1 - Para cada idreg mantido nesse registro de INV, fazer:

4.1.1.1 - Se ORD for uma ordem válida então:

4.1.1.1.1 - Buscar o registro de ARQ apontado por este idreg.

4.1.1.1.2 - Adicionar o mesmo à lista resultado.

4.1.1.2 - Do contrário:

4.1.1.2.1 - Adicionar este idreg à lista LI.

5 - Se ORD não for uma ordem válida e o trecho t_2 não for vazio, fazer:

5.1 - Classificar a lista LI em ordem crescente (gerando LC).

5.2 - Para cada registro de LC, fazer:

5.2.1 - Buscar o registro de ARQ apontado pelo idreg mantido nesse registro de LC.

5.2.2 - Adicionar o mesmo à lista resultado.

2 - COLA, COLAU

Algoritmos apresentados:

- COLA1: percorre seqüencialmente um arquivo de associações por item ou por idreg.

- COLA2: busca um registro de associação com determinado valor de chave primária e, a partir desse ponto, percorre seqüencialmente o arquivo de associações por item ou por idreg.

- COLA3: busca um registro de inversão sobre os ligantes que apresente um determinado valor como chave primária e, a partir dos idregs aí mantidos, busca registros de associação candidatos à seleção.

- COLA4: percorre seqüencialmente o arquivo de dados de ligantes e, a partir dos idregs dos registros ligantes de associações candidatas, busca os registros de associação para verificar se satisfazem às demais condições de seleção.

- COLA5: percorre seqüencialmente uma lista de ligantes e uma lista de ligados que representam uma tabela ligacional mantida na zona intermediária ou no canal auxiliar.

- Observações:

I) Nos algoritmos COLA1 a COLA4, que manipulam arquivos de associações mantidos na base de dados, é suposta a execução de um passo preliminar, que subdivide a expressão seletora em trechos que podem ser analisados separadamente, envolvendo estruturas de dados distintas (ou grupos distintos de estruturas de dados). Esse passo é detalhado como:

Ø - Percorrer a expressão seletora e subdividi-la nos seguintes trechos:

sel_1 - subexpressões envolvendo apenas atributos constantes no próprio registro de associação.

sel_2 - subexpressões envolvendo atributos do ligante que podem ser encontrados apenas nos respectivos registros de dados.

sel_3 - subexpressões envolvendo atributos dos ligados que podem ser encontrados apenas nos respectivos registros de dados.

sel_4 - subexpressões envolvendo, ao mesmo tempo, atributos dos ligantes + atributos dos ligados, de modo que sua resolução exija acesso aos respectivos registros de dados.

II) A resolução de uma operação COLAU é feita através dos próprios algoritmos de resolução de uma operação COLA (casos: COLA2, COLA3 e COLA5), com o uso de um parâmetro adicional (U), que determina se deve ser obtido (colecionado) apenas um registro de associação (U=1) ou vários (U≠1), pelos mesmos motivos já expostos junto à apresentação dos algoritmos que descrevem as operações COLR e COLRU.

III) Valem as mesmas considerações sobre subexpressões de primeiro nível e valor feitas para as operações COLR e COLRU.

COLA1 (ASS)

- Condição:

ASS é um arquivo de associações por item ou por idreg.

- Resultado:

Lista de dados de ligantes (LISTA DE LIGANTES), classificada segundo a chave primária do arquivo de associações, e lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante está classificado segundo a chave primária de uma tabela de ligados de ASS.

- Método:

Estruturas de dados utilizadas:

ARQD1 - Arquivo de dados contendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.

Observação: Conforme a expressão de seleção, poderão existir outras estruturas de dados. As operações LOBAN constantes na expressão de seleção associada a um COLA serão resolvidas separadamente, por algoritmos específicos para tanto, junto aos quais estarão descritas tais estruturas adicionais.

Passos:

1 - ULTLIGADO ← (próxima posição a ser preenchida na LISTA DE LIGADOS) - 1.

2 - Para cada registro de ASS, fazer:

2.1 - Se sel_1 não é vazia, então verificar se a mesma é satisfeita por este registro de ASS.

Não → Se sel_1 é condição necessária* para a expressão de seleção ser satisfeita, então passar para o próximo registro de ASS. Senão, seguir.

2.2 - Se sel_2 não é vazia, então:

2.2.1 - Buscar o registro de ARQD1 apontado pelo idreg mantido na construção sob nome LIGANTE desse registro de ASS.

2.2.2 - Verificar se sel_2 é satisfeita por este registro de ARQD1.

Não → Se sel_2 é condição necessária para a expressão de seleção ser satisfeita, então passar para o próximo registro de ASS. Senão, seguir.

2.3 - Se sel_3 não é vazia, então:

2.3.1 - Buscar dados dos ligados desse registro de ASS, e resolver as operações constantes em sel_3, conforme o algoritmo apropriado.

2.3.2 - Verificar se sel_3 é satisfeita por este registro de associação.

Não → Se sel_3 é condição necessária para a expressão de seleção ser satisfeita, então passar para o próximo registro de ASS. Senão, seguir.

2.4 - Se sel_4 não é vazia, então:

2.4.1 - Buscar dados de ligante (caso não tenham ainda sido buscados) e ligados, e resolver as operações constantes em sel_4, conforme o algoritmo apropriado.

2.4.2 - Verificar se sel_4 é satisfeita por este registro de associação.

Não → Se sel_4 é condição necessária para a ex

---

* No caso de esta condição ser falsa, toda a expressão de seleção é falsa (por exemplo, o trecho sel_1 está ligado por ET a todos os demais trechos).

pressão ser satisfeita, então passar para o próximo registro de ASS. Senão, seguir.

2.5 - Se a expressão de seleção foi totalmente satisfeita por este registro de associação, então:

2.5.1 - Para cada registro de inversão mantido na construção sob nome LIGADOS desse registro de associação, fazer:

2.5.1.1 - Buscar o registro de ARQD2 apontado pelo idreg mantido nesse registro de inversão.

2.5.1.2 - Adicioná-lo à LISTA DE LIGADOS.

2.5.2 - Buscar em ARQD1 o registro apontado pelo idreg mantido na construção sob nome LIGANTE desse registro de associação (caso o mesmo ainda não tenha sido trazido).

2.5.3 - Se o conteúdo da construção sob nome CARD-LIGADOS desse registro de associação é zero, então:

2.5.3.1 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro trazido de ARQD1 (ligante)
- indicação de tabela de ligados vazia.

2.5.3.2 - Do contrário fazer:

2.5.3.2.1 - PRIMLIGADO ← ULTLIGADO + 1.

2.5.3.2.2 - ULTLIGADO ← ULTLIGADO + (conteúdo da construção sob nome CARD-LIGADOS desse registro de associação).

2.5.3.2.3 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro trazido de ARQD1(ligante)
- PRIMLIGADO
- ULTLIGADO.

COLA2 (ASS,U)

- Condição:

ASS é um arquivo de associações por item. A expressão seletora envolve trechos todos ligados por ET, em primeiro nível, existindo uma subexpressão do tipo C A $\gtrless$ valor, onde A representa a chave primária do arquivo de associações.

- Resultado:

Se U=1, resulta um ou nenhum registro de associação no topo da pilha.

Se U≠1, resultam uma lista de dados de ligantes (LISTA DE LIGANTES), classificada segundo a chave primária do arquivo de associações, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante está classificado segundo a chave primária de uma tabela de ligados de ASS.

- Método:

Estruturas de dados utilizadas:

ARQD1 - Arquivo de dados contendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.

(Vale a mesma observação do COLA1).

Passos:

1 - ULTLIGADO ← (próxima posição a ser preenchida na LISTA DE LIGADOS) - 1.

2 - Subdividir a subexpressão C A $\gtrless$ valor nos trechos:
t_1: C A = valor
t_2: C A > valor (ou C A < valor)

3 - Buscar o registro de ASS que satisfaz a expressão C A = valor.

4 - Se encontrado, fazer:

4.1 - Se o trecho t_1 não é vazio, então:

4.1.1 - Executar os passos 2.1 até 2.4 do algoritmo COLA1. /* Testar as demais subexpressões */

4.1.2 - Se U=1 então: /* só um registro deve ser selecionado */

4.1.2.2 - Se a expressão de seleção foi totalmente satisfeita então colocar no topo da pilha esse registro de associação.

4.1.3 - Do contrário:

4.1.3.1 - Executar o passo 2.5 do algoritmo COLA1. /* Colecionar esse registro de associação*/

/* Se não foi encontrado um registro de associação nessas condições, supõe-se que a busca pára no mais próximo, ou seja, no primeiro registro de associação cuja chave primária seja superior a valor */

5 - Se o trecho t_2 não é vazio, fazer:

5.1 - Para cada registro de ASS a partir do registro corrente (em ordem ascendente ou descendente, conforme o trecho t_2), fazer:

5.1.1 - Executar os passos 2.1 a 2.5 do algoritmo COLA1.

/* Testar as demais subexpressões e colecionar esse registro de associação, caso satisfaça a toda a expressão de seleção */

COLA3 (ASS,U)

- Condição:

ASS é um arquivo de associações por item ou por idreg. A expressão seletora envolve trechos todos ligados por ET (em primeiro nível), existindo uma subexpressão ou conjunto de subexpressões, na forma C A = valor, definidas sobre uma seqüência de atributos sobre os quais o arquivo de ligantes de ASS possui um arquivo de inversões.

- Resultado:

Se U=1, resulta um ou nenhum registro de associação no topo da pilha.

Se U≠1, resultam uma lista de dados de ligantes (LISTA DE LIGANTES), classificada segundo a chave primária do arquivo de inversões sobre os ligantes de ASS, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante está classificado segundo a chave primária de uma tabela de ligados de ASS.

- Método:

Estruturas de dados utilizadas:

ARQD1 - Arquivo de dados contendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.
INV - Arquivo de inversões sobre ARQD1.

(Vale a mesma observação do COLA1).

Passos:

1 - ULTLIGADO ← (próxima posição a ser preenchida na LISTA DE LIGADOS) - 1.

2 - Buscar em INV o registro de inversão cuja construção sob nome ITENS satisfaça a subexpressão (ou as subexpressões) envolvendo esta chave de

inversão sobre o arquivo de ligantes.

3 - Se encontrado, fazer:

3.1 - Se U≠1, fazer:

3.1.1 - Para cada idreg mantido nesse registro de associação, fazer:

3.1.1.1 - Buscar o registro de ASS cuja construção sob nome LIGANTE tenha como conteúdo este idreg.

3.1.1.2 - Executar os passos 2.1 a 2.5 do algoritmo COLA1. /* Testar as demais subexpressões e, caso satisfeitas, colecionar este registro de associação */

3.2 - Do contrário, fazer:

3.2.1 - Buscar o registro de ASS cuja construção sob nome LIGANTE tenha como conteúdo o idreg mantido neste registro de inversão.

3.2.2 - Executar os passos 2.1 a 2.4 do algoritmo COLA1. /* Testar as demais subexpressões */

3.2.3 - Se a expressão de seleção foi totalmente satisfeita por este registro de associação, colocar o mesmo no topo da pilha.

COLA4 (ASS)

- Condição:

ASS é um arquivo de associações por idreg. A expressão seletora contém subexpressões envolvendo atributos dos ligantes, além de outras subexpressões. As subexpressões de primeiro nível estão todas ligadas por ET; as subexpressões envolvendo atributos dos ligantes podem estar ligadas por outros operadores lógicos entre si, mas estão ligadas por ET às demais subexpressões de primeiro nível.

- Resultado:

Lista de dados de ligantes (LISTA DE LIGANTES), não classificada (já que o arquivo de dados dos ligantes não se encontra classificado), e lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante está classificado segundo a chave primária de uma tabela de ligados de ASS.

- Método:

Estruturas de dados utilizadas:

ARQD1 - Arquivo de dados contendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.

(Vale a mesma observação do COLA1).

Passos:

1 - ULTLIGADO ← (próxima posição a ser ocupada na LISTA DE LIGADOS) - 1.

2 - Para cada registro de ARQD1, fazer:
2.1 - Se este ligante satisfaz o trecho sel_2 da expressão seletora, fazer:
2.1.1 - Buscar o registro de ASS cuja construção sob nome LIGANTE tenha como conteúdo o idreg des-

te registro de ARQD1.

2.1.2 - Executar o passo 2.1 do algoritmo COLA1.

/* Testa a subexpressão sel_1 */

2.1.3 - Executar os passos 2.3 a 2.5 do algoritmo COLA1.

/* Testa as demais subexpressões ainda não testadas e, se satisfeitas, coleciona este registro de associação */

COLA5 (L1, L2, U)

- Condição:

L1, L2 são listas que representam uma tabela ligacional mantida na zona intermediária ou no canal auxiliar (L1 corresponde à lista de ligantes e L2 corresponde à lista de ligados).

- Resultado:

Se U=1 resulta um ou nenhum registro de associação no topo da pilha.

Se U≠1, resultam uma lista de dados de ligantes (LISTA DE LIGANTES), classificada segundo a mesma ordem que L1, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante está classificado segundo a mesma ordem que uma tabela de ligados mantida em L2.

- Método:

Passos:

1 - Percorrer a expressão seletora subdividindo-a nos seguintes trechos:

sel_1 - subexpressões envolvendo apenas dados do ligante e cardinalidade de ligados (são estas as únicas informações possíveis de conseguir nos próprios registros da lista L1).

sel_2 - demais subexpressões.

2 - U ← (próxima posição a ser ocupada na LISTA DE LIGADOS) - 1.

3 - Para cada registro da lista L1, fazer:

3.1 - Se sel_1 é não-vazio, então verificar se este trecho é satisfeito pelo registro corrente de L1.

Não → Se sel_1 é condição necessária para a expressão de seleção ser satisfeita, então passar para o próximo registro de L1. Do contrário, seguir.

3.2 - Se sel_2 é não vazio, então:

3.2.1 - Buscar em L2 os dados dos ligados desse ligante de L1 e resolver as operações constantes em sel_2, conforme os algoritmos apropriados.

3.3 - Se a expressão de seleção foi satisfeita,então:

3.3.1 - Se U=1, então:

3.3.1.1 - Colocar no topo da pilha esta associação.

3.3.2 - Do contrário, fazer:

3.3.2.1 - Se a tabela de ligados dessa associação não é vazia, fazer:

3.3.2.1.1 - P ← U + 1

3.3.2.1.2 - U ← U + (ULTLIGADO - PRIMLIGADO + 1)
/* O parêntese dá o total de ligados dessa ligação */

3.3.2.1.3 - Para cada registro de L2 cuja posição está entre PRIMLIGADO e ULTLIGADO inclusive, fazer:
- Adicionar o mesmo à LISTA DE LIGADOS.

3.3.2.1.4 - Adicionar à LISTA DE LIGANTES um registro contendo os seguintes valores:
- o registro corrente de L1 (o ligante dessa ligação)
- P (posição do primeiro ligado)
- U (posição do último ligado).

3.3.2.2 - Do contrário, fazer:
/* Neste caso a tabela de ligados é vazia */.

3.3.2.2.1 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro corrente de L1 (o ligante dessa ligação)
- indicação de tabela de ligados vazia.

3 - COLL

Algoritmos apresentados:

- COLL1: percorre seqüencialmente um arquivo de associações.

- COLL2: percorre seqüencialmente um arquivo de associações, gerando uma lista de ligados que é classificada para eliminar duplicatas.

- COLL3: busca um registro de associação por item com determinado valor de chave e a partir daí percorre seqüencialmente o arquivo de associações por item.

- COLL4: busca um registro de inversão sobre os ligantes com determinado valor de chave e, se encontrado, busca registros de associação a partir dos idregs mantidos nesse registro de inversão.

- COLL5: idem ao COLL4, porém classifica a lista de ligados gerada, para eliminar duplicatas.

- COLL6: percorre seqüencialmente o arquivo de dados dos ligantes, buscando registros de associação a partir dos idregs dos ligantes que satisfazem um certo trecho da expressão de seleção.

- COLL7: percorre seqüencialmente uma lista de ligantes e uma lista de ligados que representam uma tabela ligacional mantida na zona intermediária ou no canal auxiliar.

- Observações:

I) Nos algoritmos COLL1 a COLL6, que manipulam arquivos de associações mantidos na base de dados, é suposta a execução de um passo preliminar, que subdivide a primeira expressão seletora (aquela que seleciona registros

de associação) em quatro trechos, e subdivide a segunda expressão seletora (aquela que seleciona ligados, dentro de um registro de associação) em dois trechos. Este passo adicional é dado pelas seguintes etapas:

Ø.1 - Percorrer a primeira expressão seletora e subdividi-la nos seguintes trechos:

- sel_1 - subexpressões envolvendo apenas atributos constantes no próprio registro de associação.
- sel_2 - subexpressões envolvendo atributos do ligante que podem ser encontrados apenas no respectivo registro de dados.
- sel_3 - subexpressões envolvendo atributos dos ligados que podem ser encontrados apenas nos respectivos registros de dados.
- sel_4 - subexpressões envolvendo ao mesmo tempo atributos do ligante + atributos dos ligados, de modo que sua resolução exija acesso aos respectivos registros de dados.

Ø.2 - Percorrer a segunda expressão seletora e subdividi-la nos seguintes trechos:

- sel_5 - subexpressões resolvíveis por acesso à chave primária dos ligados.
- sel_6 - subexpressões não resolvíveis por acesso à chave primária dos ligados.

II) Valem as mesmas considerações sobre subexpressões de primeiro nível e valor feitas para as operações COLR e COLRU.

COLL1 (ASS)

- Condição:

ASS é um arquivo de associações por item.

- Resultado:

Uma lista de dados não classificada.

- Método:

Estruturas dados utilizadas:

ARQD1 - arquivo de dados contendo ligantes.
ARQD2 - arquivo de dados contendo ligados.

Observação: Conforme a primeira expressão de seleção, poderão existir outras estruturas de dados. As operações LOBAN constantes na primeira expressão de seleção associada a um COLL serão resolvidas separadamente, por algoritmos específicos para tanto, junto aos quais estarão descritas tais estruturas adicionais.

Passos:

1 - Para cada registro de ASS, fazer:

1.1 - Se a construção sob nome CARD-LIGADOS desse registro de ASS tem conteúdo diferente de zero, fazer:

1.1.1 - Se sel_1 não é vazia, então verificar se a mesma é satisfeita por este registro de ASS. Não → Se sel_1 é condição necessária* para a expressão de seleção ser satisfeita, então passar para o próximo registro de ASS. Senão, seguir.

---

* No caso de esta condição ser falsa, toda a expressão de seleção é falsa (por exemplo, a expressão está ligada por ET a todos os demais trechos).

1.1.2 - Se sel_2 não é vazia, então:

1.1.2.1 - Buscar o registro de ARQD1 apontado pelo idreg mantido na construção sob nome LIGANTE desse registro de ASS.

1.1.2.2 - Verificar se sel_2 é satisfeita por este registro de ARQD1.

Não → Se sel_2 é condição necessária para a expressão de seleção ser satisfeita, então passar para o próximo registro de ASS. Senão, seguir.

1.1.3 - Se sel_3 não é vazia, então.

1.1.3.1 - Buscar os dados dos ligados desse registro de ASS e resolver as operações constantes em sel_3, conforme o algoritmo apropriado.

1.1.3.2 - Verificar se sel_3 é satisfeita por este registro de ASS.

Não → Se sel_3 é condição necessária para a expressão de seleção ser satisfeita, então passar para o próximo registro de ASS. Senão, seguir.

1.1.4 - Se sel_4 não é vazia, então:

1.1.4.1 - Buscar os dados de ligante (caso não tenha sido ainda acessado) e ligados, e resolver as operações constantes em sel_4, conforme o algoritmo apropriado.

1.1.4.2 - Verificar se sel_4 é satisfeita por este registro de ASS.

Não → Se sel_4 é condição necessária para a expressão de seleção ser satisfeita, então passar para o próximo registro de ASS. Senão, seguir.

1.1.5 - Se a primeira expressão de seleção foi totalmente satisfeita, então:

1.1.5.1 - Para cada registro de inversão mantido na construção sob nome LIGADOS desse registro de associação, fazer:

1.1.5.1.1 - Verificar se a construção sob nome ITENS desse registro de inversão satisfaz o trecho sel_5 da segunda expressão seletora.

Não → Se o trecho sel_5 é condição necessária para que a segunda expressão seletora seja satisfeita, então passar para o próximo registro de inversão.

1.1.5.1.2 - Buscar o registro de ARQD2 cujo idreg é mantido nesse registro de inversão.

1.1.5.1.3 - Verificar se o registro trazido de ARQD2 satisfaz o trecho sel_6 da segunda expressão seletora.

Não → Se o trecho sel_6 é condição necessária para que a segunda expressão seletora seja satisfeita, então passar para o próximo registro de inversão.

1.1.5.1.4 - Adicionar à lista resultado esse registro de ARQD2.

COLL2 (ASS,ORD)

- Condição:

ASS é um arquivo de associações por idreg.

- Resultado:

Se ORD for uma ordem válida resulta uma lista de dados classificada segundo a ordem ORD dos atributos de um ligado de um registro de associação de ASS. Do contrário resulta uma lista de dados classificada segundo a chave primária dos ligados de ASS.

- Método:

Estruturas de dados utilizadas:

LI - Lista de dados (contendo ligados colecionados).
ARQD1 - Arquivo de dados contendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.

(Vale a mesma observação do COLL1).

Passos:

1 - Executar o algoritmo COLL1, substituindo o passo 1.1.5.1.4 por:

1.1.5.1.4 - Se este registro satisfaz a segunda expressão seletora do COLL, então:
- Adicionar o mesmo à lista LI.
/* Cria lista de registros de dados para classificação */

2 - Se ORD for uma ordem válida, então:

2.1 - Classificar a lista LI segundo a ordem ORD (eliminando duplicatas e gerando a lista resultado).

3 - Do contrário:

3.1 - Classificar a lista LI segundo a chave primária dos ligados (eliminando duplicatas e gerando a lista resultado).

COLL3 (ASS)

- Condição:

ASS é um arquivo de associações por item. A primeira expressão seletora envolve trechos todos ligados por ET(em primeiro nível), existindo uma subexpressão do tipo C A $\begin{matrix}>\\=\\<\end{matrix}$ valor, onde A representa a chave primária do arquivo de associações.

- Resultado:

Lista de dados não classificada.

- Método:

Estruturas de dados utilizadas:

ARQD1 - Arquivo de dados contendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.

(Vale a mesma observação do COLL1).

Passos:

1 - Subdividir a subexpressão C A $\begin{matrix}>\\=\\<\end{matrix}$ valor nos trechos:
t_1: C A = valor
t_2: C A > valor (ou C A < valor)

2 - Buscar o registro de ASS que satisfaz a expressão C A = valor.

3 - Se encontrado, fazer:
3.1 - Se o trecho t_1 não é vazio, então:
3.1.1 - Executar o passo 1.1 do algoritmo COLL1.
/* testa as expressões de seleção e coleciona os ligados, se satisfeitas */

/* Se não foi encontrado um registro nessas con-

dições, supõe-se que a busca pára no mais próximo, ou seja, no primeiro registro de associação cuma chave primária seja superior a valor */

4 - Se o trecho t_2 não é vazio, então:

4.1 - Para cada registro de ASS a partir do registro corrente (em ordem ascendente ou descendente, conforme o trecho t_2), fazer:

4.1.1 - Executar o passo 1.1 do algoritmo COLL1.
/* testa as expressões de seleção e coleciona os ligados, se satisfeitas */

COLL4 (ASS)

- Condição:

ASS é um arquivo de associações por item. A primeira expressão seletora envolve trechos todos ligados por ET (em primeiro nível), existindo uma subexpressão ou conjunto de subexpressões, na forma C A = valor, definidas sobre uma seqüência de atributos sobre os quais o arquivo de ligantes de ASS possui um arquivo de inversões.

- Resultado:

Uma lista de dados não classificada.

- Método:

Estruturas de dados utilizadas:

ARQD1 - Arquivo de dados contendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.
INV - Arquivo de inversões sobre ARQD1 (preferencialmente sobre a chave primária dos ligantes).

(Vale a mesma observação do COLL1).

Passos:

1 - Buscar em INV o registro de inversão cuja construção sob nome ITENS satisfaça a subexpressão (ou as subexpressões) envolvendo esta chave de inversão sobre o arquivo de ligantes.

2 - Se encontrado, fazer:
2.1 - Para cada idreg mantido nesse registro de inversão, fazer:
2.1.1 - Buscar o registro de ASS cuja construção sob nome LIGANTE tenha como conteúdo o idreg corrente.

2.1.2 - Executar o passo 1.1 do algoritmo COLL1.
/* testa as expressões de seleção e coleciona os ligados, se satisfeitas */

COLL5 (ASS,ORD)

- Condição:

ASS é um arquivo de associações por idreg. A primeira expressão seletora envolve trechos todos ligados por ET (em primeiro nível), existindo uma subexpressão ou conjunto de subexpressões, na forma C A = valor, definidas sobre uma seqüência de atributos sobre os quais o arquivo de ligantes ASS possui um arquivo de inversões.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD dos atributos de um ligado de um registro de ASS. Do contrário, resulta uma lista de dados classificada segundo a chave primária dos ligados de ASS.

- Método:

Estruturas de dados utilizadas:

ARQD1 - Arquivo de dados contendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.
INV - Arquivo de inversões sobre ARQD1 (preferencialmente sobre a chave primária dos ligantes).
LI - Lista de dados (contendo ligados colecionaods).

(Vale a mesma observação do COLL1).

Passos:

1 - Buscar em INV um registro de inversão cuja construção sob nome ITENS satisfaça a subexpressão(ou as subexpressões) envolvendo esta chave de inversão sobre o arquivo de ligantes.

2 - Se encontrado, fazer:
2.1 - Para cada idreg mantido nesse registro de inversão, fazer:

2.1.1 - Buscar o registro de ASS cuja construção sob nome LIGANTE tenha como conteúdo o idreg corrente.

2.1.2 - Executar o passo 1.1 do algoritmo COLL1, substituindo o item 1.1.5.1.4 por:

/* testa as expressões de seleção */

1.1.5.1.4 - Adicionar à lista LI esse registro de ARQD2.

3 - Se ORD for uma ordem válida, então:

3.1 - Classificar a lista LI segundo a ordem ORD (eliminando as duplicatas e gerando a lista resultado).

4 - Do contrário:

4.1 - Classificar a lista LI segundo a chave primária dos ligados (eliminando duplicatas e gerando a lista resultado).

COLL6 (ASS,ORD)

- Condição:

ASS é um arquivo de associações por idreg. A primeira expressão seletora contém subexpressões envolvendo atributos dos ligantes, além de outras subexpressões. As subexpressões de primeiro nível estão todas ligadas por ET; as subexpressões envolvendo atributos dos ligantes podem estar ligadas por outros operadores lógicos entre si, mas estão ligadas por ET às demais subexpressões de primeiro nível.

- Resultado:

Se ORD for uma ordem válida resulta uma lista de dados classificada segundo a ordem ORD dos atributos de um ligado de um registro de ASS. Do contrário, resulta uma lista de dados classificada segundo a chave primária dos ligados de ASS.

- Método:

Estruturas de dados utilizadas:

ARQD1 - Arquivo de dados contendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.
LI - Lista de dados (contendo ligados colecionados).

(Vale a mesma observação do COLL1).

Passos:

1 - Para cada registro de ARQD1, fazer:
1.1 - Se este registro satisfaz o trecho sel_2 da expressão seletora, fazer:
1.1.1 - Buscar o registro de ASS cuja construção sob nome LIGANTE tenha como conteúdo o idreg do registro corrente de ARQD1.
1.1.2 - Executar o passo 1.1 do algoritmo COLL1, subs

tituindo o item 1.1.5.1.4 por:

/* testa as expressões de seleção */

1.1.5.1.4 - Adicionar à lista LI esse registro de ARQD2.

/* Deve ser eliminado aqui o passo 1.1.2 do algoritmo COLA1, que testa o trecho sel_2 */

2 - Se ORD for uma ordem válida, então:

2.1 - Classificar a lista LI segundo a ordem ORD (eliminando as duplicatas e gerando a lista resultado).

3 - Do contrário:

3.1 - Classificar a lista LI segundo a chave primária dos ligados (eliminando duplicatas e gerando a lista resultado).

COLL7 (L1, L2, ORD)

- Condição:

L1 e L2 são listas de dados que representam uma tabela ligacional mantida na zona intermediária ou no canal auxiliar (L1 corresponde à lista de ligantes e L2 corresponde à lista de ligados).

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD dos atributos de um registro da lista L2. Do contrário, resulta uma lista de dados classificada segundo a chave primária dos ligados de uma ligação mantida em L1, L2.

- Método:

Estruturas de dados utilizadas:

LI - Lista de dados (contendo ligados colecionados).

Passos:

1 - Percorrer a primeira expressão seletora subdividindo-a nos seguintes trechos:
sel_1 - subexpressões envolvendo apenas dados do ligante e cardinalidade de ligados (são estas as únicas informações possíveis de conseguir nos próprios registros da lista L1).
sel_2 - demais subexpressões.

2 - Para cada registro da lista L1, fazer:
2.1 - Se sel_1 não é vazio, então verificar se este trecho é satisfeito pelo registro corrente de L1.
Não → Se sel_1 é condição necessária para a expressão de seleção ser satisfeita, então

passar para o próximo registro de L1. Do contrário, seguir.

2.2 - Se sel_2 não é vazio, então buscar em L2 os dados dos ligados desse ligante de L1 e resolver as operações constantes em sel_2, conforme os algoritmos apropriados.

2.3 - Se a primeira expressão de seleção foi satisfeita, então:

2.3.1 - Se este ligante de L1 não possui uma tabela de ligados vazia, fazer:

2.3.1.1 - Para cada registro de L2 cuja posição esteja entre os idregs de primeiro e último ligados desse ligante de L1 (inclusive), fazer:

2.3.1.1.1 - Adicionar à lista LI esse registro de L2.

3 - Se ORD for uma ordem válida, então:

3.1 - Classificar a lista LI segundo a ordem ORD (eliminando as duplicatas e gerando a lista resultado).

4 - Do contrário:

4.1 - Classificar a lista LI segundo a chave primária dos ligados (eliminando duplicatas e gerando a lista resultado).

4 - ESTR, ESTRND

Algoritmos apresentados:

- ESTREIT1: percorre seqüencialmente um arquivo de inversões cuja chave primária é constituída exatamente pelos atributos mantidos pelo estreitamento.

- ESTREIT2: percorre seqüencialmente um arquivo de inversões onde os atributos mantidos pelo estreitamento são "os primeiros" da chave de inversão, podendo realizar eliminação de duplicatas.

- ESTREIT3: percorre seqüencialmente um arquivo/lista de dados, estreitando sem eliminar duplicatas.

- ESTREIT4: percorre seqüencialmente um arquivo/lista de dados, realizando o estreitamento e eliminando as duplicatas.

- ESTREIT5: classifica um arquivo/lista de dados produzindo como saída o resultado do estreitamento, com eliminação de duplicatas.

ESTREIT1 (ARQ)

- Condição:

ARQ é um arquivo de dados que possui um arquivo de inversões definido exatamente sobre os atributos mantidos pelo estreitamento.

- Resultado:

Lista de dados classificada segundo a chave primária do arquivo de inversões.

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ.

Passos:

1 - Para cada registro de INV, fazer:

1.1 - Adicionar à lista resultado um registro cujo conteúdo é o conteúdo da construção sob nome ITENS desse registro de inversão.

ESTREIT2 (ARQ)

- Condição:

ARQ é um arquivo de dados que possui um arquivo de inversões onde os atributos mantidos pelo estreitamento são "os primeiros" da chave de inversão.

- Resultado:

Lista de dados classificada segundo os atributos mantidos pelo estreitamento (constantes na chave de inversão).

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ.

Passos:

1 - ANTERIOR ← valores dos atributos mantidos pelo estreitamento ocorrentes na construção sob nome ITENS do primeiro registro de INV.

2 - Adicionar à lista resultado um registro contendo os valores mantidos em ANTERIOR.

3 - Para cada registro de INV, fazer:

3.1 - Se o conteúdo da construção sob nome ITENS do registro corrente de INV é diferente do mantido em ANTERIOR, nos campos correspondentes, então:

3.1.1 - ANTERIOR ← valores dos atributos mantidos pelo estreitamento ocorrentes na construção sob nome ITENS desse registro de INB.

3.1.2 - Adicionar à lista resultado um registro contendo os valores mantidos em ANTERIOR.

ESTREIT3 (ARQ)

- Condição:

ARQ é um arquivo/lista de dados. São mantidos pelo estreitamento todos os atributos que constituem a chave primária de ARQ.

- Resultado:

Lista de dados classificada segundo a ordem de ARQ.

- Método:

Passos:

1 - Para cada registro de ARQ, fazer:
1.1 - Formar um registro contendo os valores dos atributos mantidos pelo estreitamento ocorrentes neste registro de ARQ.
1.2 - Adicionar à lista resultado o registro formado.

ESTREIT4 (L)

- Condição:

L é uma lista de dados classificada segundo alguma ordem dos atributos mantidos pelo estreitamento.

- Resultado:

Lista de dados classificada segundo a ordem de L.

- Método:

Passos:

1 - ANTERIOR ← valores dos atributos mantidos pelo estreitamento ocorrentes no primeiro registro de L.

2 - Adicionar à lista resultado um registro contendo os valores mantidos em ANTERIOR.

3 - Para cada registro de L, fazer:

3.1 - Se os valores dos atributos mantidos pelo estreitamento ocorrentes neste registro de L são diferentes dos mantidos em ANTERIOR, fazer:

3.1.1 - ANTERIOR ← valores dos atributos mantidos pelo estreitamento ocorrentes neste registro de L.

3.1.2 - Adicionar à lista resultado um registro contendo os valores mantidos em ANTERIOR.

ESTREIT5 (ARQ,ORD)

- Condição:

ARQ é um arquivo/lista de dados.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD dos atributos mantidos pelo estreitamento. Do contrário, resulta uma lista de dados classificada segundo a ordem em que os atributos mantidos pelo estreitamento aparecem nos registros de ARQ.

- Método:

Passos:

1 - Se ORD for uma ordem válida, então classificar ARQ segundo a ordem ORD dos atributos mantidos pelo estreitamento (produzindo como saída registros já estreitados, com eliminação de duplicatas).

2 - Do contrário, classificar ARQ segundo a ordem em que os atributos mantidos pelo estreitamento aparecem nos registros de ARQ (produzindo como saída registros já estreitados, com eliminação de duplicatas).

5 - DIF

Algoritmos apresentados:

- DIF1: percorre seqüencialmente duas listas de dados classificadas segundo atributos que não são, necessariamente, chave primária.

- DIF2: é mostrado esquematicamente na figura 5.1. Usa como operandos dois arquivos de dados (ARQ1 e ARQ2) possuindo arquivos de inversões sobre mesmos atributos. É composto de quatro passos fundamentais:

  (1) Percorre seqüencialmente os dois arquivos de inversões, gerando duas listas de idregs (LIDS1 e LIDS2, respectivamente) contendo idregs de registros de ARQ1 e ARQ2 que são candidatos à interseção entre estes dois arquivos.

  (2) Classifica a lista de idregs de registros de ARQ1 candidatos à interseção (produzindo LIDS1C).

  (3) Materializa a lista de idregs de registros de ARQ2 candidatos à interseção (buscando os registros de dados e gerando LT2).

  (4) Percorre seqüencialmente ARQ1, verificando,para cada registro, se o mesmo é candidato à interseção (seu idreg está em LIDSC) e, em caso afirmativo, verificando se este registro ocorre em ARQ2 (se está em LT2), produzindo então o resultado.

- DIF2M: percorre seqüencialmente dois arquivos de inversões sobre chave primária, buscando os respectivos registros de dados.

- DIF3: é mostrado esquematicamente na figura 5.2. Usa como operandos um arquivo de dados (ARQ1) e uma lista de dados classificada (L2). É composto de três passos fundamentais:

  (1) Percorre seqüencialmente a lista de dados e um arquivo de inversões sobre ARQ1, gerando uma lista de idregs (LIDS) de registros de ARQ1 candidatos à interseção entre os dois operandos e uma lista de registros de dados (LT2) provenientes de L2, candidatos à interseção.

  (2) Classifica a lista de idregs de registros de ARQ1 candidatos à interseção (produzindo LIDSC).

  (3) Percorre seqüencialmente ARQ1, verificando, para cada registro, se o mesmo é candidato à interseção (seu idreg está em LIDSC) e, em caso afirmativo, verificando se este registro ocorre em L2 (se está em LT2), produzindo então o resultado.

- DIF4: é mostrado esquematicamente na figura 5.3. Usa como operandos uma lista de dados classificada (L1) e um arquivo de dados (ARQ2). É composto de quatro passos fundamentais:

  (1) Percorre seqüencialmente a lista de dados e um arquivo de inversões sobre ARQ1, gerando uma lista de idregs (LIDS) de registros de ARQ2 candidatos à interseção entre os dois operandos e uma lista de registros de dados (LT1) provenientes de L1, candidatos à interseção; produz como resultado parcial uma lista contendo os registros de L1 que não são candidatos à interseção.

(2) Classifica a lista de idregs de registros de ARQ2 candidatos à interseção (produzindo LIDSC).

(3) Percorre seqüencialmente ARQ2, verificando, para cada registro, se o mesmo é candidato à interseção (seu idreg está em LIDSC) e, em caso afirmativo, verificando se este registro ocorre em L1 (se está em LT1), produzindo marcações naqueles registros de LT1 que efetivamente pertencem à interseção entre os dois operandos.

(4) Adiciona ao resultado parcial todos os registros de LT1 que não estão marcados como pertencentes à interseção entre os operandos, produzindo o resultado final.

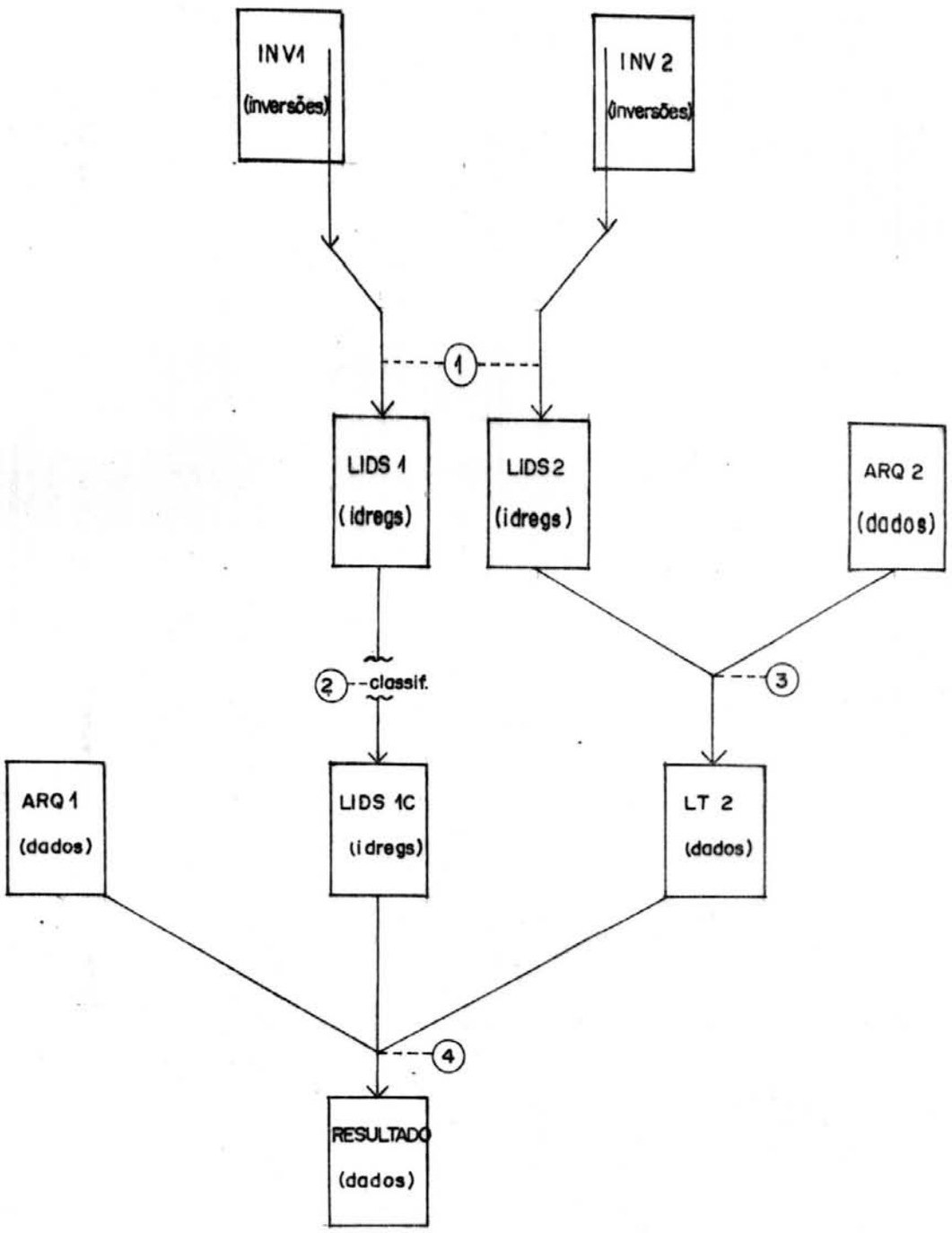


FIGURA 5.1: Algoritmo DIF2

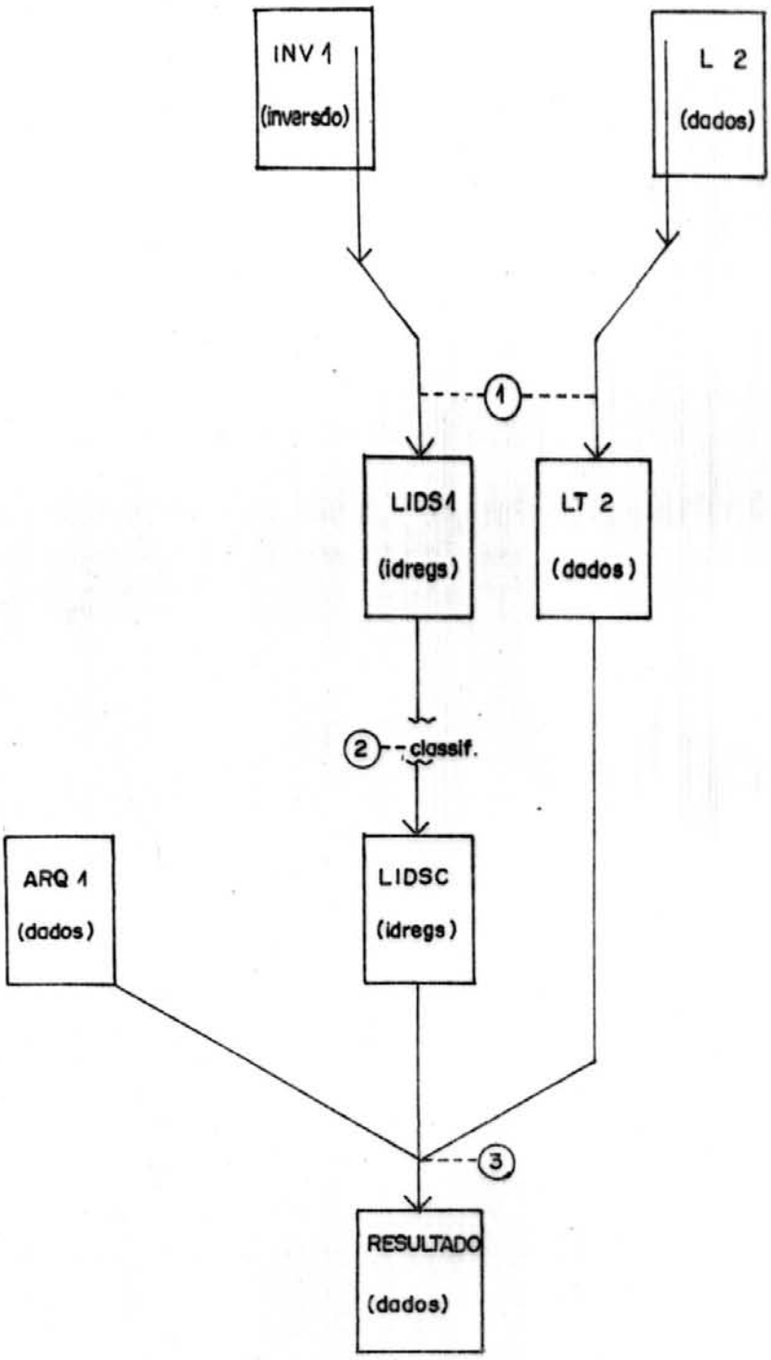


FIGURA 5.2: Algoritmo DIF3

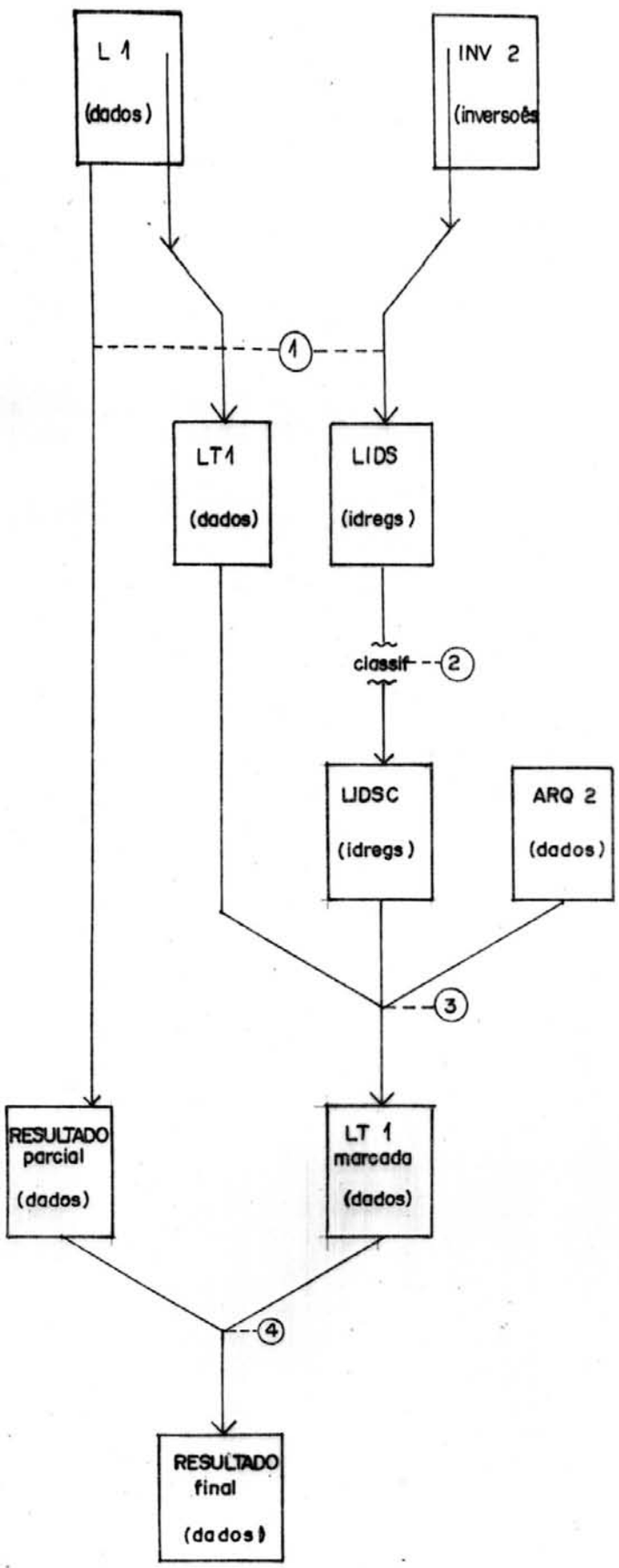


FIGURA 5.3: Algoritmo DIF4

DIF1 (ARQ1, ARQ2, ORD)

- Condição:

ARQ1, ARQ2 são arquivos/listas de dados.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD. Do contrário, resulta uma lista de dados classificada:

I) Segundo a chave primária de ARQ1, se nem ARQ1 nem ARQ2 estão classificadas.

II) Segundo a ordem de classificação do arquivo/lista de dados de maior cardinalidade, se ambos ARQ1, ARQ2 já estiverem classificados segundo diferentes ordens.

- Método:

Estruturas de dados utilizadas:

LARQ1 - é o próprio ARQ1, ou uma lista de dados contendo as tuplas de ARQ1 classificadas.

LARQ2 - é o próprio ARQ2, ou uma lista de dados contendo as tuplas de ARQ2 classificadas.

Passos:

1 - Se ORD for uma ordem válida, fazer:

1.1 - Verificar se ARQ1 já está classificado segundo a ordem ORD.
Sim → seguir.
Não → classificar ARQ1 segundo ORD (gerando LARQ1).

1.2 - Verificar se ARQ2 já está classificado segundo a ordem ORD.
Sim → seguir.
Não → classificar ARQ2 segundo ORD (gerando LARQ2).

2 - Se ORD não for uma ordem válida, fazer:

2.1 - Verificar se ambos ARQ1, ARQ2 já estão classificados sob diferentes ordens.

Sim → classificar o arquivo de menor cardinalidade, de acordo com a ordem de classificação do outro (gerando LARQ1, se o arquivo a classificar for ARQ1, ou LARQ2, caso contrário).

Não → classificar ambos ARQ1,ARQ2 segundo a chave primária de ARQ1 (gerando LARQ1, LARQ2).

3 - ANTERIOR ← valores inválidos para os atributos de ordenação.

4 - Para cada registro de LARQ1, fazer:

4.1 - Se ANTERIOR $\neq$ valores dos atributos de ordenação ocorrentes neste registro de LARQ1, fazer:

4.1.1 - ANTERIOR ← valores dos atributos de ordenação ocorrentes neste registro de LARQ1.

4.1.2 - Buscar em LARQ2 (a partir do registro corrente ou do primeiro registro, se esta for a primeira execução deste comando) um registro com valores iguais aos mantidos em ANTERIOR, nos atributos de ordenação.

4.1.3 - P ← idreg do registro corrente de LARQ2.

4.2 - D ← verdadeiro.

4.3 - Para cada registro de LARQ2, a partir do registro apontado por P, cujos valores dos atributos de ordenação sejam iguais aos mantidos em ANTERIOR, enquanto D for verdadeiro, fazer:

4.3.1 - Verificar se os registros correntes de LARQ1 e LARQ2 são iguais em todos os campos.

Sim → D ← falso

Não → seguir.

4.4 - Se D = Verdadeiro então adicionar à lista resultado o registro corrente de LARQ1.

DIF2 (ARQ1, ARQ2)

- Condição:

ARQ1, ARQ2 são arquivos de dados que possuem arquivos de inversões sobre mesmos atributos.

- Resultado:

Lista de dados não classificada (já que é obtida a partir de ARQ1, que não está classificado).

- Método:

Estruturas de dados utilizadas:

INV1 - arquivo de inversões sobre ARQ1.
INV2 - arquivo de inversões sobre ARQ2.
LIDS1,LIDS2 - listas de idregs (de registros de ARQ1, ARQ2, respectivamente).
LT2 - lista de dados (de registros de ARQ2).
LIDS1C - lista LIDS1 classificada.

Passos:

1 - Para cada registro de INV1, fazer:
1.1 - Buscar em INV2 um registro de inversão cuja construção sob nome ITENS tenha conteúdo igual ao da construção sob nome ITENS do registro corrente de INV1.
1.2 - Se encontrado, fazer:
1.2.1 - Para cada idreg mantido no registro corrente de INV2, fazer:
1.2.1.1 - Adicionar à lista LIDS2 um registro contendo esse idreg.
1.2.2 - Para cada idreg mantido no registro corrente de INV1, fazer:
1.2.2.1 - Adicionar à lista LIDS1 um registro contendo esse idreg.

2 - Classificar a lista LIDS1 (gerando LIDS1C).

3 - Para cada registro de LIDS2, fazer:

3.1 - Buscar o registro de ARQ2 apontado pelo idreg mantido nesse registro de LIDS2.

3.2 - Adicionar este registro à lista LT2.

4 - Para cada registro de ARQ1, fazer:

4.1 - Verificar se o idreg do mesmo está presente em LIDS1C (pesquisa binária).

4.2 - Se estiver presente, então verificar se este registro está em LT2 (pesquisa binária modificada*, usando como chave os atributos que compõem a chave primária dos arquivos de inversões utilizados).

Sim → seguir.

Não → adicionar este registro de ARQ1 à lista resultado.

4.3 - Se não estiver presente, então adicionar este registro de ARQ1 à lista resultado.

- Observação: A lista LIDS2 não é classificada por idreg a fim de manter-se sua classificação pelos mesmos atributos pelos quais estão definidas as inversões INV1 e INV2.

---

* É aqui denominada "pesquisa binária modificada" a modalidade de pesquisa binária em que a chave de busca não é, necessariamente, chave primária: admite duplicatas.

DIF2M (ARQ1, ARQ2)

- Condição:

ARQ1, ARQ2 são arquivos de dados que possuem mesma chave primária.

- Resultado:

Lista de dados classificada segundo a chave primária de ARQ1 e ARQ2.

- Método:

Estruturas de dados utilizadas:

INV1 - Arquivo de inversões sobre ARQ1.
INV2 - Arquivo de inversões sobre ARQ2.

Passos:

1 - Para cada registro de INV1, fazer:
1.1 - Buscar em INV2 um registro de inversão cuja construção sob nome ITENS tenha mesmo conteúdo que esta construção, no registro corrente de INV1.
1.2 - Se encontrado, fazer:
1.2.1 - Buscar em ARQ1 o registro apontado pelo idreg mantido no registro corrente de INV1.
1.2.2 - Buscar em ARQ2 o registro apontado pelo idreg mantido no registro corrente de INV2.
1.2.3 - Verificar se ambos os registros são iguais, em todos os campos:
Sim → seguir.
Não → adicionar à lista resultado o registro trazido de ARQ1.
1.3 - Do contrário, fazer:
1.3.1 - Buscar em ARQ1 o registro apontado pelo idreg mantido no registro corrente de INV1.
1.3.2 - Adicionar o mesmo à lista resultado.

DIF3 (ARQ1, L2)

- Condição:

ARQ1 é um arquivo de dados. L2 é uma lista de dados. Existe um atributo ou conjunto de atributos a pelo qual L2 está classificada e pelo qual ARQ1 possui um arquivo de inversões.

- Resultado:

Lista de dados não classificada (já que ARQ1 é um arquivo de dados).

- Método:

Estruturas de dados utilizadas:

INV1 - arquivo de inversões sobre ARQ1.
LT2 - lista de dados (de registros de L2).
LIDS - lista de idregs (de registros de ARQ1).
LIDSC - lista LIDS classificada.

Passos:

1 - Para cada registro de L2, fazer:
1.1 - Buscar em INV1 um registro de inversão cuja construção sob nome ITENS tenha mesmo conteúdo que esse registro de L2, nos campos correspondentes.
1.2 - Se encontrado, fazer:
1.2.1 - Adicionar à lista LT2 esse registro de L2.
1.2.2 - Para cada idreg mantido nesse registro de INV1, fazer:
1.2.2.1 - Adicionar à lista LIDS este idreg.

2 - Classificar a lista LIDS (gerando LIDSC).

3 - Para cada registro de ARQ1, fazer:
3.1 - Verificar se o idreg desse registro está pre-

sente em LIDSC (pesquisa binária).

3.2 - Se estiver presente, fazer:

3.2.1 - Verificar se este registro está em LT2 (pesquisa binária modificada).

Sim → seguir.

Não → Adicionar à lista resultado este registro de ARQ1.

3.3 - Se não estiver presente, adicionar à lista resultado este registro de ARQ1.

DIF4 (L1, ARQ2)

- Condição:

L1 é uma lista de dados. ARQ2 é um arquivo de dados. Existe um atributo ou conjunto de atributos a pelo qual L1 está classificada e pelo qual ARQ2 possui um arquivo de inversões.

- Resultado:

Lista de dados não classificada.

- Método:

Estruturas de dados utilizadas:

INV2 - arquivo de inversões sobre ARQ2.
LT1 - lista de registros de L1 (ver observação no final deste algoritmo).
LIDS - lista de idregs (de registros de ARQ2).
LIDSC - lista LIDS classificada.

Passos:

1 - Para cada registro de L1, fazer:
1.1 - Buscar em INV2 um registro de inversão cuja construção sob nome ITENS tenha mesmo conteúdo que esse registro de L1, nos campos correspondentes.
1.2 - Se encontrado, fazer:
1.2.1 - Adicionar à lista LT1 esse registro de L1.
1.2.2 - Para cada idreg mantido nesse registro de INV2, fazer:
1.2.2.1 - Adicionar à lista LIDS um registro contendo este idreg.
1.2 - Se não encontrado, adicionar este registro de L1 à lista resultado.

2 - Classificar a lista LIDS (gerando LIDSC).

3 - Para cada registro de ARQ2, fazer:

3.1 - Verificar se o idreg desse registro está presente em LIDSC.

3.2 - Se estiver presente, fazer:

3.2.1 - Verificar se este registro se encontra em LT1.

Sim → preencher o campo "marca" deste registro em LT1 como interseção.

Não → seguir.

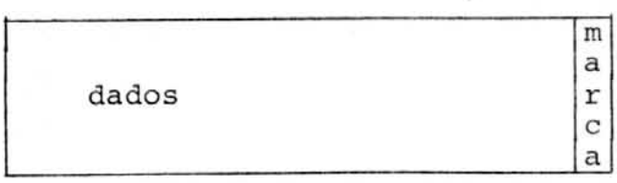
4 - Para cada registro de LT1, fazer:

4.1 - Verificar se o mesmo está marcado como interseção.

Sim → seguir

Não → adicionar à lista resultado este registro de LT1.

- Observação: Cada registro da lista LT1, para este algoritmo, especificamente, é da forma:



dados - corresponde a um registro de dados de L1 candidato à interseção entre os dois operandos.

marca - permitirá saber se esse registro de L1 pertence ou não à interseção dos dois operandos dessa operação DIF.

6 - UNI

Algoritmos apresentados:

- UNI1: percorre seqüencialmente duas listas de dados classificadas segundo atributos que não são, necessariamente, chave primária, fazendo a intercalação das mesmas e eliminando as duplicatas.

- UNI2: classifica os dois arquivos/listas de dados, através de uma única chamada da primitiva de classificação, eliminando as duplicatas.

- UNI3: utiliza um método semelhante ao proposto no algoritmo DIF2 para determinar quais os registros candidatos a participarem da interseção entre os dois arquivos/listas. Percorre seqüencialmente o primeiro operando (ARQ1), adicionando todos os seus registros à lista resultado. Por último percorre o segundo operando (ARQ2), também seqüencialmente, adicionando à lista resultado aqueles registros que não pertencem à interseção.

- UNI4: utiliza um método semelhante ao proposto no algoritmo DIF3 para determinar os registros candidatos à interseção entre os operandos. Adiciona à lista resultado todos os registros de seu segundo operando (uma lista classificada), e então adiciona à lista resultado apenas aqueles registros do primeiro operando que não pertencem à interseção.

UNI1 (ARQ1, ARQ2, ORD)

- Condição:

ARQ1, ARQ2 são arquivos/listas de dados.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD.

Se ORD não for uma ordem válida, resulta uma lista de dados classificada:

I) Segundo a chave primária de ARQ1, se nem ARQ1 nem ARQ2 estão classificados*.

II) Segundo a ordem de classificação do arquivo/lista de dados de maior cardinalidade, se ambos ARQ1, ARQ2 já estiverem classificados segundo diferentes ordens.

- Método:

Estruturas de dados utilizadas:

LARQ1 - é o próprio ARQ1, ou uma lista de dados contendo ARQ1 classificado.

LARQ2 - é o próprio ARQ2, ou uma lista de dados contendo ARQ2 classificado.

Passos:

1 - Se ORD for uma ordem válida, então:

1.1 - Verificar se ARQ1 está classificado segundo a ordem ORD.
Sim → seguir.
Não → classificar ARQ1 segundo a ordem ORD (gerando LARQ1).

---

* Atualmente, este algoritmo não está sendo escolhido, neste caso. A união de dois arquivos/listas de dados não classificados é resolvida pelo algoritmo UNI3 (descrito logo a seguir).

1.2 - Verificar se ARQ2 está classificado segundo a ordem ORD.
Sim → seguir.
Não → classificar ARQ2 segundo a ordem ORD (gerando LARQ2).

2 - Se ORD não for uma ordem válida, então:
2.1 - Verificar se ambos ARQ1, ARQ2 já estão classificados em diferentes ordens:
Sim → classificar o arquivo de menor cardinalidade, de acordo com a ordem de classificação do outro (gerando LARQ1, se o arquivo a classificar for ARQ1, ou LARQ2, caso contrário).
Não → classificar ambos ARQ1, ARQ2 segundo a chave primária de ARQ1 (gerando LARQ1, LARQ2).

3 - ANTERIOR ← valores inválidos para os atributos de ordenação.

4 - Para cada registro de LARQ1, fazer:
4.1 - Se ANTERIOR ≠ valores dos atributos de ordenação ocorrentes neste registro de LARQ1, então:
4.1.1 - ANTERIOR ← valores dos atributos de ordenação ocorrentes neste registro de LARQ1.
4.1.2 - P ← posição do registro corrente de LARQ2 (ou do primeiro registro de LARQ2, se este é o primeiro valor atribuído a P).
4.1.3 - Para cada registro de LARQ2 contendo, nos atributos de ordenação, valores < (se a ordem é crescente) ou > (se a ordem é decrescente) que o registro corrente de LARQ1, a partir do registro apontado por P, fazer:

4.1.3.1 - Adicionar à lista resultado este registro de LARQ2.

4.1.4 - Se algum registro foi adicionado à lista resultado no passo anterior, então:
P ← posição do registro corrente de LARQ2.

4.1.5 - ENCONTRADO ← Falso.

4.1.6 - Se o registro corrente de LARQ2 contém, nos atributos de ordenação, valores iguais aos do registro corrente de LARQ1, fazer:

4.1.6.1 - Para cada registro de LARQ2 com valores iguais aos do registro corrente de LARQ1, nos atributos de ordenação (a partir do registro apontado por P), fazer:

4.1.6.1.1 - Verificar se os registros correntes de LARQ1 e LARQ2 são iguais em todos os campos.
Sim → ENCONTRADO ← Verdadeiro.

4.1.6.1.2 - Adicionar à lista resultado o registro corrente de LARQ2.

4.1.7 - Se ENCONTRADO = Falso então adicionar à lista resultado o registro corrente de LARQ1.

4.2 - Do contrário, fazer:
/* Neste caso os atributos de ordenação têm mesmo valor que os mantidos em ANTERIOR */

4.2.1 - ENCONTRADO ← Falso.

4.2.2 - Para cada registro de LARQ2, a partir do registro apontado por P, enquanto os valores dos atributos de ordenação não forem > (se a ordem é crescente) ou < (se a ordem é decrescente) que os respectivos atributos no registro corrente de LARQ1, fazer:

4.2.2.1 - Verificar se os registros correntes de LARQ1 e LARQ2 são iguais em todos os campos.
Sim → ENCONTRADO ← Verdadeiro
Não → seguir.

4.2.3 - Se ENCONTRADO = Falso, então adicionar à lista resultado o registro corrente de LARQ1.

5 - Para cada registro restante de LARQ2 (a partir do registro apontado por P), fazer:

5.1 - Adicionar o mesmo à lista resultado.

UNI2 (ARQ1, ARQ2, ORD)

- Condição:

ARQ1, ARQ2 são arquivos/listas de dados.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD. Do contrário, resulta uma lista de dados classificada segundo a chave primária de ARQ1.

- Método:

Passos:

1 - Classificar os dois arquivos/listas de dados segundo a ordem ORD (se ORD for uma ordem válida), ou segundo a chave primária de ARQ1 (em caso contrário), através de uma única chamada da primitiva de classificação, com eliminação de duplicatas no resultado.

UNI3 (ARQ1, ARQ2)

- Condição:

ARQ1, ARQ2 são arquivos de dados para os quais existem definidos arquivos de inversões sobre um mesmo conjunto de atributos.

- Resultado:

Lista de dados não classificada.

- Método:

Estruturas de dados utilizadas:

INV1 - Arquivo de inversões sobre ARQ1.
INV2 - Arquivo de inversões sobre ARQ2.
ID1 - Lista de idregs (de registros de ARQ1).
ID2 - Lista de idregs (de registros de ARQ2).
INT - Lista de dados.
ID2C - Lista ID2 classificada.

Passos:

1 - Para cada registro de INV1, fazer:
1.1 - Buscar em INV2 um registro de inversão cujo conteúdo da construção sob nome ITENS seja igual ao conteúdo da mesma construção, no registro corrente de INV1.
1.2 - Se encontrado, fazer:
1.2.1 - Para cada idreg mantido no registro corrente de INV1, fazer:
1.2.1.1 - Adicionar à lista ID1 um registro contendo este idreg.
1.2.2 - Para cada idreg mantido no registro corrente de INV2, fazer:
1.2.2.1 - Adicionar à lista ID2 um registro contendo este idreg.

2 - Para cada registro de ID1, fazer:

2.1 - Buscar o registro de ARQ1 apontado pelo idreg mantido nesse registro.

2.2 - Adicionar o mesmo à lista INT.

3 - Classificar a lista ID2 (gerando ID2C).

4 - Para cada registro de ARQ1, fazer:

4.1 - Adicionar à lista resultado o registro mantido no mesmo.

5 - Para cada registro de ARQ2, fazer:

5.1 - Verificar se o idreg desse registro de ARQ2 está em ID2 (pesquisa binária).
Não → adicionar este registro à lista resultado.
Sim → verificar se este registro se encontra em INT (pesquisa binária modificada).
Não → adicionar o mesmo à lista resultado.
Sim → seguir.

UNI4 (ARQ1, L2)

- Condição:

ARQ1 é um arquivo de dados sobre o qual existe definido um arquivo de inversões sobre um conjunto a de atributos. L2 é uma lista de dados classificada segundo o mesmo conjunto a de atributos.

- Resultado:

Lista de dados não classificada.

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ1.
ID1 - Lista de idregs (de registros de ARQ1).
INT - Lista de dados.
ID1C - Lista ID1 classificada.

Passos:

1 - Para cada registro de L2, fazer:
1.1 - Adicionar o mesmo à lista resultado.
1.2 - Buscar em INV um registro cuja construção sob nome ITENS tenha como conteúdo os mesmos valores que o registro corrente de L2, nos campos correspondentes.
1.3 - Se encontrado, fazer:
1.3.1 - Adicionar à lista INT este registro de L2.
1.3.2 - Para cada idreg mantido no registro corrente de INV, fazer:
1.3.2.1 - Adicionar à lista ID1 um registro contendo este idreg.

2 - Classificar a lista ID1 (gerando ID1C).

3 - Para cada registro de ARQ1, fazer:

3.1 - Verificar se o idreg desse registro de ARQ1 está em ID1C (pesquisa binária).

Não → adicionar este registro à lista resultado.

Sim → verificar se este registro se encontra em INT (pesquisa binária modificada).

Não → adicionar este registro à lista resultado.

Sim → seguir.

7 - ISEC

Algoritmos apresentados:

- ISEC1: percorre seqüencialmente duas listas de dados classificadas segundo atributos que não são, necessariamente, chave primária.

- ISEC2: utiliza um método semelhante ao proposto no algoritmo DIF2, determinando duas listas de idregs correspondentes aos registros candidatos à interseção. Materializa a lista de idregs de menor cardinalidade e classifica a outra lista. Percorre seqüencialmente a lista de idregs classificada, buscando os respectivos registros de dados e pesquisando a existência dos mesmos na lista materializada de candidatos à interseção.

- ISEC2M: percorre seqüencialmente dois arquivos de inversões sobre chave primária, buscando os registros de dados candidatos à interseção.

- ISEC3: classifica os dois arquivos/listas de dados, através de uma única chamada da primitiva de classificação, sem eliminar duplicatas. Percorre seqüencialmente a lista resultante da classificação, adicionando ao resultado os registros que apresentam duplicatas.

- ISEC4: utiliza um método semelhante ao proposto no algoritmo DIF3, para determinar os registros candidatos à interseção. Materializa a lista de idregs, verificando, a cada registro de dados obtido, se o mesmo se encontra na lista de registros candidatos à interseção, e gerando a lista resultado.

ISEC1 (ARQ1, ARQ2, ORD)

- Condição:

ARQ1, ARQ2 são arquivos/listas de dados.

- Resultado:

Se ORD for uma ordem válida resulta uma lista de dados classificada segundo a ordem ORD.

Se ORD não for uma ordem válida, resulta uma lista de dados classificada:

I) Segundo a chave primária de ARQ1, se nam ARQ1 nem ARQ2 estão classificados*;

II) Segundo a ordem de classificação do arquivo/lista de dados de maior cardinalidade, se ambos ARQ1, ARQ2 já estão classificados sob diferentes ordens.

- Método:

Estruturas de dados utilizadas:

LARQ1 - é o próprio ARQ1, ou uma lista de dados contendo ARQ1 classificado.

LARQ2 - é o próprio ARQ2, ou uma lista de dados contendo ARQ2 classificado.

Passos:

1 - Se ORD for uma ordem válida, então:

1.1 - Verificar se ARQ1 está classificado segundo a ordem ORD.
Sim → seguir.
Não → classificar ARQ1 segundo a ordem ORD (gerando LARQ1).

---

* Atualmente, se este caso ocorrer o algoritmo efetivamente escolhido é o UNI4 (descrito mais adiante).

1.2 - Verificar se ARQ2 está classificado segundo a ordem ORD.
Sim → seguir.
Não → classificar ARQ2 segundo a ordem ORD (gerando LARQ2).

2 - Se ORD não for uma ordem válida, então:

2.1 - Verificar se ambos ARQ1, ARQ2 já estão classificados sob diferentes ordens.
Sim → classificar o arquivo de menor cardinalidade, de acordo com a ordem de classificação do outro (gerando LARQ1, se o arquivo a classificar for ARQ1, ou LARQ2, caso contrário).
Não → classificar ambos ARQ1, ARQ2 segundo a chave primária de ARQ1 (gerando LARQ1, LARQ2).

3 - ANTERIOR ← valores inválidos para os atributos de ordenação.

4 - Para cada registro de LARQ1, fazer:

4.1 - Se ANTERIOR ≠ valores dos atributos de ordenação ocorrentes neste registro de LARQ1, fazer:

4.1.1 - ANTERIOR ← valores dos atributos de ordenação ocorrentes neste registro de LARQ1.

4.1.2 - Buscar em LARQ2 (a partir do registro corrente ou do primeiro registro, se esta for a primeira execução deste comando), um registro com valores iguais aos mantidos em ANTERIOR, nos atributos de ordenação.

4.1.3 - P ← posição do registro corrente de LARQ2.

4.2 - ENCONTRADO ← falso.

4.3 - Para cada registro de LARQ2, a partir do registro apontado por P, cujos valores dos atributos de ordenação sejam iguais aos mantidos em ANTE

RIOR, enquanto ENCONTRADO for falso, fazer:

4.3.1 - Verificar se os registros correntes de LARQ1 e LARQ2 são iguais em todos os campos.
Sim → ENCONTRADO ← verdadeiro.
Não → seguir.

4.4 - Se ENCONTRADO = verdadeiro, então adicionar à lista resultado o registro corrente de LARQ1.

ISEC2 (ARQ1, ARQ2)

- Condição:

ARQ1, ARQ2 são arquivos de dados para os quais existem definidos arquivos de inversões sobre um mesmo conjunto de atributos.

- Resultado:

Lista de dados não classificada.

- Método:

Estruturas de dados utilizadas:

INV1 - Arquivo de inversões sobre ARQ1.
INV2 - Arquivo de inversões sobre ARQ2.
ID1 - Lista de idregs (de registros de ARQ1).
ID2 - Lista de idregs (de registros de ARQ2).
INT - Lista de dados.
IDC - Lista ID1 classificada.

Passos:

1 - Para cada registro de INV1, fazer:
1.1 - Buscar em INV2 um registro de inversão cujo conteúdo da construção sob nome ITENS seja igual ao conteúdo dessa construção, no registro corrente de INV1.
1.2 - Se encontrado, fazer:
1.2.1 - Para cada idreg mantido no registro corrente de INV1, fazer:
1.2.1.1 - Adicionar à lista ID1 um registro contendo este idreg.
1.2.2 - Para cada idreg mantido no registro corrente de INV2, fazer:
1.2.2.1 - Adicionar à lista ID2 um registro contendo este idreg.

2 - Se ID2 é a lista de idregs de menor cardinalidade, então:

2.1 - Para cada registro de ID1, fazer:

2.1.1 - Buscar o registro de ARQ1 apontado pelo idreg mantido nesse registro.

2.1.2 - Adicionar o mesmo à lista INT.

2.2 - Classificar a lista ID2 (gerando IDC).

3 - Se ID2 não é a lista de idregs de menor cardinalidade, então:

3.1 - Para cada registro de ID2, fazer:

3.1.1 - Buscar o registro de ARQ2 apontado pelo idreg mantido nesse registro.

3.1.2 - Adicionar o mesmo à lista INT.

3.2 - Classificar a lista ID1 (gerando IDC).

4 - Para cada registro de IDC, fazer:

4.1 - Buscar o registro (de ARQ1 ou ARQ2, conforme execução dos passos 2, 3) apontado pelo idreg mantido nesse registro.

4.2 - Verificar se o registro trazido está presente em INT (pesquisa binária modificada).
Sim → adicionar este registro à lista resultado.
Não → seguir.

ISEC2M (ARQ1, ARQ2)

- Condição:

ARQ1, ARQ2 são arquivos de dados com mesma chave primária.

- Resultado:

Lista de dados classificada segundo a chave primária de ARQ1 e ARQ2.

- Método:

Estruturas de dados utilizadas:

INV1 - Arquivo de inversões sobre a chave primária de ARQ1.
INV2 - Arquivo de inversões sobre a chave primária de ARQ2.

Passos:

1 - Para cada registro de INV1, fazer:
1.1 - Buscar em INV2 um registro cuja construção sob nome ITENS tenha mesmo conteúdo que a referida construção, no registro corrente de INV2.
1.2 - Se encontrado, fazer:
1.2.1 - Buscar o registro de ARQ1 apontado pelo idreg mantido no registro corrente de INV1.
1.2.2 - Buscar o registro de ARQ2 apontado pelo idreg mantido no registro corrente de INV2.
1.2.3 - Verificar se as mesmas são iguais, em todos os campos.
Sim → adicionar à lista resultado o registro trazido de ARQ1.
Não → seguir.

ISEC3 (ARQ1, ARQ2, ORD)

- Condição:

ARQ1, ARQ2 são arquivos/listas de dados.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD. Do contrário, resulta uma lista de dados classificada segundo a chave primária de ARQ1.

- Método:

Estruturas de dados utilizadas:

LC - lista de registros de dados, classificada.

Passos:

1 - Classificar os dois arquivos/listas de dados segundo a ordem ORD (se ORD for uma ordem válida), ou segundo a chave primária de ARQ1 (em caso contrário), através de uma única chamada da primitiva de classificação, sem eliminação de duplicatas no resultado (gerando LC).

2 - Buscar o primeiro registro da lista LC.

3 - ANTERIOR ← primeiro registro da lista LC.

4 - Enquanto a lista LC não estiver esgotada, fazer:
4.1 - Buscar o próximo registro da lista LC.
4.2 - Se o conteúdo do registro corrente de LC é igual ao mantido em ANTERIOR, então:
4.2.1 - Adicionar à lista resultado o registro corrente de LC.
4.2.2 - Buscar o próximo registro da lista LC.
4.3 - ANTERIOR ← registro corrente da lista LC.

ISEC4 (ARQ1, L2)

- Condição:

ARQ1 é um arquivo de dados sobre o qual existe definido um arquivo de inversões sobre um conjunto a ordenado de atributos. L2 é uma lista de dados classificada segundo o mesmo conjunto a ordenado de atributos.

- Resultado:

Lista de dados classificada segundo L2.

- Método:

Estruturas de dados utilizadas:

INV - arquivo de inversões sobre ARQ1.
INT - lista de dados (de registros de L2).
ID1 - lista de idregs (de registros de ARQ1).

Passos:

1 - Para cada registro de L2, fazer:
1.1 - Buscar em INV um registro de inversão cujo conteúdo da construção sob nome ITENS seja igual ao conteúdo dos campos correspondentes, nesse registro de L2.
1.2 - Se encontrado, fazer:
1.2.1 - Adicionar à lista INT esse registro de L2.
1.2.2 - Para cada idreg contido nesse registro de inversão, fazer:
1.2.2.1 - Adicionar à lista ID1 um registro contendo este idreg.

2 - Para cada registro da lista ID1, fazer:
2.1 - Buscar o registro de ARQ1 apontado pelo idreg mantido nesse registro.
2.2 - Verificar se o mesmo está presente em INT (pesquisa binária modificada).

Sim → adicionar à lista resultado esse registro de ARQ1.

Não → seguir.

8 - AGRUPAR

Algoritmos apresentados:

- AGRUPAR1: percorre seqüencialmente um arquivo de inversões definido sobre os atributos de agrupamento, buscando os registros de dados para compor os ligados de cada associação.

- AGRUPAR2: percorre seqüencialmente uma lista de dados classificada segundo os atributos de agrupamento.

AGRUPAR1 (ARQ)

- Condição:

ARQ é um arquivo de dados possuindo um arquivo de inversões definido exatamente sobre os atributos de agrupamento ou os atributos de agrupamento são "os primeiros" de uma chave de inversão sobre ARQ1.

- Resultado:

Uma lista de dados de ligantes (LISTA DE LIGANTES) classificada segundo os atributos de agrupamento mantidos no arquivo de inversões, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante está classificado segundo os atributos adicionais existentes na chave de inversão, além dos atributos de agrupamento (no caso de os atributos de agrupamento serem os únicos componentes da chave de inversão, os grupos de ligados não apresentam classificação alguma).

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ.

Passos:

1 - ANTERIOR ← valores dos atributos de agrupamento mantidos na construção sob nome ITENS do primeiro registro de INV.

2 - ULTLIGADO ← (próxima posição a ser ocupada na LISTA DE LIGADOS) - 1.

3 - PRIMLIGADO ← ULTLIGADO + 1.

4 - Para cada registro de INV, fazer:

4.1 - Se os valores dos atributos de agrupamento man-

tidos na construção sob nome ITENS desse registro de INV são diferentes dos mantidos em ANTERIOR, fazer:

4.1.1 - Adicionar à lista de LIGANTES um registro contendo os valores de:
- ANTERIOR
- PRIMLIGADO
- ULTLIGADO

4.1.2 - ANTERIOR ← valores dos atributos de agrupamento mantidos no registro corrente de INV.

4.1.3 - PRIMLIGADO ← ULTLIGADO + 1.

4.2 - Para cada idreg mantido no registro corrente de INV, fazer:

4.2.1 - Buscar o registro de ARQ apontado pelo mesmo.

4.2.2 - Formar um registro com os campos desse registro de ARQ que não constituem atributos de agrupamento.

4.2.3 - Adicionar à LISTA DE LIGADOS o registro formado.

4.2.4 - ULTLIGADO ← ULTLIGADO + 1.

5 - Adicionar à LISTA DE LIGANTES um registro contendo os valores de:
- ANTERIOR
- PRIMLIGADO
- ULTLIGADO.

AGRUPAR2 (ARQ,ORD)

- Condição:

ARQ é um arquivo/lista de dados.

- Resultado:

Uma lista de dados de ligantes (LISTA DE LIGANTES) classificada:

I) Se ORD for uma ordem válida, segundo a ordem ORD dos atributos de agrupamento;

II) Se ORD não for uma ordem válida, segundo a ordem em que os atributos de agrupamento aparecem nos registros de ARQ).

... e uma lista de dados de ligados (LISTA DE LIGADOS) onde cada grupo de ligados de um mesmo ligante se encontra classificado segundo os atributos adicionais existentes na ordem de classificação de ARQ além dos atributos de agrupamento.

- Método:

Estruturas de dados utilizadas:

LARQ - é o próprio ARQ, se já estiver classificado, ou uma lista de dados contendo ARQ classificado.

Passos:

1 - Se ORD for uma ordem válida, fazer:

1.1 - Verificar se ARQ já está classificado segundo a ordem ORD dos atributos de agrupamento.
Sim → seguir.
Não → classificar ARQ segundo a ordem ORD dos atributos de agrupamento (gerando LARQ).

2 - Se ORD não for uma ordem válida, fazer:

2.1 - Classificar ARQ segundo a ordem em que os atri-

butos de agrupamento aparecem em seus registros.

3 - ANTERIOR ← valores dos atributos de agrupamento ocorrentes no primeiro registro de LARQ.

4 - ULTLIGADO ← (próxima posição a ser preenchida na LISTA DE LIGADOS) - 1.

5 - PRIMLIGADO ← ULTLIGADO + 1.

6 - Para cada registro de LARQ, fazer:

6.1 - Se os valores dos atributos de agrupamento ocorrentes neste registro de LARQ são diferentes dos mantidos em ANTERIOR, fazer:

6.1.1 - Adicionar à LISTA DE LIGANTES um registro contendo os valores de:
- ANTERIOR
- PRIMLIGADO
- ULTLIGADO

6.1.2 - ANTERIOR ← valores dos atributos de agrupamento ocorrentes neste registro de LARQ.

6.1.3 - PRIMLIGADO ← ULTLIGADO + 1.

6.2 - ULTLIGADO ← ULTLIGADO + 1.

6.3 - Formar um registro com os campos desse registro de LARQ que não constituem atributos de agrupamento.

6.4 - Adicionar à LISTA DE LIGADOS o registro formado.

7 - Adicionar à LISTA DE LIGANTES um registro contendo os valores de:
- ANTERIOR
- PRIMLIGADO
- ULTLIGADO.

9 - DESAGRUPAR

Algoritmos apresentados:

- DESAGRUPAR1: percorre seqüencialmente um arquivo de associações por item, buscando os ligados de cada associação.

- DESAGRUPAR2: percorre seqüencialmente um arquivo de associações por item ou por idreg, buscando os ligantes e os ligados de cada associação.

- DESAGRUPAR3: percorre seqüencialmente uma lista de ligantes e uma lista de ligados que representam uma tabela ligacional na zona intermediária ou no canal auxiliar.

DESAGRUPAR1 (ASS)

- Condição:

ASS é um arquivo de associações por item cuja chave primária é constituída por todo o ligante.

- Resultado:

Lista de dados classificada segundo a chave primária do arquivo de associações por item.

- Método:

Estruturas de dados utilizadas:

ARQD2 - arquivo de dados contendo ligados.

Passos:

1 - Para cada registro de ASS, fazer:
1.1 - Se o conteúdo da construção sob nome CARD-LIGADOS é diferente de zero, então:
1.1.1 - Para cada registro de inversão mantido na construção sob nome LIGADOS desse registro de ASS, fazer:
1.1.1.1 - Buscar o registro de ARQD2 apontado por este idreg.
1.1.1.2 - Adicionar à lista resultado um registro constituído pela concatenação do conteúdo da construção sob nome ITENS desse registro de ASS com o registro trazido de ARQD2.

DESAGRUPAR2 (ASS,ORD)

- Condição:

ASS é um arquivo de associações por item ou por idreg.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a chave primária do arquivo de associações. Do contrário, resulta uma lista de dados não classificada.

- Método:

Estruturas de dados utilizadas:

ARQD1 - Arquivo de dados contendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.
LII - Lista de pares de idregs.
LIIC - Lista de pares de idregs classificados segundo o primeiro idreg de cada par.

Passos:

1 - Se ORD for uma ordem válida, fazer:
1.1 - Para cada registro de ASS, fazer:
1.1.1 - Se o conteúdo da construção sob nome CARD-LIGADOS é diferente de zero, então:
1.1.1.1 - Buscar o registro de ARQD1 cujo idreg é mantido na construção sob nome LIGANTE desse registro de ASS.
1.1.1.2 - Para cada registro de inversão mantido na construção sob nome LIGADOS desse registro de ASS, fazer:
1.1.1.2.1 - Buscar o registro de ARQD2 apontado por este idreg.
1.1.1.2.2 - Adicionar à lista resultado um registro constituído da concatenação desses dois registros de ARQD1, ARQD2.

2 - Se ORD não for uma ordem válida, fazer:

2.1 - Para cada registro de ASS, fazer:

2.1.1 - Se o conteúdo da construção sob nome CARD-LIGADOS é diferente de zero, então:

2.1.1.1 - Para cada registro de inversão mantido na construção sob nome LIGADOS desse registro de ASS, fazer:

2.1.1.1.1 - Para cada idreg mantido nesse registro de inversão, fazer:

- Adicionar à lista LII o par composto do idreg proveniente da construção sob nome LIGANTE desse registro de ASS, mais o idreg correntemente em análise.

2.2 - Classificar LII segundo o primeiro idreg de cada par (gerando LIIC).

2.3 - Para cada registro de LIIC, fazer:

2.3.1 - Buscar o registro de ARQD1 apontado pelo primeiro idreg desse par.

2.3.2 - Buscar o registro de ARQD2 apontado pelo segundo idreg desse par.

2.3.3 - Adicionar à lista resultado o registro formado pela concatenação do registro trazido de ARQD1 com o registro trazido de ARQD2.

- Observação:

Se ASS for um arquivo de associações por idreg, na formação do registro concatenado deve ser eliminado o primeiro componente dos registros de ARQD2 (que constitui o idreg do ligante ao qual está associado este ligado).

DESAGRUPAR3 (L1, L2)

- Condição:

L1, L2 são listas de dados contendo, respectivamente, dados de ligantes e dados de ligados, que representam uma tabela ligacional mantida na zona intermediária ou no canal auxiliar.

- Resultado:

Lista de dados classificada segundo a mesma ordem que L1 (isto é, segundo a ordem dos ligantes).

- Método:

Passos:

1 - Para cada registro de L1, fazer:
1.1 - Se os campos PRIMLIGADO e ULTLIGADO desse registro contêm valores válidos, então fazer:
1.1.1 - Para cada registro de L2 cuja posição esteja entre PRIMLIGADO e ULTLIGADO (inclusive estes), fazer:
1.1.1.1 - Adicionar à lista resultado o registro formado pela concatenação dos registros correntes de L1 e L2.

10 - SUB, SUB=

Algoritmos apresentados:

- SUB1: percorre seqüencialmente o primeiro operando (uma lista de dados classificada) e, via o arquivo de inversões sobre chave primária do segundo operando, busca os registros de dados do mesmo.

- SUB2: percorre seqüencialmente os dois operandos (duas listas de dados classificadas segundo atributos que não são, necessariamente, chave primária).

- SUB3: percorre seqüencialmente dois arquivos de inversões sobre chave primária, buscando os registros de dados dos dois operandos.

- SUB4: percorre seqüencialmente um arquivo de inversões sobre a chave primária do primeiro operando e uma lista classificada (o segundo operando), buscando os registros correspondentes no arquivo de dados.

- Observação:

As operações SUB e SUB= são resolvidas pelos mesmos algoritmos. A distinção é feita apenas pelo valor do parâmetro PROP, que determina se deve ser testada a situação de subconjunto próprio (PROP=1) ou, apenas, subconjunto (PROP≠1).

SUB1 (L, ARQ, PROP)

- Condição:

L é uma lista de dados. ARQ é um arquivo de dados.

- Resultado:

Se PROP=1, então resulta o valor booleano VERDADEIRO, se L for subconjunto próprio de ARQ, ou o valor booleano FALSO, em caso contrário. Se PROP≠1, resulta o valor booleano VERDADEIRO, se L for subconjunto (não próprio) de ARQ, ou o valor booleano FALSO, se isto não ocorrer.

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre a chave primária de ARQ.

Passos:

1 - Se (PROP=1 e a cardinalidade de L é maior ou igual à cardinalidade de ARQ) ou se (PROP≠1 e a cardinalidade de L é maior que a cardinalidade de ARQ) então, finalizar devolvendo resultado FALSO. Senão, seguir.

2 - Para cada registro de L, enquanto não tiver sido devolvido resultado algum, fazer:

2.1 - Buscar em INV um registro de inversão cuja construção sob nome ITENS tenha mesmo conteúdo que o registro corrente de L, nos campos referentes à chave primária de ARQ.
/* Se L estiver classificada segundo a chave primária de ARQ, a busca será seqüencial. Do contrário, será usada a primitiva que busca registro de inversão com determinado valor */

2.2 - Se não for encontrado em INV um registro nessas condições, finalizar devolvendo o resultado FALSO.

2.3 - Se for encontrado em INV o registro procurado, fazer:

2.3.1 - Buscar o registro de ARQ apontado pelo idreg mantido nesse registro de inversão.

2.3.2 - Verificar se este registro tem conteúdo exatamente igual ao conteúdo do registro corrente de L, em todos os seus campos.

Sim → Seguir.

Não → Finalizar devolvendo o resultado FALSO.

3 - Se todos os registros de L foram encontrados em ARQ, finalizar devolvendo o resultado VERDADEIRO.

SUB2 (L1, L2, PROP)

- Condição:

L1, L2 são arquivos/listas de dados.

- Resultado:

Se PROP=1, então resulta o valor booleano VERDADEIRO se L1 for subconjunto próprio de L2, ou o valor booleano FALSO, em caso contrário.

Se PROP≠1, resulta o valor booleano VERDADEIRO, se L1 for subconjunto (não próprio) de L2, ou o valor booleano FALSO, se isto não ocorrer.

- Método:

Estruturas de dados utilizadas:

L1C - é o próprio arquivo/lista de dados L1, se já estiver classificado segundo a ordem desejada, ou uma lista contendo L1 classificado.

L2C - é o próprio arquivo/lista de dados L2, se já estiver classificado segundo a ordem desejada, ou uma lista contendo L2 classificado.

Passos:

1 - Se (PROP=1 e a cardinalidade de L1 é maior ou igual à cardinalidade de L2) ou se (PROP≠1 e a cardinalidade de L1 é maior que a cardinalidade de L2), então finalizar devolvendo o resultado FALSO. Senão, seguir.

2 - Verificar se os arquivos/listas L1 e L2 já estão classificados segundo alguma ordem.

3 - Se ambos já estão classificados sob ordens diferentes, então classificar o arquivo/lista de me-

nor cardinalidade segundo a ordem de classificação do outro (gerando L1C ou L2C, conforme o caso).

4 - Se apenas um dos arquivos/listas está classificado, então classificar o outro arquivo/lista segundo esta ordem de classificação (gerando L1C ou L2C, conforme o caso).

5 - Se nenhum dos arquivos/listas está classificado, então classificar ambos segundo a chave primária de L2 (gerando L1c ou L2C).

6 - ANTERIOR ← valores dos atributos de ordenação ocorrentes no primeiro registro de L1C.

7 - P ← posição do primeiro registro da lista L2C.

8 - Para cada registro de L1C, enquanto não tiver sido devolvido resultado algum fazer:

8.1 - Se ANTERIOR ≠ valores dos atributos de ordenação ocorrentes neste registro de L1C, então:

8.1.1 - ANTERIOR ← valores dos atributos de ordenação ocorrentes neste registro de L1C.

8.1.2 - P ← posição do registro corrente da lista L2C.

8.2 - Buscar em L2C, a partir do registro apontado por P, um registro cujos valores dos atributos de ordenação sejam iguais aos valores dos atributos correspondentes, no registro corrente de L1C.

8.3 - Se não for encontrado, então finalizar devolvendo o resultado FALSO.

8.4 - Do contrário, fazer:

8.4.1 - ENCONTRADO ← Falso.

8.4.2 - Para cada registro de L2C cujos valores dos atributos de ordenação sejam iguais aos cor-

respondentes, no registro corrente de L1C, enquanto ENCONTRADO tiver valor Falso, fazer:

8.4.2.1 - Se os registros correntes de L1C e L2C são iguais, em todos os campos, então:

8.4.2.1.1 - ENCONTRADO ← Verdadeiro.

8.4.3 - Se ENCONTRADO tiver valor falso então finalizar devolvendo o resultado FALSO.

9 - Se todos os registros de L1C foram encontrados em L2C, então finalizar devolvendo o resultado VERDADEIRO.

SUB3 (ARQ1, ARQ2, PROP)

- Condição:

ARQ1, ARQ2 são arquivos de dados com mesma chave primária.

- Resultado:

Se PROP=1 então resulta o valor booleano VERDADEIRO, se ARQ1 é um subconjunto próprio de ARQ2, ou o valor booleano FALSO, caso isto não se verifique.

Se PROP≠1, resulta o valor booleano VERDADEIRO se ARQ1 for subconjunto (não próprio) de ARQ, ou o valor booleano FALSO se isto não ocorrer.

- Método:

Estruturas de dados utilizadas:

INV1 - Arquivo de inversões sobre a chave primária de ARQ1.
INV2 - Arquivo de inversões sobre a chave primária de ARQ2.

Passos:

1 - Se (PROP=1 e a cardinalidade de ARQ1 é maior ou igual à cardinalidade de ARQ2) ou se (PROP≠1 e a cardinalidade de L é maior que a cardinalidade de ARQ) então finalizar devolvendo resultado FALSO. Senão, seguir.

2 - Para cada registro de INV1, fazer:

2.1 - Buscar em INV2 um registro cuja construção sob nome ITENS tenha conteúdo igual ao da correspondente construção ITENS do registro corrente de INV1.

2.2 - Se não for encontrado em INV2 um registro nessas condições, então finalizar devolvendo o resultado FALSO. Do contrário, seguir.

2.3 - Buscar o registro de ARQ1 apontado pelo idreg mantido no registro corrente de INV1.

2.4 - Buscar o registro de ARQ2 apontado pelo idreg mantido no registro corrente de INV2.

2.5 - Verificar se o conteúdo desses registros buscados de ARQ1, ARQ2 (respectivamente) é igual, em todos os campos.
Sim → Seguir.
Não → Finalizar devolvendo o resultado FALSO.

3 - Se todos os registros de ARQ1 foram encontrados em ARQ2, então finalizar devolvendo o resultado VERDADEIRO.

SUB4 (ARQ, L, PROP)

- Condição:

ARQ é um arquivo de dados. L é uma lista de dados.

- Resultado:

Se PROP=1, então resulta o valor booleano VERDADEIRO, se ARQ for subconjunto próprio de L, ou o valor booleano FALSO, em caso contrário.

Se PROP≠1, então resulta o valor booleano VERDADEIRO, se ARQ for subconjunto (não próprio) de L, ou o valor booleano FALSO, se isto não ocorrer.

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre a chave primária de ARQ.
LC - É a própria lista L, caso a mesma já esteja classificada, ou uma lista contendo a lista L classificada.

Passos:

1 - Se (PROP=1 e a cardinalidade de ARQ é maior ou igual à cardinalidade de L) ou se (PROP≠1 e a cardinalidade de ARQ é maior que a cardinalidade de L) então finalizar devolvendo resultado FALSO. Senão, seguir.

2 - Verificar se L está classificada segundo a chave primária de ARQ.
Sim → Seguir.
Não → Classificar L segundo esta ordem (gerando LC).

3 - Para cada registro de INV, fazer:

3.1 - Buscar em LC um registro cujo conteúdo seja igual ao da construção sob nome ITENS do registro corrente de INV, nos respectivos campos que constituem a chave primária de ARQ.

3.2 - Se não for encontrado, em LC, um registro nessas condições, finalizar devolvendo o resultado FALSO. Do contrário, seguir.

3.3 - Buscar o registro de ARQ apontado pelo idreg mantido no registro corrente de INV.

3.4 - Verificar se este registro de ARQ é igual, em todos os campos, ao registro corrente de LC.
Sim → Seguir.
Não → Finalizar devolvendo o resultado FALSO.

4 - Se todos os registros de ARQ foram encontrados em LC, então finalizar devolvendo o resultado VERDADEIRO.

11 - ELEM

Algoritmos apresentados:

- ELEM1: realiza um acesso a um arquivo de dados via um arquivo de inversões sobre sua chave primária.

- ELEM2: realiza pesquisa binária sobre uma lista de dados classificada.

- ELEM3: percorre seqüencialmente um arquivo/lista de dados.

- ELEM4: busca um registro de associação por item com determinado valor de chave e, a partir do mesmo, busca os registros de dados de seu ligante e ligados.

- ELEM5: busca um registro de inversão com valor de chave dado pelo ligante de um registro de associação e,a partir do idreg mantido no mesmo, chega até um registro de associação por idreg, buscando os registros de dados de seus ligados.

- ELEM6: realiza pesquisa binária sobre uma lista de ligantes mantida na zona intermediária ou canal auxiliar.

- ELEM7: percorre seqüencialmente uma lista de ligantes mantida na zona intermediária ou canal auxiliar.

ELEM1 (REGISTRO, ARQ)

- Condição:

ARQ1 é um arquivo de dados. REGISTRO é um registro de dados no topo da pilha.

- Resultado:

O valor booleano VERDADEIRO, se REGISTRO pertencer a ARQ, ou o valor booleano FALSO, se REGISTRO não pertencer a ARQ.

- Método:

Estruturas de dados utilizadas:

INV - arquivo de inversões sobre a chave primária de ARQ.

Passos:

1 - Buscar em INV o registro de inversão cuja construção sob nome ITENS tenha conteúdo igual ao conteúdo de REGISTRO, nos respectivos campos que constituem a chave primária de ARQ.
/* Usando a primitiva que busca registro de inversão com determinado valor de chave */

2 - Se não for encontrado em INV um registro nessas condições, então finalizar devolvendo o resultado FALSO.

3 - Do contrário, fazer:

3.1 - Buscar em ARQ o registro apontado pelo idreg mantido no registro corrente de INV.

3.2 - Verificar se todos os campos desse registro de ARQ têm conteúdo igual aos respectivos campos de REGISTRO.
Sim → finalizar devolvendo o resultado VERDADEIRO.
Não → finalizar devolvendo o resultado FALSO.

ELEM2 (REGISTRO, L)

- Condição:

L é uma lista de dados classificada segundo sua chave primária. REGISTRO é um registro de dados no topo da pilha.

- Resultado:

O valor booleano VERDADEIRO, se REGISTRO pertencer a L, ou o valor booleano FALSO, se REGISTRO não pertencer a L.

- Método:

Passos:

1 - Aplicar um algoritmo de pesquisa binária para buscar o registro de L que tem conteúdo igual ao de REGISTRO, nos respectivos campos que constituem a chave primária de L.

2 - Se encontrado, finalizar devolvendo o resultado VERDADEIRO. Do contrário, finalizar devolvendo o resultado FALSO.

ELEM3 (REGISTRO, L)

- Condição:

L é um arquivo/lista de dados. REGISTRO é um registro de dados no topo da pilha.

- Resultado:

O valor booleano VERDADEIRO, se REGISTRO pertencer a L, ou o valor booleano FALSO, se REGISTRO não pertencer a L.

- Método:

Passos:

1 - ENCONTRADO ← FALSO.

2 - Para cada registro de L, enquanto ENCONTRADO tiver valor FALSO, fazer:

2.1 - Verificar se o registro corrente de L é igual a REGISTRO, em todos os campos.
Sim → ENCONTRADO ← VERDADEIRO.
Não → seguir.

3 - Se ENCONTRADO tiver valor FALSO, então finalizar devolvendo resultado FALSO.

4 - Do contrário, finalizar devolvendo resultado VERDADEIRO.

ELEM4 (REGISTRO, ASS)

- Condição:

ASS é um arquivo de associações por item. REGISTRO é um registro de associação no topo da pilha.

- Resultado:

O valor booleano VERDADEIRO, se REGISTRO for encontrado em ASS, ou o valor booleano FALSO, se REGISTRO não pertencer a ASS.

- Método:

Estruturas de dados utilizadas:

ARQD1 - Arquivo de dados cotendo ligantes.
ARQD2 - Arquivo de dados contendo ligados.

Passos:

1 - Buscar em ASS o registro de associação cuja construção sob nome ITENS tenha mesmo conteúdo que o ligante de REGISTRO, nos respectivos campos que constituem a chave primária de ASS.

2 - Se não for encontrado em ASS um registro nessas condições, então finalizar devolvendo o resultado FALSO. Do contrário, fazer:

2.1 - Se a chave primária desse arquivo de associações é formada por todo o ligante, então seguir para o próximo passo. Do contrário,fazer:

2.1.1 - Buscar o registro de ARQD1 cujo idreg é o mantido na construção sob nome LIGANTE desse registro de associação.

2.1.2 - Verificar se o conteúdo desse registro de ARQD1 é exatamente o mesmo que o conteúdo do ligante de REGISTRO (em todos os campos).
Não → finalizar devolvendo o resultado FALSO.

Sim → seguir.

2.2 - Verificar se o conteúdo da construção sob nome CARD-LIGADOS desse registro de associação é igual à cardinalidade da tabela de ligados de REGISTRO.
Não → finalizar devolvendo o resultado FALSO.
Sim → seguir.

2.3 - Verificar se o conteúdo da construção sob nome CARD-LIGADOS tem valor zero.
/* Neste caso, também será zero a cardinalidade da tabela de ligados de REGISTRO */
Não → seguir.
Sim → finalizar devolvendo o resultado VERDADEIRO.

2.4 - Para cada registro da tabela de ligados de REGISTRO, fazer:

2.4.1 - Buscar, na construção sob nome LIGADOS do registro corrente de ASS, um registro de inversão cuja construção sob nome ITENS tenha conteúdo igual ao do registro em exame na tabela de ligados de REGISTRO, nos respectivos campos.
/* A busca será feita sem chamada de primitiva específica */

2.4.2 - Se encontrado, fazer:

2.4.2.1 - Buscar o registro de ARQD2 apontado pelo idreg mantido nesse registro de inversão.

2.4.2.2 - Verificar se este registro de ARQD2 tem, em todos os seus campos, conteúdo igual ao do registro corrente da tabela de ligados de REGISTRO.
Sim → seguir.
Não → finalizar devolvendo o resultado FALSO.

2.4.3 - Se não encontrado, então finalizar devolvendo o resultado FALSO.

2.5 - Se todos os registros da tabela de ligados de

REGISTRO foram encontrados como ligados desse registro de ASS, então finalizar devolvendo o resultado VERDADEIRO.

ELEM5 (REGISTRO,ASS)

- Condição:

ASS é um arquivo de associações por idreg. REGISTRO é um registro de associação no topo da pilha.

- Resultado:

O valor booleano VERDADEIRO, se REGISTRO for encontrado em ASS, ou o valor bolleano FALSO, se REGISTRO não pertencer a ASS.

- Método:

Estruturas de dados utilizadas:

ARQD1 - arquivo de dados contendo ligantes.
ARQD2 - arquivo de dados contendo ligados.
INV - arquivo de inversões sobre a chave primária de ARQD1.

Passos:

1 - Buscar em INV um registro de inversão cuja construção sob nome ITENS tenha mesmo conteúdo que o ligante de REGISTRO, nos campos correspondentes.

2 - Se não for encontrado em INV um registro nessas condições, então finalizar devolvendo o resultado FALSO.

3 - Do contrário, fazer:
3.1 - Buscar o registro de ARQD1 apontado pelo idreg mantido no registro corrente de INV.
3.2 - Se o conteúdo desse registro de ARQD1 é exatamente igual ao conteúdo do ligante de REGISTRO (em todos os seus campos), então seguir. Do contrário, finalizar devolvendo o resultado FALSO.
3.3 - Buscar em ASS um registro de associação cuja

construção sob nome LIGANTE tenha como conteúdo o idreg do registro acessado em ARQD1.

3.4 - Se for encontrado em ASS um registro nessas condições, então fazer:

3.4.1 - Executar os passos 2.2 a 2.5 do algoritmo ELEM4.

/* Testa se os ligados são os mesmos, para os dois registros de associação em comparação */

ELEM6 (REGISTRO, L1, L2)

- Condição:

L1, L2 são listas de dados que representam uma tabela ligacional mantida na zona intermediária ou no canal auxiliar, contendo, respectivamente, dados de ligantes e dados de ligados. L1 está classificada segundo a chave primária dos ligantes. REGISTRO é um registro de associação no topo da pilha.

- Resultado:

O valor booleano VERDADEIRO, se REGISTRO pertencer à tabela ligacional representada por L1, L2, ou o valor booleano FALSO, em caso contrário.

- Método:

Estruturas de dados utilizadas:

LC - é a própria tabela de ligados de REGISTRO, se já estiver classificada, ou uma lista de dados contendo esta tabela classificada.

Passos:

1 - Buscar em L1, aplicando um algoritmo de pesquisa binária, o registro de dados de ligante cujo conteúdo seja completamente igual ao conteúdo do ligante de REGISTRO, em todos os campos que constituem atributos do ligante.

2 - Se não for encontrado, finalizar devolvendo o resultado FALSO..

3 - Do contrário, fazer:

3.1 - Verificar se a tabela de ligados de REGISTRO e a tabela de ligados desse ligante de L1 têm cardinalidades diferentes.

Sim → finalizar devolvendo o resultado FALSO.

Não → seguir.

3.2 - Verificar se ambas as listas de ligados têm cardinalidade zero.

Sim → finalizar devolvendo o resultado VERDADEIRO.

Não → seguir.

3.3 - Verificar se a tabela de ligados de REGISTRO está classificada segundo a mesma ordem que os ligados de uma tabela de ligados de uma ligação mantida em L1, L2.

Sim → seguir.

Não → classificar a tabela de ligados de REGISTRO segundo a ordem de classificação de uma tabela de ligados de uma ligação mantida em L1, L2 (gerando LC).

3.4 - Para cada registro de L2 contendo um ligado do ligante mantido no registro corrente de L1, fazer:

3.4.1 - Buscar em LC um registro cujo conteúdo seja exatamente igual ao conteúdo do registro corrente de L2.

3.4.2 - Se não for encontrado em LC um registro nessas condições, finalizar devolvendo o resulta do FALSO.

3.5 - Se todos os registros de L2 associados ao registro corrente de L1 foram encontrados em LC, então finalizar devolvendo o resultado VERDADEIRO.

ELEM7 (REGISTRO, L1, L2)

- Condição:

L1, L2 são listas de dados que representam uma tabela ligacional mantida na zona intermediária ou no canal auxiliar, contendo, respectivamente, dados de ligantes e dados de ligados. REGISTRO é um registro de associação no topo da pilha.

- Resultado:

O valor booleano VERDADEIRO, se REGISTRO pertencer à tabela ligacional representada por L1, L2, ou o valor booleano FALSO, em caso contrário.

- Método:

Estruturas de dados utilizadas:

LC - é a própria tabela de ligados de REGISTRO, se já estiver classificada, ou uma lista de dados contendo esta tabela classificada.

Passos:

1 - Buscar em L1 (seqüencialmente) o registro de dados de ligante cujo conteúdo seja completamente igual ao conteúdo do ligante de REGISTRO, em todos os campos que constituem atributos do ligante.

2 - Executar os passos 2 e 3 do algoritmo ELEM6.
/* Testa se os ligados são os mesmos, para os dois registros de associação em comparação */

12 - JUNT

Algoritmos apresentados:

- JUNT1A: percorre seqüencialmente dois arquivos de inversões para atingir os registros de dados a serem juntados.

- JUNT1B: idem ao JUNT1A porém admite duplicatas das chaves comparadas, nos arquivos de inversões (construção sob nome ITENS dos registros de inversão).

- JUNT2A: percorre seqüencialmente o primeiro operando (uma lista de dados), fazendo acesso a um arquivo de inversões sobre o segundo operando, a partir de cada registro da lista (seqüencialmente ou não, conforme a lista de dados esteja ou não classificada).

- JUNT2B: idem ao JUNT2A, porém admite duplicatas da chave comparada, nos registros de dados e de inversão.

- JUNT3: percorre seqüencialmente duas listas de dados classificadas segundo atributos que não são, necessariamente, chave primária.

- JUNT4: percorre seqüencialmente um arquivo de associações por item, buscando os registros de dados candidatos à junção.

- JUNT5: realiza uma operação DESAGRUPAR (usando o algoritmo DESAGRUPAR2).

JUNT1A (ARQ1, ARQ2, ORD)

- Condição:

ARQ1 e ARQ2 são arquivos de dados para os quais existem arquivos de inversões cuja chave primária é constituída exatamente por um conjunto dos atributos de junção.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD dos atributos de junção. Do contrário, resulta uma lista de dados não classificada.

- Método:

Estruturas de dados utilizadas:

INV1 - arquivo de inversões sobre ARQ1.
INV2 - arquivo de inversões sobre ARQ2.
LI - lista de pares de idregs.
LIC - lista de pares de idregs classificada de acordo com o primeiro idreg de cada par.

Passos:

1 - Se ORD for uma ordem válida, fazer:
1.1 - Para cada registro de INV1, fazer:
1.1.1 - Buscar em INV2 um registro de inversão cuja construção sob nome ITENS tenha mesmo conteúdo que a construção ITENS do registro corrente de INV1.
1.1.2 - Se encontrado, fazer:
1.1.2.1 - Para cada idreg mantido no registro corrente de INV1, fazer:
1.1.2.1.1 - Buscar o registro de ARQ1 apontado pelo mesmo.
1.1.2.1.2 - Para cada idreg mantido no registro corrente de INV2, fazer:

- Buscar o registro de ARQ2 apontado pelo mesmo.
- Verificar se os registros correntes de ARQ1 e ARQ2 satisfazem a toda a expressão de junção.

Sim → - Formar um registro contendo os valores dos registros correntes de ARQ1 e ARQ2, de acordo com a configuração de um registro resultante da junção.

- Adicionar esse registro à lista resultado.

Não → seguir.

2 - Do contrário, fazer:
2.1 - Para cada registro de INV1, fazer:
2.1.1 - Buscar em INV2 um registro de inversão cuja construção sob nome ITENS tenha mesmo conteúdo que a construção ITENS do registro corrente de INV1.
2.1.2 - Se encontrado, fazer:
2.1.2.1 - Para cada idreg mantido no registro corrente de INV1, fazer:
2.1.2.1.1 - Para cada idreg mantido no registro corrente de INV2, fazer:
- Adicionar à lista LI um registro contendo o par de idregs correntes (de INV1 e INV2, respectivamente).

2.2 - Classificar a lista Li de acordo com o primeiro idreg de cada par (gerando LIC).
2.3 - Para cada registro de LIC, fazer:
2.3.1 - Buscar o registro de ARQ1 apontado pelo primeiro idreg do par mantido no registro corrente de LIC.
2.3.2 - Buscar o registro de ARQ2 apontado pelo segundo idreg desse par.
2.3.3 - Se os registros trazidos de ARQ1 e ARQ2 satis

fazem a toda a expressão de junção, então:

2.3.3.1 - Formar um registro contendo valores de ambos, de acordo com a configuração de um registro resultante da junção.

2.3.3.2 - Adicionar o mesmo à lista resultado.

JUNT1B (ARQ1, ARQ2, ORD)

- Condição:

ARQ1 e ARQ2 são arquivos de dados para os quais existem arquivos de inversões cuja chave primária inclui como componentes um conjunto de atributos de junção.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD dos atributos de junção. Do contrário, resulta uma lista de dados não classificada.

- Método:

Estruturas de dados utilizadas:

INV1 - arquivo de inversões sobre ARQ1.
INV2 - arquivo de inversões sobre ARQ2.
LI - lista de pares de idregs.
LIC - lista de pares de idregs classificada de acordo com o primeiro idreg de cada par.

Passos:

1 - Se ORD for uma ordem válida, fazer:
1.1 - ANTERIOR ← valores inválidos para os atributos de junção.
1.2 - Para cada registro de INV1, fazer:
1.2.1 - Se ANTERIOR ≠ valores dos atributos de junção ocorrentes nesse registro de INV1, então:
1.2.1.1 - ANTERIOR ← valores dos atributos de junção ocorrentes nesse registro de INV1.
1.2.1.2 - P ← posição do registro corrente de INV2 (ou posição do primeiro registro de INV2, se esta for a primeira execução deste comando).

1.2.2 - Buscar em INV2, a partir do registro apontado por P, um registro de inversão cujos valores dos atributos de junção ocorrentes na construção sob nome ITENS sejam iguais aos mantidos em ANTERIOR.

1.2.3 - Se encontrado, fazer:

1.2.3.1 - P ← posição do registro corrente de INV2.

1.2.3.2 - Para cada idreg mantido no registro corrente de INV1, fazer:

1.2.3.2.1 - Buscar o registro de ARQ1 apontado pelo mesmo.

1.2.3.2.2 - Para cada registro de INV2 cujos valores dos atributos de junção ocorrentes na construção sob nome ITENS sejam iguais aos mantidos em ANTERIOR, fazer:

- Para cada idreg mantido nesse registro de inversão, fazer:
  - Buscar o registro de ARQ2 apontado pelo mesmo.
  - Verificar se toda a expressão de junção é satisfeita pelo par de registros de dados em análise.

    Sim → - Formar um registro contendo valores dos registros correntes de ARQ1 e ARQ2, conforme a configuração de um registro resultante da junção.
    - Adicionar o mesmo à lista resultado.

    Não → seguir.

2 - Se ORD não for uma ordem válida, fazer:

2.1 - ANTERIOR ← valores inválidos para os atributos de junção.

2.2 - Para cada registro de INV1, fazer:

2.2.1 - Se ANTERIOR ≠ valores dos atributos de junção ocorrentes nesse registro de INV1, então:

2.2.1.1 - ANTERIOR ← valores dos atributos de junção ocorrentes neste registro de INV1.

2.2.1.2 - P ← posição do registro corrente de INV2 (ou posição do primeiro registro de INV2, se esta for a primeira execução deste comando).

2.2.2 - Buscar em INV2, a partir do registro apontado por P, um registro de inversão cujos valores dos atributos de junção ocorrentes na construção sob nome ITENS sejam iguais aos mantidos em ANTERIOR.

2.2.3 - Se encontrado, fazer:

2.2.3.1 - P ← posição do registro corrente de INV2.

2.2.3.2 - Para cada idreg mantido no registro corrente de INV1, fazer:

2.2.3.2.1 - Para cada registro de INV2 cujos valores dos atributos de junção ocorrentes na construção sob nome ITENS semam iguais aos mantidos em ANTERIOR, fazer:

- Para cada idreg mantido nesse registro de INV2, fazer:
  - Adicionar à lista LI o par constituído dos idregs correntemente em análise.

2.3 - Classificar a lista LI segundo o primeiro idreg de cada par (gerando LIC).

2.4 - Para cada registro de LIC, fazer:

2.4.1 - Buscar o registro de ARQ1 apontado pelo primeiro idreg do par de idregs mantido nesse registro de LIC.

2.4.2 - Buscar o registro de ARQ2 apontado pelo segundo idreg mantido nesse par.

2.4.3 - Se os registros trazidos de ARQ1 e ARQ2 satisfazem a toda a expressão de junção, então:

2.4.3.1 - Formar um registro contendo valores desses dois registros de dados, de acordo com a

configuração de um registro resultante da junção.

2.4.3.2 - Adicionar o mesmo à lista resultado.

JUNT2A (ARQ1, ARQ2, QUEM)

- Condição:

ARQ1 é um arquivo de dados para o qual existe um arquivo de inversões cuja chave primária é constituída exatamente por um conjunto dos atributos de junção. ARQ2 é um arquivo/lista de dados.

- Resultado:

Lista de dados classificada segundo ARQ2.

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ1.

Passos:

- Observação:

Se ARQ2 está classificado segundo o arquivo de inversões, o arquivo de inversões é percorrido seqüencialmente. Do contrário, deve ser usada a primitiva que busca registro de inversão com determinado valor.

1 - Para cada registro de ARQ2, fazer
1.1 - Buscar o registro de INV cuja construção sob nome ITENS tenha conteúdo igual ao dos respectivos campos do registro corrente de ARQ2.
1.2 - Se encontrado, fazer:
1.2.1 - Para cada idreg mantido no registro corrente de INV, fazer:
1.2.1.1 - Buscar o registro de ARQ1 apontado por este idreg.
1.2.1.2 - Se toda a expressão de junção é satisfeita pelo par de registros de dados em análise, então:

1.2.1.2.1 - Formar um registro contendo valores dos registros correntes de ARQ1 e ARQ2, de acordo com a configuração* de um registro resultante da junção.

1.2.1.2.2 - Adicionar esse registro à lista resultado.

---

* O valor do parâmetro QUEM dirá qual registro (se o de ARQ1 ou o de ARQ2) irá fornecer a primeira parte do registro resultante; conseqüentemente, o outro fornecerá a segunda parte.

JUNT2B (ARQ1, ARQ2, QUEM)

- Condição:

ARQ1 é um arquivo de dados para o qual existe um arquivo de inversões cuja chave primária inclui como componentes um conjunto dos atributos de junção. ARQ2 é um arquivo/lista de dados.

- Resultado:

Lista de dados classificada segundo ARQ2.

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ1.

Passos:

- Observação:

Se ARQ2 está classificado segundo o arquivo de inversões, o arquivo de inversões é percorrido seqüencialmente. Do contrário, deve ser usada a primitiva que busca registro de inversão com determinado valor.

1 - Para cada registro de ARQ2, fazer:

1.1 - Buscar em INV um registro de inversão cujos valores dos atributos de junção ocorrentes na construção sob nome ITENS sejam iguais aos atributos correspondentes, no registro corrente de ARQ2.

1.2 - Se encontrado, fazer:

1.2.1 - Para cada registro de INV, enquanto os valores dos atritubos de junção ocorrentes na construção sob nome ITENS sejam iguais aos atributos correspondentes, no registro corrente de ARQ2, fazer:

1.2.1.1 - Para cada idreg mantido nesse registro de inversão, fazer:

1.2.1.1.1 - Buscar o registro de ARQ1 apontado pelo mesmo.

1.2.1.1.2 - Se toda a expressão de junção é satisfeita pelos registros correntes de ARQ2 e ARQ1, então:

- Formar um registro contendo valores dos registros correntes de ARQ1 e ARQ2, conforme a configuração* de um registro resultante da junção.
- Adicionar o mesmo à lista resultado.

---

* Dada pelo parâmetro QUEM (ver algoritmo JUNT2A).

JUNT3 (ARQ1, ARQ2, ORD)

- Condição:

ARQ1 e ARQ2 são arquivos/listas de dados.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD. Do contrário, resulta uma lista de dados classificada:

I) Segundo a ordem em que os atributos de junção aparecem nos registros de ARQ1, se nem ARQ1 nem ARQ2 estiverem classificados segundo alguma ordem dos atributos de junção;

II) Segundo a ordem de classificação da lista de maior cardinalidade, se ambos ARQ1, ARQ2 já estiverem classificados sob diferentes ordens dos atributos de junção.

- Método:

Estruturas de dados utilizadas:

LARQ1 - É o próprio ARQ1 (se já estiver classificado) ou uma lista de dados contendo ARQ1 classificado.

LARQ2 - É o próprio ARQ2 (se já estiver classificado) ou uma lista contendo ARQ2 classificado.

Passos:

1 - Se ORD for uma ordem válida, então:

1.1 - Verificar se ARQ1 está classificado segundo a ordem ORD dos atributos de junção.
Sim → seguir.
Não → classificar ARQ1 segundo a ordem ORD dos atributos de junção (gerando LARQ1).

1.2 - Verificar se ARA2 está classificado segundo a ordem ORD dos atributos de junção.

Sim → seguir.

Não → classificar ARQ2 segundo a ordem ORD dos atributos de junção (gerando LARQ2).

2 - Se ORD não for uma ordem válida, então:

2.1 - Verificar se ambos ARQ1, ARQ2 já estão classificados por diferentes ordens dos atributos de junção.

Sim → classificar o arquivo de menor cardinalidade, de acordo com a ordem de classificação do outro (gerando LARQ1, se o arquivo a classificar for ARQ1, ou LARQ2, caso contrário).

Não → classificar ambos ARQ1, ARQ2 segundo a ordem em que os atributos de junção aparecem nas tuplas de ARQ1.

3 - ANTERIOR ← valores inválidos para os atributos de junção.

4 - Para cada registro de LARQ1, razer:

4.1 - Se ANTERIOR ≠ valores dos atributos de junção ocorrentes nesse registro de LARQ1, fazer:

4.1.1 - ANTERIOR ← valores dos atributos de junção ocorrentes nesse registro de INV1.

4.1.2 - P ← posição do registro corrente de LARQ2 (ou posição do primeiro registro de LARQ2, se esta for a primeira execução deste comando).

4.2 - Buscar em LARQ2, a partir do registro apontado por P, um registro cujos valores dos atributos de junção sejam iguais aos mantidos em ANTERIOR.

4.3 - Se encontrado, fazer:

4.3.1 - P ← posição do registro corrente de LARQ2.

4.3.2 - Para cada registro de LARQ2, enquanto os valores dos atributos de junção forem iguais aos

mantidos em ANTERIOR, fazer:

4.3.2.1 - Formar um registro, contendo valores dos registros correntes de LARQ1 e LARQ2, conforme a configuração de um registro resultante da junção.

4.3.2.2 - Adicionar o mesmo à lista resultado.

JUNT4 (ARQ1, ARQ2)

- Condição:

ARQ1 e ARQ2 são arquivos de dados sobre os quais existe definido um arquivo de associações por item cuja chave primária é constituída exatamente por um conjunto dos atributos de junção.

- Resultado:

Lista de dados classificada segundo a chave primária do arquivo de associações por item.

- Método:

Estruturas de dados utilizadas:

ASS - Arquivo de associações por item.

Passos:

1 - Para cada registro de ASS, fazer:
1.1 - Buscar o registro de ARQ1 cujo idreg é mantido na construção sob nome LIGANTE desse registro de ASS.
1.2 - Para cada registro de inversão mantido na construção sob nome LIGADOS desse registro de ASS, fazer:
1.2.1 - Buscar o registro de ARQ2 apontado pelo idreg mantido nesse registro de inversão.
1.2.2 - Se o restante da expressão de junção é satisfeito por este par de registro de ARQ1, ARQ2, então:
1.2.2.1 - Formar um registro contendo os valores dos registros correntes de ARQ1 e ARQ2, de acordo com a configuração de um registro resultante da junção.
1.2.2.2 - Adicionar o mesmo à lista resultado.

JUNT5 (ARQ1, ARQ2, ORD)

- Condição:

ARQ1, ARQ2 são arquivos de dados sobre os quais existe definido um arquivo de associações por item ASS. A chave primária do arquivo de associações por item é constituída exatamente pelos atributos de junção.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a chave primária do arquivo de associações. Do contrário, resulta uma lista de dados não classificada.

- Método:

Transformar esta operação em um DESAGRUPAR2 (ASS, ORD).

13 - LIG

Algoritmos apresentados:

- LIG1A: percorre seqüencialmente dois arquivos de inversões para atingir os registros de dados a serem ligados.

- LIG1B: idem ao LIG1A porém admite duplicatas da chave comparada, nos arquivos de inversões (construção sob nome ITENS dos registros de inversão).

- LIG2A: percorre seqüencialmente um arquivo de inversões, a partir do qual serão buscados os ligantes, e uma lista de dados classificada, que fornecerá os ligados.

- LIG2B: idem ao LIG2A porém admite duplicatas da chave comparada, nos registros de dados e de inversão.

- LIG3A: percorre seqüencialmente um arquivo/lista de dados que fornecerá os ligantes e via um arquivo de inversões obtém os ligados.

- LIG3B: idem ao LIG3A, porém admite duplicatas da chave comparada, nos registros de dados e de inversão.

- LIG4: percorre seqüencialmente duas listas de dados classificadas segundo atributos que não são, necessariamente, chave primária.

LIG1A (ARQ1, ARQ2)

- Condição:

ARQ1 e ARQ2 são arquivos de dados para os quais existem arquivos de inversões:

- definidos exatamente sobre o conjunto dos atributos de ligação; ou
- definidos exatamente sobre um subconjunto dos atributos de ligação, que é chave primária em ARQ1 e ARQ2.

- Resultado:

Uma lista de dados de ligantes (LISTA DE LIGANTES), classificada segundo a chave primária dos arquivos de inversões, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante não se encontra classificada sob ordem alguma.

- Método:

Estruturas de dados utilizadas:

INV1 - Arquivo de inversões sobre ARQ1.
INV2 - Arquivo de inversões sobre ARQ2.

Passos:

1 - ULTLIGADO ← (primeira posição a ser preenchida na LISTA DE LIGADOS) - 1.

2 - Para cada registro de INV1, fazer:

2.1 - Buscar em INV2 um registro de inversão cuja construção sob nome ITENS tenha conteúdo igual ao da construção sob nome ITENS do registro corrente de INV1, nos atributos de ligação.
/* Primitiva que busca registro de inversão por conteúdo da construção sob nome ITENS */

2.2 - Se encontrado, fazer:

2.2.1 - Se os arquivos de inversões são definidos exatamente sobre o conjunto de atributos de ligação, então:

2.2.1.1 - PRIMLIGADO ← ULTLIGADO + 1.

2.2.1.2 - ULTLIGADO ← PRIMLIGADO + (conteúdo da construção sob nome CARD-IDREGS do registro corrente de INV2) - 1.

2.2.1.3 - Buscar o registro de ARQ1 apontado pelo primeiro idreg mantido no registro corrente de INV1.

2.2.1.4 - Adicionar à LISTA DE LIGANTES um registro contendo:
- este registro trazido de ARQ1
- PRIMLIGADO
- ULTLIGADO.

2.2.1.5 - Para cada idreg mantido no registro corrente de INV2, fazer:

2.2.1.5.1 - Buscar o registro de ARQ2 apontado por este idreg.

2.2.1.5.2 - Formar um registro com os valores do registro trazido de ARQ2, de acordo com a configuração de um ligado*.

2.2.1.5.3 - Adicionar à LISTA DE LIGADOS o registro formado.

2.2.1.6 - Para cada idreg restante mantido no registro corrente de INV1, fazer:

2.2.1.6.1 - Buscar o registro de ARQ1 apontado por este idreg.

2.2.1.6.2 - Adicionar à LISTA DE LIGANTES um registro contendo:
- este registro trazido de ARQ1
- PRIMLIGADO
- ULTLIGADO.

---

* A operação LIGAR permite exclusão de atributos de ligação, na formação dos ligados (ver capítulo 2, item 2.4.2).

2.2.2 - Do contrário, fazer:
/* Neste caso, são utilizados arquivos de inversões sobre chave primária */
2.2.2.1 - Buscar o registro de ARQ1 apontado pelo idreg mantido no registro corrente de INV1.
2.2.2.2 - Buscar o registro de ARQ2 apontado pelo idreg mantido no registro corrente de INV2.
2.2.2.3 - Se ambos satisfazem a toda a expressão de ligação, então:
2.2.2.3.1 - ULTLIGADO ← ULTLIGADO + 1.
2.2.2.3.2 - PRIMLIGADO ← ULTLIGADO.
2.2.2.3.3 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro trazido de ARQ1
- PRIMLIGADO
- ULTLIGADO.
2.2.2.3.4 - Formar um registro com os valores do registro trazido de ARQ2, de acordo com a configuração de um ligado.
2.2.2.3.5 - Adicionar à LISTA DE LIGADOS o registro formado.
2.2.2.4 - Do contrário, fazer:
/* A expressão de ligação não é totalmente satisfeita */
2.2.2.4.1 - Adicionar à LISTA DE LIGANTES um registro contendo:
- este registro trazido de ARQ1.
- indicação de tabela de ligados vazia.
2.3 - Se não encontrado, fazer:
2.3.1 - Para cada idreg mantido no registro corrente de INV1, fazer:
2.3.1.1 - Buscar o registro de ARQ1 apontado por este idreg.
2.3.1.2 - Adicionar à LISTA DE LIGANTES um registro contendo:
- este registro trazido de ARQ1.
- indicação de tabela de ligados vazia.

LIG1B (ARQ1, ARQ2)

- Condição:

ARQ1 e ARQ2 são arquivos de dados para os quais existem arquivos de inversões cuja chave primária inclui como "primeiros" componentes os atributos de ligação.

- Resultado:

Uma lista de dados de ligantes (LISTA DE LIGANTES), classificada segundo a ordem dos atributos de ligação nas chaves de inversão, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante se encontra classificada segundo os atributos adicionais existentes na chave de inversão sobre ARQ2, além dos atributos de ligação.

- Método:

Estruturas de dados utilizadas:

INV1 - Arquivo de inversões sobre ARQ1.
INV2 - Arquivo de inversões sobre ARQ2.

Passos:

1 - ANTERIOR ← valores inválidos para os atributos de ligação.

2 - ULTLIGADO ← (primeira posição a ser preenchida na LISTA DE LIGADOS) - 1.

3 - Para cada registro de INV1, fazer:
3.1 - Se ANTERIOR ≠ valores dos atributos de ligação ocorrentes nesse registro de INV1, então:
3.1.1 - ANTERIOR ← valores dos atributos de ligação ocorrentes nesse registro de INV1.
3.1.2 - P ← posição do registro corrente de INV2 (ou posição do primeiro registro de INV2, se

esta for a primeira execução deste comando).

3.1.3 - Buscar em INV2, a partir do registro apontado por P, um registro de inversão cujos valores dos atributos de ligação ocorrentes na construção sob nome ITENS sejam iguais aos mantidos em ANTERIOR.

3.1.4 - Se encontrado, fazer:

3.1.4.1 - P ← posição do registro corrente de INV2.

3.1.4.2 - Buscar o registro de ARQ1 apontado pelo primeiro idreg mantido no registro corrente de INV1.

3.1.4.3 - PRIMLIGADO ← ULTLIGADO + 1.

3.1.4.4 - Para cada registro de INV2, a partir do registro apontado por P, enquanto os valores dos atributos de ligação ocorrentes na construção sob nome ITENS forem iguais aos mantidos em ANTERIOR, fazer:

4.1.4.4.1 - ULTLIGADO ← ULTLIGADO + (conteúdo da construção sob nome CARD-IDREGS desse registro de INV2).

3.1.4.4.2 - Para cada idreg mantido nesse registro de INV2, fazer:

- Buscar o registro de ARQ2 apontado pelo mesmo.
- Formar um registro com os valores do registro trazido de ARQ2, de acordo com a configuração de um ligado.
- Adicionar à LISTA DE LIGADOS o registro formado.

3.1.4.5 - Adicionar à LISTA DE LIGANTES um registro contendo:

- o registro trazido de ARQ1
- PRIMLIGADO
- ULTLIGADO.

3.1.4.6 - Para cada idreg restante mantido no registro corrente de INV1, fazer:

3.1.4.6.1 - Buscar o registro de ARQ1 apontado pelo mesmo.

3.1.4.6.2 - Adicionar à LISTA DE LIGANTES um registro contendo:
- este registro trazido de ARQ1
- PRIMLIGADO
- ULTLIGADO.

3.1.5 - Se não encontrado, então:

3.1.5.1 - Para cada idreg mantido no registro corrente de INV1, fazer:

3.1.5.1.1 - Buscar o registro de ARQ1 apontado pelo mesmo.

3.1.5.1.2 - Adicionar à LISTA DE LIGANTES um registro contendo:
- este registro trazido de ARQ1.
- indicação de tabela de ligados vazia.
/* Valores inválidos para PRIMLIGADO e ULTLIGADO */

3.2 - Do contrário, fazer:
/* ANTERIOR = valores dos atributos de ligação mantidos na construção sob nome ITENS desse registro de inversão */

3.2.1 - Para cada idreg mantido no registro corrente de INV1, fazer:

3.2.1.1 - Buscar o registro de ARQ1 apontado pelo mesmo.

3.2.1.2 - Adicionar à LISTA DE LIGANTES um registro contendo:
- este registro trazido de ARQ1
- PRIMLIGADO
- ULTLIGADO.
/* No caso de tabelas de ligados vazias as variáveis PRIMLIGADO e ULTLIGADO já estarão atualizadas adequadamente! */

LIG2A (ARQ1, ARQ2)

- Condição:

ARQ1 é um arquivo de dados para o qual existe definido um arquivo de inversões:

- exatamente sobre o conjunto dos atributos de ligação; ou
- exatamente sobre um subconjunto dos atributos de ligação, que é chave primária.

ARQ2 é um arquivo/lista de dados.

- Resultado:

Uma lista de dados de ligantes (LISTA DE LIGANTES), classificada segundo a chave primária do arquivo de inversões, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante se encontra classificado segundo os atributos adicionais sob os quais está classificado ARQ2, além daqueles atributos mantidos na chave de inversão sobre ARQ1. (Caso não existam esses "atributos adicionais" na ordem de ARQ2, os grupos de ligados não se encontram classificados sob ordem alguma).

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ1.

LARQ2 - É o próprio ARQ2, se o mesmo já estiver classificado, ou uma lista de dados contendo ARQ2 classificado.

Passos:

1 - Verificar se ARQ2 está classificado segundo a mesma ordem que os atributos de ligação mantidos na chave de inversão.

Sim → seguir.

Não → classificar ARQ2 segundo esta ordem(gerando LARQ2).

2 - ULTLIGADO ← (primeira posição a ser preenchida na LISTA DE LIGADOS) - 1.

3 - Para cada registro de INV, fazer:

3.1 - Buscar um LARQ2 (seqüencialmente, a partir do último registro visitado) um registro cujos valores dos atributos de ligação sejam iguais aos mantidos na construção sob nome ITENS do registro corrente de INV.

3.2 - Se encontrado, fazer:

3.2.1 - Se o arquivo de inversões é definido exatamente sobre o conjunto de atributos de ligação, então:

3.2.1.1 - PRIMLIGADO ← ULTLIGADO + 1.

3.2.1.2 - Para cada registro de LARQ2 cujos valores dos atributos de ligação sejam iguais aos mantidos na construção sob nome ITENS do registro corrente de INV, fazer:

3.2.1.2.1 - Formar um registro com os valores deste registro de LARQ2, de acordo com a configuração de um ligado.

3.2.1.2.2 - Adicionar à LISTA DE LIGADOS o registro formado.

3.2.1.2.3 - ULTLIGADO ← ULTLIGADO + 1.

3.2.1.3 - Para cada idreg mantido no registro corrente de INV, fazer:

3.2.1.3.1 - Buscar o registro de ARQ1 apontado pelo mesmo.

3.2.1.3.2 - Adicionar à LISTA DE LIGANTES um registro contendo:

- este registro trazido de ARQ1
- PRIMLIGADO
- ULTLIGADO.

3.2.2 - Do contrário fazer:
/* Neste caso, é utilizado um arquivo de inversões sobre chave primária */
3.2.2.1 - Buscar o registro de ARQ1 apontado pelo idreg mantido no registro corrente de INV.
3.2.2.2 - Se o registro trazido de ARQ1 e o registro em análise em LARQ2 satisfazem a toda a expressão de ligação, então:
3.2.2.2.1 - ULTLIGADO ← ULTLIGADO + 1.
3.2.2.2.2 - PRIMLIGADO ← ULTLIGADO.
3.2.2.2.3 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro trazido de ARQ1
- PRIMLIGADO
- ULTLIGADO.
3.2.2.2.4 - Formar um registro com os valores deste registro de LARQ2, de acordo com a configuração de um ligado.
3.2.2.2.5 - Adicionar à LISTA DE LIGADOS o registro formado.
3.2.2.3 - Do contrário:
/* A expressão de ligação não é totalmente satisfeita */
3.2.2.3.1 - Adicionar à LISTA DE LIGANTES um registro contendo:
- este registro trazido de ARQ1.
- indicação de tabela de ligados vazia.
3.3 - Se não encontrado, fazer:
3.3.1 - Para cada idreg mantido no registro corrente de INV, fazer:
3.3.1.1 - Buscar o registro de ARQ1 apontado pelo mesmo.
3.3.1.2 - Adicionar à LISTA DE LIGANTES um registro contendo:
- este registro trazido de ARQ1.
- indicação de tabela de ligados vazia.

LIG2B (ARQ1, ARQ2)

- Condição:

ARQ é um arquivo de dados para o qual existe um arquivo de inversões cuja chave primária inclui como "primeiros" componentes os atributos de ligação. ARQ2 é um arquivo/lista de dados.

- Resultado:

Uma lista de dados de ligantes (LISTA DE LIGANTES), classificada segundo a chave primária do arquivo de inversões, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante se encontra classificado segundo os atributos adicionais sob os quais está classificado ARQ2, além daqueles atributos mantidos na chave de inversão sobre ARQ1. (Caso não existam esses "atributos adicionais" na ordem de ARQ2, os grupos de ligados não se encontram classificados sob ordem alguma).

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ1.

LARG2 - É o próprio ARQ2, se o mesmo já estiver classificado, ou uma lista de dados contendo ARQ2 classificado.

Passos:

1 - Verificar se ARQ2 está classificado segundo a mesma ordem que os atributos de ligação mantidos na chave de inversão.
Sim → seguir
Não → classificar ARQ2 segundo esta ordem (gerando LARQ2).

2 - ULTLIGADO ← (primeira posição a ser preenchida

na LISTA DE LIGADOS) - 1.

3 - ANTERIOR ← valores inválidos para os atributos de ligação.

4 - Para cada registro de INV, fazer:

4.1 - Se ANTERIOR ≠ valores dos atributos de ligação ocorrentes nesse registro de INV, então:

4.1.1 - ANTERIOR ← valores dos atributos de ligação ocorrentes nesse registro de INV.

4.1.2 - Buscar em LARQ2 (seqüencialmente, a partir do último registro visitado) um registro cujos valores dos atributos de ligação sejam iguais aos atributos correspondentes mantidos na construção sob nome ITENS do registro corrente de INV.

4.1.3 - Se encontrado, fazer:

4.1.3.1 - Buscar o registro de ARQ1 apontado pelo primeiro idreg mantido no registro corrente de INV.

4.1.3.2 - PRIMLIGADO ← ULTLIGADO + 1.

4.1.3.3 - Para cada registro de LARQ2 cujos valores dos atributos de ligação sejam iguais aos mantidos na construção sob nome ITENS do registro corrente de INV, fazer:

4.1.3.3.1 - Formar um registro com os valores deste registro de LARQ2, de acordo com a configuração de um ligado.

4.1.3.3.2 - Adicionar à LISTA DE LIGADOS o registro formado.

4.1.3.3.3 - ULTLIGADO ← ULTLIGADO + 1.

4.1.3.4 - Para cada idreg mantido no registro corrente de INV, fazer:

4.1.3.4.1 - Buscar o registro de ARQ1 apontado pelo mesmo.

4.1.3.4.2 - Adicionar à LISTA DE LIGANTES um registro contendo:

- este registro trazido de ARQ1
- PRIMLIGADO
- ULTLIGADO.

4.1.4 - Se não encontrado, então:

4.1.4.1 - Para cada idreg mantido no registro corrente de INV, fazer:

4.1.4.1.1 - Buscar o registro de ARQ1 apontado pelo mesmo.

4.1.4.1.2 - Adicionar à LISTA DE LIGANTES um registro contendo:

- este registro trazido de ARQ1,
- indicação de tabela de ligados vazia.

4.2 - Do contrário, fazer:

/* ANTERIOR = valores dos atributos de ligação mantidos na construção sob nome ITENS desse registro de inversão */

4.2.1 - Para cada idreg mantido no registro corrente de INV, fazer:

4.2.1.1 - Buscar o registro de ARQ1 apontado pelo mesmo.

4.2.1.2 - Adicionar à LISTA DE LIGANTES um registro contendo:

- este registro trazido de ARQ1
- PRIMLIGADO
- ULTLIGADO.

LIG3A (ARQ1, ARQ2)

- Condição:

ARQ1 é um arquivo/lista de dados. ARQ2 é um arquivo de dados para o qual existe definido um arquivo de inversões:

- exatamente sobre o conjunto dos atributos de ligação; ou
- exatamente sobre um subconjunto dos atributos de ligação, que é chave primária.

- Resultado:

Uma lista de dados de ligantes (LISTA DE LIGANTES), classificada segundo a ordem de ARQ1, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante não se encontra classificado sob ordem alguma.

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ2.

Observação: Se ARQ1 está classificado segundo o arquivo de inversões, então o arquivo de inversões é percorrido seqüencialmente. Do contrário, deve ser usada a primitiva que busca registro de inversão com determinado valor.

Passos:

1 - ULTLIGADO ← (primeira posição a ser preenchida na LISTA DE LIGADOS) - 1.

2 - Para cada registro de ARQ1, fazer:

2.1 - Buscar em INV um registro cuja construção sob nome ITENS tenha conteúdo igual ao dos respectivos campos do registro corrente de ARQ1.

2.2 - Se encontrado, fazer:

2.2.1 - Se a chave primária do arquivo de inversões é composta de todos os atributos de ligação, então:

2.2.1.1 - PRIMLIGADO ← ULTLIGADO + 1.

2.2.1.2 - ULTLIGADO ← ULTLIGADO + (conteúdo da construção sob nome CARD-IDREGS desse registro de INV).

2.2.1.3 - Para cada idreg mantido no registro corrente de INV, fazer:

2.2.1.3.1 - Buscar o registro de ARQ2 apontado pelo mesmo.

2.2.1.3.2 - Formar um registro com os valores desse registro trazido de ARQ2, de acordo com a configuração de um ligado.

2.2.1.3.3 - Adicionar à LISTA DE LIGADOS o registro formado.

2.2.1.4 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro corrente de ARQ1
- PRIMLIGADO
- ULTLIGADO.

2.2.2 - Do contrário, fazer:
/* Este é um arquivo de inversões sobre chave primária */

2.2.2.1 - Buscar o registro de ARQ2 apontado pelo idreg mantido no registro corrente de INV.

2.2.2.2 - Se o registro corrente de ARQ1 e o registro trazido de ARQ2 satisfazem a toda a expressão de ligação, então:

2.2.2.2.1 - ULTLIGADO ← ULTLIGADO + 1.

2.2.2.2.2 - PRIMLIGADO ← ULTLIGADO.

2.2.2.2.3 - Formar um registro com os valores do registro trazido de ARQ2, de acordo com a configuração de um ligado.

2.2.2.2.4 - Adicionar à LISTA DE LIGADOS o registro formado.

2.2.2.2.5 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro corrente de ARQ1
- PRIMLIGADO
- ULTLIGADO

2.2.2.3 - Do contrário, fazer:
/* A expressão de ligação não é totalmente satisfeita */

2.2.2.3.1 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro corrente de ARQ1.
- indicação de tabela de ligados vazia.

2.3 - Se não encontrado, fazer:

2.3.1 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro corrente de ARQ1.
- indicação de tabela de ligados vazia.

LIG3B (ARQ1, ARQ2)

- Condição:

ARQ1 é um arquivo/lista de dados. ARQ2 é um arquivo de dados para o qual existe definido um arquivo de inversões cuja chave primária inclui como "primeiros" componentes os atributos de ligação.

- Resultado:

Uma lista de dados de ligantes (LISTA DE LIGANTES), classificada segundo a ordem de ARQ1, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante se encontra classificado segundo os atributos adicionais existentes na chave de inversão sobre ARQ2, além dos atributos de ligação.

- Método:

Estruturas de dados utilizadas:

INV - Arquivo de inversões sobre ARQ2.

Observação: Se ARQ1 está classificado segundo o arquivo de inversões, então o arquivo de inversões é percorrido seqüencialmente. Do contrário, deve ser usada a primitiva que busca registro de inversão com determinado valor.

Passos:

1 - ULTLIGADO ← (primeira posição a ser preenchida na LISTA DE LIGADOS) - 1.

2 - Para cada registro de ARQ1, fazer:

2.1 - Buscar em INV um registro cuja construção sob nome ITENS tenha conteúdo igual ao dos respectivos campos do registro corrente de ARQ1.

2.2 - Se encontrado, fazer:

2.2.1 - PRIMLIGADO ← ULTLIGADO + 1.

2.2.2 - Para cada registro de INV cujos valores dos atributos de ligação mantidos na construção sob nome ITENS sejam iguais aos do registro corrente de ARQ1, fazer:

2.2.2.1 - ULTLIGADO ← ULTLIGADO + (conteúdo da construção sob nome CARD-IDREGS desse registro de INV).

2.2.2.2 - Para cada idreg mantido nesse registro de INV, fazer:

2.2.2.2.1 - Buscar o registro de ARQ2 apontado por este idreg.

2.2.2.2.2 - Formar um registro com os valores do registro trazido de ARQ2, de acordo com a configuração de um ligado.

2.2.2.2.3 - Adicionar à LISTA DE LIGADOS o registro formado.

2.2.3 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro corrente de ARQ1
- PRIMLIGADO
- ULTLIGADO.

2.3 - Se não encontrado, fazer:

2.3.1 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro corrente de ARQ1
- PRIMLIGADO
- ULTLIGADO.

LIG4 (ARQ1, ARQ2, ORD)

- Condição:

ARQ1, ARQ2 são arquivos/listas de dados.

- Resultado:

Se ORD for uma ordem válida, resultam uma lista de dados ligantes (LISTA DE LIGANTES), classificada segundo a ordem ORD, e uma lista de dados de ligados (LISTA DE LIGADOS), onde cada grupo de ligados de um ligante se encontra classificado segundo os atributos adicionais sob os quais está classificado ARQ2, além dos atributos de ligação (caso não existam esses "atributos adicionais" na ordem de ARQ2, os grupos de ligados não se encontram classificados sob ordem alguma).

Se ORD não for uma ordem válida, resultam uma lista de dados de ligantes (LISTA DE LIGANTES), classificada:

I) Segundo a ordem em que os atributos de ligação aparecem nas tuplas de ARQ1 (se nem ARQ1, nem ARQ2 estiverem classificados segundo alguma ordem dos atributos de ligação).

II) Segundo a ordem de classificação do arquivo/lista de dados de maior cardinalidade (se ambos ARQ1 e ARQ2 já estiverem classificados segundo diferentes ordens dos atributos de ligação).

... e uma lista de dados de ligados (LISTA DE ligados), onde cada grupo de ligados de um ligante se encontra classificado segundo os atributos adicionais sob os quais está classificado ARQ2, além dos atributos de ligação.

- Método:

Estruturas de dados utilizadas:

LARQ1 - é o próprio ARQ1, ou uma lista de dados contendo ARQ1 classificado.

LARQ2 - é o próprio ARQ2, ou uma lista de dados contendo ARQ2, classificado.

Passos:

1 - Se ORD for uma ordem válida, então:

1.1 - Verificar se ARQ1 está classificado segundo a ordem ORD dos atributos de ligação.
Sim → seguir.
Não → classificar ARQ1 segundo a ordem ORD dos atributos de ligação (gerando LARQ1).

1.2 - Verificar se ARQ2 está classificado segundo a ordem ORD dos atributos de ligação.
Sim → seguir.
Não → classificar ARQ2 segundo a ordem ORD dos atributos de ligação (gerando LARQ2).

2 - Se ORD não for uma ordem válida, então:

2.1 - Verificar se ambos ARQ1, ARQ2 já estão classificados por diferentes ordens dos atributos de ligação.
Sim → classificar o arquivo de menor cardinalidade, de acordo com a ordem de classificação do outro (gerando LARQ1, se o arquivo a classificar for ARQ1, ou LARQ2, caso contrário).
Não → classificar ambos ARQ1, ARQ2 pela ordem em que os atributos de ligação aparecem nas tuplas de ARQ1 (gerando LARQ1,LARQ2).

3 - ANTERIOR ← valores inválidos para o conjunto de atributos de ligação.

4 - ULTLIGADO ← (primeira posição a ser preenchida na LISTA DE LIGADOS) - 1.

5 - Para cada registro de LARQ1, fazer:

5.1 - Se ANTERIOR ≠ valores dos atributos de ligação ocorrentes nesse registro de LARQ1, fazer:

5.1.1 - ANTERIOR ← valores dos atributos de ligação ocorrentes neste registro de LARQ1.

5.1.2 - Buscar em LARQ2 (seqüencialmente, a partir do último registro visitado) um registro cujos valores dos atributos de ligação sejam iguais aos mantidos em ANTERIOR.

5.1.3 - Se encontrado, fazer:

5.1.3.1 - PRIMLIGADO ← ULTLIGADO + 1.

5.1.3.2 - Para cada registro de LARQ2 cujos valores dos atributos de ligação sejam iguais aos mantidos em ANTERIOR, fazer:

5.1.3.2.1 - ULTLIGADO ← ULTLIGADO + 1.

5.1.3.2.2 - Formar um registro com os valores deste registro de ARQ2, de acordo com a configuração de um ligado.

5.1.3.2.3 - Adicionar à LISTA DE LIGADOS o registro formado.

5.1.3.3 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro corrente de LARQ1
- PRIMLIGADO
- ULTLIGADO.

5.1.4 - Se não encontrado, fazer:

5.1.4.1 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro corrente de LARQ1.
- indicação de tabela de ligados vazia.

5.2 - Do contrário:

5.2.1 - Adicionar à LISTA DE LIGANTES um registro contendo:
- o registro corrente de LARQ1
- PRIMLIGADO
- ULTLIGADO.

/* Sendo esta uma tabela de ligados vazia, então PRIMLIGADO e ULTLIGADO já contêm valores adequados*/

14 - ESTREITT

Algoritmos apresentados:

- ESTREITT1: percorre seqüencialmente apenas os registros de inversão sobre os ligados, obtendo, a partir dos mesmos, os registros estreitados.

- ESTREITT2: idem ao ESTREITT1, porém elimina itens para obter os registros estreitados, preocupando-se com eliminação de duplicatas.

- ESTREITT3: percorre seqüencialmente os registros de inversão buscando os registros de dados e realizando o estreitamento.

- ESTREITT4: realiza o equivalente ao ESTREITT3, classificando a lista resultante do estreitamento a fim de eliminar duplicatas.

- ESTREITT5: busca os ligados de uma lista (mantida na zona intermediária ou canal auxiliar) e realiza o estreitamento.

- ESTREITT6: realiza o equivalente ao ESTREITT5, classificando a lista resultante do estreitamento a fim de eliminar duplicatas.

ESTREITT1 (AS)

- Condição:

AS é um registro de associação por item ou por idreg. Os únicos atributos mantidos são os que constituem a chave primária da tabela de ligados (conteúdo da construção sob nome ITENS dos registros de inversão mantidos na construção sob nome LIGADOS).

- Resultado:

Lista de dados classificada segundo a chave primária da tabela de ligados de AS.

- Método:

Passos:

1 - Para cada registro de inversão mantido na construção sob nome LIGADOS, fazer:

1.1 - Adicionar à lista resultado um registro cujo conteúdo é exatamente o conteúdo da conttrução sob nome ITENS do presente registro de inversão.

ESTREITT2 (AS)

- Condição:

AS é um registro de associação por item ou por idreg. Os atributos mantidos são os "primeiros" da chave primária da tabela de ligados.

- Resultado:

Lista de dados classificada segundo os atributos mantidos pelo estreitamento e constantes na chave primária da tabela de ligados de AS.

- Método:

Passos:

1 - ANTERIOR ← valores dos atributos mantidos pelo estreitamento ocorrentes no primeiro registro de inversão mantido na construção sob nome LIGADOS.

2 - Adicionar à lista resultado um registro contendo os valores mantidos em ANTERIOR.

3 - Para cada registro de inversão mantido na construção sob nome LIGADOS, fazer:

3.1 - Se o conteúdo da construção sob nome ITENS do registro corrente de INV é diferente do mantido em ANTERIOR, nos campos correspondentes, então:

3.1.1 - ANTERIOR ← valores dos atributos mantidos pelo estreitamento ocorrentes na construção sob nome ITENS desse registro de INV.

3.1.2 - Adicionar à lista resultado um registro contendo os valores mantidos em ANTERIOR.

ESTREITT3 (AS)

- Condição:

AS é um registro de associação por item ou por idreg. Entre os atributos mantidos está a chave primária da tabela de ligados.

- Resultado:

Lista de dados classificada segundo a chave primária da tabela de ligados de AS.

- Método:

Estruturas de dados utilizadas:

ARQD2 - Arquivo de dados dos ligados.

Passos:

1 - Para cada registro de inversão mantido na construção sob nome LIGADOS, fazer:

1.1 - Buscar o registro de ARQD2 apontado pelo idreg mantido neste registro de inversão.

1.2 - Formar um registro de dados com os atributos mantidos pelo estreitamento desse registro de ARQD2.

1.3 - Adicionar esse registro à lista resultado.

ESTREITT4 (AS, ORD)

- Condição:

AS é um registro de associação por item ou por idreg.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD dos atributos mantidos pelo estreitamento. Do contrário, resulta uma lista de dados classificada segundo a ordem em que os atributos mantidos aparecem nos registros de ARQD2.

- Método:

Estruturas de dados utilizadas:

ARQD2 - Arquivo de dados dos ligados.
LT - Lista de registros de ARQD2.

Passos:

1 - Para cada registro de inversão mantido na construção sob nome LIGADOS, fazer:
1.1 - Buscar o registro de ARQD2 apontado pelo idreg mantido neste registro de inversão.
1.2 - Formar um registro de dados com os atributos mantidos pelo estreitamento desse registro de ARQD2.
1.3 - Adicionar o mesmo à lista LT.

2 - Se ORD for uma ordem válida, então classificar LT segundo a ordem ORD dos atributos mantidos (eliminando as duplicatas e gerando a lista resultado).

3 - Do contrário, classificar LT segundo a ordem em que os atributos mantidos aparecem nos registros

de ARQD2 (eliminando as duplicatas e gerando o resultado).

/* Supõe-se que a lista LT nunca estará classificada segundo os atributos mantidos. Se estes últimos são parte ou toda a chave primária dos ligados, os algoritmos escolhidos são o ESTREITT2 ou ESTREITT1, respectivamente */

ESTREITT5 (AS)

- Condição:

AS é um registro de associação mantido no canal auxiliar ou na zona intermediária. Entre os atributos mantidos pelo estreitamento, está a chave primária dos ligados.

- Resultado:

Lista de dados classificada segundo a tabela de ligados de AS.

- Método:

Estruturas de dados utilizadas:

LISTA DE LIGADOS - Lista de dados contendo ligados.

Passos:

1 - Para cada registro da LISTA DE LIGADOS cuja posição esteja entre PRIMLIGADO e ULTLIGADO inclusive, fazer:

1.1 - Formar um registro de dados com os atributos mantidos pelo estreitamento desse registro da LISTA DE LIGADOS.

1.2 - Adicioná-lo à lista resultado.

ESTREITT6 (AS, ORD)

- Condição:

AS é um registro de associação mantido no canal auxiliar ou na zona intermediária.

- Resultado:

Se ORD for uma ordem válida, resulta uma lista de dados classificada segundo a ordem ORD dos atributos mantidos pelo estreitamento. Do contrário, resulta uma lista de dados classificada segundo a ordem em que os atributos mantidos aparecem nos registros da LISTA DE LIGADOS.

- Método:

Estruturas de dados utilizadas:

LISTA DE LIGADOS - Lista de dados contendo ligados.
LT - Lista de registros estreitados.

Passos:

1 - Se ORD for uma ordem válida, então:
1.1 - Para cada registro da LISTA DE LIGADOS cuja posição esteja entre PRIMLIGADO e ULTLIGADO, inclusive, fazer:
1.1.1 - Formar um registro de dados com os atributos mantidos pelo estreitamento desse registro da LISTA DE LIGADOS.
1.1.2 - Verificar se o registro formado é igual ao último registro inserido na lista resultado.
Sim → seguir.
Não → adicionar esse registro à lista LT.
1.2 - Classificar a lista LT segundo a ordem ORD, eliminando as duplicatas e gerando a lista resultado.

2 - Se ORD não for uma ordem válida, então:

2.1 - Para cada registro da LISTA DE LIGADOS cuja posição esteja entre PRIMLIGADO e ULTLIGADO inclusive, fazer:

2.1.1 - Formar um registro de dados com os atributos mantidos pelo estreitamento desse registro da LISTA DE LIGADOS.

2.1.2 - Adicionar o mesmo à lista LT.

2.2 - Classificar a lista LT segundo a ordem em que os atributos mantidos aparecem nos registros da LISTA DE LIGADOS (eliminando as duplicatas e gerando a lista resultado).

15 - COLECT

Algoritmos apresentados:

- COLECT1: percorre seqüencialmente os registros de inversão sobre os ligados, buscando apenas os registros de dados dos ligados colecionados.

- COLECT2: idem ao COLECT1, porém sempre busca os registros de dados, para todos os ligados.

- COLECT3: percorre seqüencialmente o grupo de ligados de um mesmo ligante, mantido numa lista de ligados na zona intermediária ou canal auxiliar.

COLECT1 (AS)

- Condição:

AS é um registro de associação por item ou por idreg e a expressão seletora envolve apenas atributos que pertencem à chave primária da tabela de ligados.

- Resultado:

Lista de dados classificada segundo a chave primária da tabela de ligados de AS.

- Método:

Estruturas de dados utilizadas:

ARQD2 - Arquivo de dados dos ligados.

Passos:

1 - Para cada registro de inversão contido na construção sob nome LIGADOS, fazer:

1.1 - Se o conteúdo da construção sob nome ITENS desse registro de inversão satisfaz a expressão de seleção, então:

1.1.1 - Buscar o registro de ARQD2 apontado pelo idreg mantido nesse registro.

1.1.2 - Adicionar o mesmo à lista resultado.

COLECT2 (AS)

- Condição:

AS é um registro de associação por item ou por idreg.

- Resultado:

Lista de dados classificada segundo a chave primária da tabela de ligados de AS.

- Método:

Estruturas de dados utilizadas:

ARQD2 - Arquivo de dados dos ligados.

Passos:

1 - Percorrer a expressão seletora e subdividi-la nos seguintes trechos:
sel_1 - subexpressões resolvíveis por acesso à chave primária da tabela de ligados.
sel_2 - subexpressões não resolvíveis por acesso à chave primária da tabela de ligados (exigem acesso aos registros de dados).

2 - Para cada registro de inversão contido na construção sob nome LIGADOS, fazer:

2.1 - Se a construção sob nome ITENS desse registro de inversão satisfaz o trecho sel_1 da expressão seletora, então:

2.1.1 - Buscar o registro de ARQD2 apontado pelo idreg mantido nesse registro de inversão.

2.1.2 - Se o mesmo satisfaz o trecho sel_2 da expressão seletora, então:

2.1.2.1 - Adicionar este registro à lista resultado.

Observações:

- Se o trecho sel_1 da expressão seletora for vazio, o mesmo será considerado como sempre VERDADEIRO.

- Se o trecho sel_2 da expressão seletora for vazio, terá sido escolhido o algoritmo COLECT1.

COLECT3 (AS)

- Condição:

AS é um registro de associação mantido na zona intermediária ou no canal auxiliar.

- Resultado:

Lista de dados classificada segundo a tabela de ligados de AS.

- Método:

Estruturas de dados utilizadas:

LISTA DE LIGADOS - Lista de dados contendo ligados.

Passos:

1 - Para cada registro da LISTA DE LIGADOS cuja posição esteja entre PRIMLIGADO e ULTLIGADO inclusive, fazer:

1.1 - Se o mesmo satisfaz a expressão seletora, então:

1.1.1 - Adicioná-lo à lista resultado.

16 - AGRUPART

Algoritmos apresentados:

- AGRUPART1: percorre seqüencialmente os registros de inversão sobre os ligados, obtendo cada ligante a partir da construção sob nome ITENS desses registros.

- AGRUPART2: produz uma lista classificada contendo os ligados e a percorre seqüencialmente, realizando o agrupamento.

- AGRUPART3: percorre seqüencialmente um grupo de ligados de uma associação mantida na zona intermediária ou canal auxiliar, realizando o agrupamento.

- AGRUPART4: idem ao AGRUPART3, porém antes produz uma lista classificada, e opera sobre esta lista realizando o agrupamento.

AGRUPART1 (AS)

- Condição:

AS é um registro de associação por item ou por idreg, e a chave primária de sua tabela de ligados é constituída exatamente pelos atributos de agrupamento, ou os atributos de agrupamento são "os primeiros" dessa chave primária.

- Resultado:

Lista de dados de ligantes (LL1), classificada segundo a chave primária da tabela de ligados, e lista de dados de ligados (LL2), onde cada grupo de ligados de um ligante está classificado segundo os atributos adicionais existentes na chave primária da tabela de ligados, além dos atributos de agrupamento.

- Método:

Estruturas de dados utilizadas:

ARQD2 - Arquivo de dados contendo os ligados de AS.

Passos:

1 - ANTERIOR ← valores dos atributos de agrupamento mantidos na construção sob nome ITENS do primeiro registro de inversão mantido na construção sob nome LIGADOS.

2 - UL ← (próxima posição a ser ocupada na lista LL2) - 1.

3 - PL ← UL + 1.

4 - Para cada registro de inversão mantido na construção sob nome LIGADOS, fazer:

4.1 - Se os valores dos atributos de agrupamento man-

tidos na construção sob nome ITENS desse registro de inversão são diferentes dos mantidos em ANTERIOR, fazer:

4.1.1 - Adicionar à lista LL1 um registro contendo os valores de:
- ANTERIOR (dados do ligante dessa ligação).
- PL (posição do primeiro ligado dessa ligação).
- UL (posição do último ligado dessa ligação).

4.1.2 - ANTERIOR ← valores dos atributos de agrupamento mantidos no registro de inversão correntemente em análise.

4.1.3 - PL ← UL + 1.

4.2 - Buscar o registro de ARQD2 apontado por este registro de inversão.

4.3 - Formar um registro com os campos desse registro de ARQD2 que não constituem atributos de agrupamento.

4.4 - Adicionar à lista LL2 o registro formado.

4.5 - UL ← UL+1.

5 - Adicionar à lista LL1 um registro contendo os valores de:
- ANTERIOR (dados do ligante dessa ligação).
- PL (posição do primeiro ligado dessa ligação).
- UL (posição do último ligado dessa ligação).

AGRUPART2 (AS, ORD)

- Condição:

AS é um registro de associação por item ou por idreg.

- Resultado:

Uma lista de dados de ligantes (LL1) classificada:

I) Se ORD for uma ordem válida, segundo a ordem ORD dos atributos de agrupamento;

II) Se ORD não for uma ordem válida, segundo a ordem em que os atributos de agrupamento aparecem nos registros de dados de ligados de AS.

... e uma lista de dados de ligados (LL2) onde cada grupo de ligados de um ligante não se encontra classificada sob ordem alguma.

- Método:

Estruturas de dados utilizadas:

LT - lista de dados.
LTC- lista LT classificada.

Passos:

1 - Para cada registro de inversão mantido na construção sob nome LIGADOS, fazer:
1.1 - Buscar o registro de ARQD2 apontado pelo idreg mantido nesse registro de inversão.
1.2 - Adicioná-lo à lista LT.

2 - Se ORD for uma ordem válida, então:
2.1 - Classificar a lista LT segundo a ordem ORD (gerando LTC).

3 - Do contrário, fazer:

3.1 - Classificar a lista LT pelos atributos de agrupamento, como estes aparecem nos registros de ARQD2 (gerando LTC).

4 - ANTERIOR ← valores dos atributos de agrupamento ocorrentes no primeiro registro de LTC.

5 - UL ← (próxima posição a ser preenchida na lista LL2) - 1.

6 - PL ← UL + 1.

7 - Para cada registro de LTC, fazer:

7.1 - Se os valores dos atributos de agrupamento ocorrentes neste registro de LTC são diferentes dos mantidos em ANTERIOR, fazer:

7.1.1 - Adicionar à lista LL1 um registro contendo os valores de:
- ANTERIOR (dados do ligante dessa ligação).
- PL (posição do primeiro ligado).
- UL (posição do último ligado).

7.1.2 - ANTERIOR ← valores dos atributos de agrupamento ocorrentes neste registro de LTC.

7.1.3 - PL ← UL + 1.

7.2 - UL ← UL + 1.

7.3 - Formar um registro com os campos desse registro de LTC que não constituem atributos de agrupamento.

7.4 - Adicionar à lista LL2 o registro formado.

8 - Adicionar à lista LL1 um registro contendo os valores de:
- ANTERIOR (dados do ligante dessa ligação).
- PL (posição do primeiro ligado).
- UL (posição do último ligado).

AGRUPART3 (AS)

- Condição:

AS é um registro de associação mantido na zona intermediária ou no canal auxiliar, e sua tabela de ligados está classificada segundo os atributos de agrupamento.

- Resultado:

Lista de dados de ligantes (LL1) classificada segundo a ordem da tabela de ligados de AS, e lista de dados de ligados (LL2) classificada segundo os atributos adicionais existentes na ordem de classificação da tabela de ligados de AS, além dos atributos de agrupamento.

- Método:

Estruturas de dados utilizadas:

LISTA DE LIGADOS - lista contendo dados dos ligados de AS.

Passos:

1 - ANTERIOR ← valores dos atributos de agrupamento ocorrentes no registro de LISTA DE LIGADOS cuja posição é mantida em PRIMLIGADO.

2 - UL ← (próxima posição a ser ocupada na lista LL2) - 1.

3 - PL ← UL + 1.

4 - Para cada registro da LISTA DE LIGADOS com posição entre PRIMLIGADO e ULTLIGADO, inclusive, fazer:

4.1 - Se os valores dos atributos de agrupamento ocorrentes neste registro da LISTA DE LIGADOS são diferentes dos mantidos em ANTERIOR, fazer:

4.1.1 - Adicionar à lista LL1 um registro contendo os valores de:
- ANTERIOR (dados do ligante dessa ligação).
- PL (posição do primeiro ligado).
- UL (posição do último ligado).

4.1.2 - ANTERIOR ← valores dos atributos de agrupamento ocorrentes no registro em análise na LISTA DE LIGADOS.

4.1.3 - PL ← UL + 1.

4.2 - UL ← UL + 1.

4.3 - Formar um registro com os campos desse registro da LISTA DE LIGADOS que não constituem atributos de agrupamento.

4.4 - Adicionar à lista LL2 o registro formado.

5 - Adicionar à lista LL1 um registro contendo os valores de:
- ANTERIOR (dados do ligante dessa ligação).
- PL (posição do primeiro ligado).
- UL (posição do último ligado).

AGRUPART4 (AS, ORD)

- Condição:

AS é um registro de associação mantido na zona intermediária ou no canal auxiliar.

- Resultado:

Uma lista de dados de ligantes (LL1) classificada:

I) Se ORD for uma ordem válida, segundo a ordem ORD dos atributos de agrupamento;

II) Se ORD não for uma ordem válida, segundo a ordem em que os atributos de agrupamento aparecem nos registros da tabela de ligados de AS.

... e uma lista de dados de ligados (LL2) onde cada grupo de ligados de um ligante não se encontra classificado sob ordem alguma.

- Método:

Estruturas de dados utilizadas:

LISTA DE LIGADOS - lista contendo dados dos ligados de AS.
LT - lista de registros da LISTA DE LIGADOS.
LTC - lista LT classificada.

Passos:

1 - Para cada registro da LISTA DE LIGADOS com posição entre PRIMLIGADO e ULTLIGADO, inclusive, fazer:

1.1 - Adicionar o mesmo à lista LT.

2 - Se ORD for uma ordem válida, então:

2.1 - Classificar a lista LT segundo a ordem ORD dos atributos de agrupamento (gerando LTC).

3 - Do contrário, fazer:

3.1 - Classificar a lista LT segundo a ordem em que os atributos de agrupamento aparecem nos registros da LISTA DE LIGADOS (gerando LTC).

4 - ANTERIOR ← valores dos atributos de agrupamento ocorrentes no primeiro registro da lista LTC.

5 - UL ← (próxima posição ocupada na lista LL2) - 1.

6 - PL ← UL + 1.

7 - Para cada registro da lista LTC fazer:

7.1 - Se os valores dos atributos de agrupamento ocorrentes nestes registro são diferentes dos mantidos em ANTERIOR, fazer:

7.1.1 - Adicionar à lista LL1 um registro contendo os valores de:
- ANTERIOR (dados do ligante dessa ligação).
- PL (posição do primeiro ligado).
- UL (posição do último ligado).

7.1.2 - ANTERIOR ← valores dos atributos de agrupamento ocorrentes nesse registro de LTC.

7.1.3 - PL ← UL + 1.

7.2 - UL ← UL + 1.

7.3 - Formar um registro com os campos desse registro de LTC que não constituem atributos de agrupamento.

7.4 - Adicionar à lista LL2 o registro formado.

8 - Adicionar à lista LL1 um registro contendo os valores de:
- ANTERIOR (dados do ligante dessa ligação).
- PL (posição do primeiro ligado).
- UL (posição do último ligado).

6. ADEQUAÇÃO DOS ALGORITMOS

6.1 Introdução

Entre os algoritmos propostos para as diversas operações, existem opções dirigidas a situações específicas, sempre com o objetivo de economizar recursos de tempo e espaço, a partir das condições existentes em um certo nodo da árvore de operações.

Este capítulo tem por objetivo mostrar qual o algoritmo mais interessante a cada situação, para cada uma das operações estudadas. A adequação dos algoritmos é mostrada na forma de tabelas descrevendo, para cada operação, o conjunto de condições que podem ocorrer, e o algoritmo que melhor serve a cada uma delas.

6.2 Critérios observados

As primeiras condições analisadas, junto à maioria das operações, dizem respeito às ordens de classificação existentes sobre os operandos; logo a seguir, é verificada a disponibilidade de estruturas de dados auxiliares.

A forma de resolução das operações está sempre condicionada ao modo como os operandos são mantidos internamente. Por exemplo, uma tabela ligacional pode estar armazenada, internamente, como um arquivo de associações por item, como um arquivo de associações por idreg ou como uma lista de ligantes + lista de ligados mantida na zona intermediária ou no canal auxiliar. Uma tabela relacional, por sua vez, pode estar na própria base de dados (armazenada como um arquivo de dados não classificado, mas talvez possuidor de estruturas de acesso adicionais) ou na zona intermediária ou canal auxiliar (sob forma de uma lista de dados que pode estar classificada segundo alguma ordem).

Os critérios básicos utilizados na seleção de um algoritmo incluem:

1º) No caso das operações que necessitam entradas classificadas, o primeiro critério levado em conta é o aproveitamento das ordens de classificação já existentes, permitindo utilizar diretamente os arquivos/listas de dados disponíveis.

2º) No caso de operações que manipulam operandos da própria base de dados, tenta-se o aproveitamento das estruturas de dados disponíveis. Sempre que o volume de recursos a utilizar puder ser diminuído com o uso de arquivos de inversões ou arquivos de associações, o algoritmo escolhido fará uso destas estruturas.

Por exemplo, é possível citar o algoritmo JUNT5, que realiza uma operação JUNTAR a partir de um arquivo de associações definido sobre os atributos de junção, envolvendo os dois operandos (no caso, arquivos de dados) dessa operação.

Apenas em último caso é feita a classificação de arquivos ou listas de dados, devido ao sensível acréscimo que acarreta sobre o tempo de execução das operações, além de propiciar a geração de um arquivo intermediário adicional.

6.3 Tabelas de adequação dos algoritmos

6.3.1 Apresentação

A adequação dos respectivos algoritmos à resolução de cada operação é mostrada na forma de tabelas compostas de 2 colunas, a saber: CONDIÇÕES e ALGORITMO.

A leitura das tabelas apresentadas deve ser feita de cima para baixo, e da esquerda para a direita. Cada grupo de condições supõe sempre a negação de todos os conjuntos de condições anteriores, isto é: o algoritmo de uma linha pode ser escolhido apenas se nenhum dos algoritmos das linhas anteriores tiver sido escolhido. Assim, fica estabelecida uma prioridade, quando existem diferentes opções de algoritmos disponíveis para uma mesma operação.

Com a finalidade de uniformizar-se a apresentação do estudo de adequação dos algoritmos, o termo "arquivo de associações" será utilizado daqui por diante para representar, também, as tabelas ligacionais mantidas na zona intermediária ou no canal auxiliar, sob a forma de lista de ligantes + lista de ligados. O termo "registro de associação" (ou, simplesmente, "associação") fica, em decorrência, estendido para representar também uma ligação pertencente a uma tabela ligacional mantida na zona intermediária ou no canal auxiliar.

Considerando as características particulares da implementação a ser realizada, para cada operação, resolveu-se eliminar, na etapa de seleção, alguns algoritmos cuja execução será certamente mais demorada, devido ao grande número de pesquisas realizadas em listas. Este é o caso dos algoritmos UNI3, UNI4, ISEC2, ISEC4, DIF2, DIF3 e DIF4.

A tabela 6.1 contém os termos utilizados nas tabelas de adequação dos algoritmos, com o respectivo significado.

TABELA 6.1: Termos utilizados nas tabelas de adequação

TERMO	SIGNIFICADO
ARQ1,ARQ2,ARQ	Arquivos/listas de dados
ASS	Arquivo de associações por item ou por idreg, ou lista de ligantes + lista de ligados representando uma tabela ligacional mantida na zona intermediária ou no canal auxiliar.
AS	Registro de associação de um arquivo de associações por item ou por idreg ou de uma lista de ligantes + lista de ligados representando uma tabela ligacional mantida na zona intermediária ou no canal auxiliar.
REGDADOS	Registro de dados no topo da pilha.
REGASS	Registro de associação no topo da pilha.

6.3.2 Tabelas desenvolvidas

São mostradas, a seguir, as tabelas de adequação dos algoritmos desenvolvidos para as operações estudadas (tabelas 6.2 a 6.20).

TABELA 6.2: COLR(ARQ), COLRU(ARQ) - Adequação

CONDIÇÕES	ALGORITMO
ARQ possui um arquivo de inversões definido sobre atributos que estão envolvidos em uma subexpressão de seleção, e as subexpressões de seleção estão todas ligadas por ET (em primeiro nível).	COLR2
Em caso contrário	COLR1

TABELA 6.3: COLA(ASS) - Adequação

CONDIÇÕES	ALGORITMO
ASS é um arquivo de associações por item e pelo menos uma das subexpressões de primeiro nível, que estão todas ligadas por ET, é do tipo C A $\begin{matrix}>\\=\\<\end{matrix}$ valor, onde A é a chave primária de ASS.	COLA2
ASS é um arquivo de associações por item ou idreg, e existe pelo menos uma subexpressão de primeiro nível, do tipo C A = valor, envolvendo atributos dos ligantes sobre os quais existe definido um arquivo de inversões. As subexpressões de primeiro nível estão todas ligadas por ET.	COLA3
ASS é um arquivo de associações por idreg e existe pelo menos uma subexpressão de primeiro nível envolvendo dados do ligante. As subexpressões de 1º nível estão todas ligadas por ET. A subexpressão envolvendo dados do ligante pode conter outros conectores lógicos.	COLA4
ASS é um arquivo de associações mantido na zona intermediária ou no canal auxiliar.	COLA5
Nenhuma das anteriores.	COLA1

TABELA 6.4: COLAU(ASS) - Adequação

CONDIÇÕES	ALGORITMO
ASS é um arquivo de associações por item.	COLA2
ASS é um arquivo de associações por idreg.	COLA3
ASS é um arquivo de associações mantido na zona intermediária ou no canal auxiliar.	COLA5

TABELA 6.5: COLL(ASS) - Adequação

CONDIÇÕES	ALGORITMO
ASS é um arquivo de associações por item e a primeira expressão de seleção contém subexpressões todas ligadas por ET, em primeiro nível. Uma das subexpressões é do tipo C A $\geq \leq$ valor, onde A é chave primária de ASS.	COLL3
ASS é um arquivo de associações por item a a primeira expressão de seleção contém pelo menos uma subexpressão de primeiro nível,do tipo C A = valor, envolvendo atributos dos ligantes sobre os quais existe definido um arquivo de inversões (as subexpressões de primeiro nível estão todas ligadas por ET).	COLL4
ASS é um arquivo de associações por idreg e a primeira expressão de seleção contém pelo menos uma subexpressão de primeiro nível, do tipo C A = valor, envolvendo atributos do ligante sobre os quais existe definido um arquivo de inversões (as subexpressões de primeiro nível estão todas ligadas por ET).	COLL5
ASS é um arquivo de associações por idreg e a primeira expressão de seleção contém pelo menos uma subexpressão de primeiro nível envolvendo dados do ligante. As subexpressões de primeiro nível estão todas ligadas por ET. A subexpressão envonvendo dados do ligante pode conter outros conectores lógicos.	COLL6
ASS é um arquivo de associações por item.	COLL1
ASS é um arquivo de associações por idreg.	COLL2
ASS é um arquivo de associações mantido na zona intermediária ou no canal auxiliar.	COLL7

TABELA 6.6: ESTR(ARQ) - Adequação

CONDIÇÕES	ALGORITMO
ARQ possui um arquivo de inversões definido exatamente sobre os atributos mantidos.	ESTREIT1
ARQ possui um arquivo de inversões onde os "primeiros" atributos da chave de inversão são os mantidos.	ESTREIT2
ARQ é uma lista de dados classificada segundo alguma ordem dos atributos mantidos.	ESTREIT4
Nenhuma das anteriores.	ESTREIT5

TABELA 6.7: ESTRND(ARQ) - Adequação

CONDIÇÕES	ALGORITMO
ARQ possui um arquivo de inversões definido exatamente sobre os atributos mantidos.	ESTREIT1
ARQ possui um arquivo de inversões onde os "primeiros" atributos da chave de inversão são os mantidos.	ESTREIT2
Nenhuma das anteriores.	ESTREIT3

TABELA 6.8: DIF(ARQ1,ARQ2) - Adequação

CONDIÇÕES	ALGORITMO
ARQ1, ARQ2 estão classificados segundo mesmos atributos.	DIF1
ARQ1, ARQ2 são arquivos de dados com mesma chave primária.	DIF2M
ARQ1, ARQ2 estão classificados sob diferen ordens.	DIF1
Apenas ARQ1 está classificado sob alguma ordem.	DIF1
Apenas ARQ2 está classificado sob alguma ordem.	DIF1
Nem ARQ1 nem ARQ2 estão classificados.	DIF1

TABELA 6.9: UNI(ARQ1,ARQ2) - Adequação

CONDIÇÕES	ALGORITMO
ARQ1,ARQ2 estão classificados segundo mesmos atributos.	UNI1
ARQ1,ARQ2 estão classificados sob ordens diferentes.	UNI1
Apenas ARQ1 está classificado sob alguma ordem.	UNI1
Apenas ARQ2 está classificado sob alguma ordem.	UNI1
Nem ARQ1 nem ARQ2 estão classificados sob ordem alguma.	UNI2

TABELA 6.10: ISEC(ARQ1,ARQ2) - Adequação

CONDIÇÕES	ALGORITMO
ARQ1,ARQ2 estão classificados segundo mesmos atributos.	ISEC1
ARQ1,ARQ2 são arquivos de dados com mesma chave primária.	ISEC2M
ARQ1, ARQ2 estão classificados sob ordens diferentes.	ISEC1
Apenas ARQ1 está classificado sob alguma ordem.	ISEC1
Apenas ARQ2 está classificado sob alguma ordem.	ISEC1
Nem ARQ1 nem ARQ2 estão classificados.	ISEC3

TABELA 6.11: AGRUPAR(ARQ) - Adequação

CONDIÇÕES	ALGORITMO
ARQ está classificado segundo alguma ordem dos atributos de agrupamento.	AGRUPAR2
ARQ possui um arquivo de inversões definido exatamente sobre os atributos de agrupamento ou estes são "os primeiros" de uma chave de inversão sobre ARQ.	AGRUPAR1
Nenhuma das anteriores.	AGRUPAR2

TABELA 6.12: DESAGRUPAR(ASS) - Adequação

CONDIÇÕES	ALGORITMO
ASS é um arquivo de associações por item cuja chave primária é constituída por todo o ligante.	DESAGRUPAR1
ASS é um arquivo de associações por item cuja chave primária não é constituída por todo o ligante, ou ASS é um arquivo de associações por idreg.	DESAGRUPAR2
ASS é um arquivo de associações mantido na zona intermediária ou no canal auxiliar.	DESAGRUPAR3

TABELA 6.13: SUB(ARQ1,ARQ2), SUB=(ARQ1,ARQ2) - Adequação

CONDIÇÕES	ALGORITMO
ARQ1 e ARQ2 estão classificados sob mesma ordem.	SUB2
ARQ1 é uma lista de dados classificada segundo a chave primária de ARQ2, e ARQ2 é um arquivo de dados.	SUB1
ARQ1 é um arquivo de dados e ARQ2 é uma lista de dados classificada segundo a chave primária de ARQ1.	SUB4
ARQ1 e ARQ2 são arquivos de dados com mesma chave primária.	SUB3
ARQ1 e ARQ2 são listas de dados classificadas sob diferentes ordens.	SUB2
ARQ1 é uma lista de dados e ARQ2 é um arquivo de dados.	SUB1
ARQ1 é um arquivo de dados e ARQ2 é uma lista de dados.	SUB4
Nenhuma das anteriores.	SUB2

TABELA 6.14: ELEM(REGDADOS,ARQ) - Adequação

CONDIÇÕES	ALGORITMO
ARQ é um arquivo de dados	ELEM1
ARQ é uma lista de dados classificada segundo sua chave primária.	ELEM2
Nenhuma das anteriores.	ELEM3

TABELA 6.15: ELEM(REGASS,ASS) - Adequação

CONDIÇÕES	ALGORITMO
ASS é um arquivo de associações por item.	ELEM4
ASS é um arquivo de associações por idreg.	ELEM5
ASS é um arquivo de associações mantido na zona intermediária ou no canal auxiliar, e sua lista de ligantes está classificada segundo a chave primária dos ligantes.	ELEM6
Nenhuma das anteriores.	ELEM7

TABELA 6.16: JUNT(ARQ1,ARQ2) - Adequação

CONDIÇÕES	ALGORITMO
Ambos ARQ1 e ARQ2 estão classificados por uma mesma ordem dos atributos de junção.	JUNT3
Existe sobre ARQ1 e ARQ2 um arquivo de associações por item cuja chave primária é constituída exatamente pelos atributos de junção.	JUNT5
Existe sobre ARQ1 e ARQ2 um arquivo de associações por item cuja chave primária é constituída exatamente por um subconjunto dos atributos de junção.	JUNT4
ARQ1 possui um arquivo de inversões definido exatamente sobre os atributos de junção, e ARQ2 está classificado segundo esta ordem dos atributos de junção.	JUNT2A
ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de junção, e ARQ1 está classificado segundo esta ordem dos atributos de junção.	JUNT2A
ARQ1 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão, e ARQ2 está classificado segundo esta ordem dos atributos de junção.	JUNT2B
ARQ2 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão, e ARQ1 está classificado segundo esta ordem dos atributos de junção.	JUNT2B

TABELA 6.16: Continuação

CONDIÇÕES	ALGORITMO
ARQ1 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de junção, sendo este um arquivo de inversões sobre chave primária, e ARQ2 está classificado segundo esta ordem.	JUNT2A
ARQ2 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de junção, sendo este um arquivo de inversões sobre chave primária, e ARQ1 está classificado segundo esta ordem.	JUNT2A
ARQ1 e ARQ2 possuem arquivos de inversões definidos exatamente sobre os atributos de junção.	JUNT1A
ARQ1 e ARQ2 possuem arquivos de inversões onde os atributos de junção são "os primeiros" da chave de inversão.	JUNT1B
ARQ1 e ARQ2 possuem arquivos de inversões definidos exatamente sobre mesmos subconjuntos dos atributos de junção, e estes são, ambos, arquivos de inversões sobre chave primária.	JUNT1A
ARQ1 possui um arquivo de inversões definido exatamente sobre os atributos de junção.	JUNT2A
ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de junção.	JUNT2A

TABELA 6.16: Continuação

CONDIÇÕES	ALGORITMO
ARQ1 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão.	JUNT2B
ARQ2 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão.	JUNT2B
ARQ1 e ARQ2 estão classificados por diferentes ordens dos atributos de junção.	JUNT3
Apenas ARQ1 classificado segundo alguma ordem dos atributos de junção.	JUNT3
Apenas ARQ2 classificado segundo alguma ordem dos atributos de junção.	JUNT3
Nem ARQ1 nem ARQ2 classificados segundo os atributos de junção.	JUNT3

TABELA 6.17: LIG(ARQ1,ARQ2) - Adequação

CONDIÇÕES	ALGORITMO
Ambos ARQ1 e ARQ2 classificados por uma mesma ordem dos atributos de ligação.	LIG4
ARQ1 possui um arquivo de inversões definido exatamente sobre os atributos de ligação, e ARQ2 está classificado segundo esta ordem dos atributos de ligação.	LIG2A
ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de ligação, e ARQ1 está classificado segundo esta ordem dos atributos de ligação.	LIG3A
ARQ1 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão, e ARQ2 está classificado segundo esta ordem dos atributos de ligação.	LIG2B
ARQ2 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão, e ARQ1 está classificado segundo esta ordem dos atributos de ligação.	LIG3B
ARQ1 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de ligação, sendo este um arquivo de inversões sobre chave primária, e ARQ2 está classificado segundo esta ordem dos atributos de ligação.	LIG2A

TABELA 6.17: Continuação

CONDIÇÕES	ALGORITMO
ARQ2 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de ligação, sendo este um arquivo de inversões sobre chave primária, e ARQ2 está classificado segundo esta ordem dos atributos de ligação.	LIG3A
ARQ1 e ARQ2 possuem arquivos de inversões definidos exatamente sobre os atributos de ligação.	LIG1A
ARQ1 e ARQ2 possuem arquivos de inversões onde os atributos de ligação são "os primeiros" das chaves de inversão.	LIG1B
ARQ1 e ARQ2 possuem arquivos de inversões definidos exatamente sobre mesmos subconjuntos dos atributos de ligação, e estes são, ambos, arquivos de inversões sobre chave primária.	LIG1A
ARQ1 possui um arquivo de inversões definido exatamente sobre os atributos de ligação.	LIG2A
ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de ligação.	LIG3A
ARQ1 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão.	LIG2B

TABELA 6.17: Continuação

CONDIÇÕES	ALGORITMO
ARQ2 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão.	LIG3B
ARQ1 e ARQ2 estão classificados segundo diferentes ordens dos atributos de ligação.	LIG4
Apenas ARQ1 está classificado segundo alguma ordem dos atributos de ligação.	LIG4
Apenas ARQ2 está classificado segundo alguma ordem dos atributos de ligação.	LIG4
Nem ARQ1 nem ARQ2 estão classificados segundo os atributos de ligação.	LIG4

TABELA 6.18: ESTREITT(AS) - Adequação

CONDIÇÕES	ALGORITMO
AS é um registro de associação por item ou por idreg, e os atributos mantidos constituem exatamente a chave primária da tabela de ligados.	ESTREITT1
AS é um registro de associação por item ou por idreg, e os atributos mantidos são "os primeiros" da chave primária da tabela de ligados.	ESTREITT2
AS é um registro de associação por item ou por idreg, e entre os atributos mantidos está a chave primária da tabela de ligados.	ESTREITT3
AS é um registro de associação por item ou por idreg.	ESTREITT4
AS é um registro de associação mantido no canal auxiliar ou na zona intermediária e entre os atributos mantidos pelo estreitamento está a chave primária da tabela de ligados.	ESTREITT5
AS é um registro de associação mantido no canal auxiliar ou na zona intermediária, e sua tabela de ligados está classificada segundo alguma ordem dos atributos mantidos pelo estreitamento, ou os atributos mantidos são os primeiros na ordem de classificação.	ESTREITT6
Nenhuma das anteriores.	ESTREITT6

TABELA 6.19: COLECT(AS) - Adequação

CONDIÇÕES	ALGORITMO
AS é um registro de associação por item ou por idreg e a expressão seletora envolve apenas atributos da chave primária de sua tabela de ligados.	COLECT1
AS é um registro de associação por item ou por idreg.	COLECT2
AS é um registro de associação mantido na zona intermediária ou no canal auxiliar.	COLECT3

TABELA 6.20: AGRUPART(AS) - Adequação

CONDIÇÕES	ALGORITMO
AS é um registro de associação por item ou por idreg e os atributos de agrupamento constituem exatamente a chave primária da tabela de ligados, ou são "os primeiros" da chave primária da tabela de ligados.	AGRUPART1
AS é um registro de associação por item ou por idreg.	AGRUPART2
AS é um registro de associação mantido na zona intermediária ou no canal auxiliar, e os atributos de agrupamento constituem toda ou a parte inicial da chave de classificação da tabela de ligados.	AGRUPART3
AS é um registro de associação mantido na zona intermediária ou no canal auxiliar.	AGRUPART4

7. CAMINHAMENTOS NA ÁRVORE DE OPERAÇÕES

7.1 Introdução

A escolha do algoritmo mais adequado a cada operação e a escolha da ordem de saída mais interessante à operação seguinte são efetuadas através de um caminhamento "bottom-up"e, posteriormente, um caminhamento"top-down"da árvore de operações que representa uma instrução operativa em CI3. A idéia básica utilizada é proveniente de Smith e Chang em/SMI 75/.

O caminhamento"bottom-up"analisa as ordens de classificação e as estruturas de dados disponíveis em cada nodo, e decide qual o algoritmo mais adequado à execução de cada operação.

Para tanto, são necessárias informações quanto a:

a) Arquivos de inversões e arquivos de associações existentes para os arquivos de dados envolvidos.

b) Chave primária dos arquivos/listas de dados envolvidos.

c) Ordem de classificação das listas de dados envolvidas.

As informações do item (a) estão disponíveis no diretório da base de dados interna, e devem ser obtidas através de um passo preliminar, que realize o levantamento dos arquivos de inversões e de associações disponíveis (naqueles nodos em que pode ser interessante uma estrutura desse tipo) e selecione quais os arquivos de inversões/associações a serem utilizados na resolução das operações (a etapa de seleção é peculiar a cada operação, e os critérios a utilizar não serão abordados neste trabalho).

As informações do item (b) já estão disponíveis na própria árvore de código intermediário 3. As ordens de classificação das listas de dados que constituem folhas da árvore de operações (mantidas no canal auxiliar) estão disponíveis no diretório da base de dados interna, enquanto que as opções de ordem de classificação possíveis para os resultados intermediários (nodos não folhas) são fornecidos durante a etapa de caminhamento "bottom-up".

Escolhido o algoritmo a ser usado na resolução de uma certa operação, é possível determinar quais as ordens de saída que o mesmo pode oferecer. Por exemplo, se é escolhido o algoritmo UNI2 para realizar uma operação de união, o mesmo pode fornecer seu resultado classificado por qualquer ordem dos atributos que constituem os registros de dados envolvidos, já que efetua uma classificação sobre os dois operandos, para eliminar duplicatas.

O caminhamento "top-down" da árvore de operações tem por objetivo decidir qual a ordem de classificação mais interessante como saída de cada operação.

Inicialmente, é escolhida a ordem de saída para a raiz da árvore de operações (esta deverá ser a ordem mais interessante ao algoritmo escolhido). A partir de então, é possível decidir qual a ordem de entrada preferencial para o algoritmo a ser executado pelo nodo em análise. Obviamente, esta ordem de entrada deve ser a ordem de saída preferencial para o algoritmo a ser executado antes deste, e assim por diante.

Em muitos casos, não é possível que um certo algoritmo forneça seu resultado na ordem de saída solicitada pelo algoritmo seguinte. Nesta situação, é adotada a opção de ordem indiferente, que permite a decisão da ordem de saída pelo próprio algoritmo, da forma como lhe seja mais interessante.

7.2 Situação de um nodo na árvore de operações

7.2.1 Operações unárias

São consideradas operações unárias aquelas operações que utilizam um único operando. As operações unárias estudadas foram: COLR, COLRU, COLA, COLAU, COLL, ESTR, ESTRND, AGRUPAR, DESAGRUPAR, ELEM, COLECT, ESTREITT e AGRUPART.

Sendo ARQ o operando de uma operação unária, duas situações podem ocorrer:

- ARQ é uma folha. Neste caso, ARQ é uma construção armazenada na base de dados ou no canal auxiliar. Se ARQ for uma construção da base de dados, poderá dispor de arquivos de inversões ou arquivos de associações (dependendo da operação envolvida); se estiver no canal auxiliar, é uma construção resultante de outra instrução operativa (podendo apresentar uma ordem de classificação conhecida).

- ARQ não é uma folha. Neste caso, o operando ARQ é um resultado intermediário, proveniente da execução de outra operação. Se for proveniente da execução de uma operação que produz como resultado uma tabela (relacional ou ligacional), estará sob a forma de lista (ou listas, no caso das tabelas ligacionais) de dados, cuja ordem de classificação pode ser conhecida.

No caso dos operandos constituídos por arquivos de associações mantidos na zona intermediária ou no canal auxiliar, é interessante recordar que os mesmos são constituídos por duas listas de dados, uma representando ligantes e outra representando ligados das associações. Tanto listas de ligantes como listas de ligados podem estar classifica-

das segundo alguma ordem conhecida. Já que o acesso a uma lista de ligados é feito, sempre, a partir da lista de ligantes correspondente, só é levada em consideração a ordem de classificação de cada grupo de ligados de uma mesma associação.

7.2.2 Operações binárias

São consideradas operações binárias aquelas operações que utilizam dois operandos. As operações binárias estudadas foram: SUB, SUB=, UNI, ISEC, DIF, JUNT e LIG.

Se ARQ1 e ARQ2 representam os dois operandos de uma operação binária, as seguintes situações podem ocorrer:

- ARQ1 e ARQ2 são folhas.
- Apenas ARQ1 é folha.
- Apenas ARQ2 é folha.
- Nem ARQ1 nem ARQ2 são folhas.

Neste caso, as características de cada operando (folha ou não-folha) obedecem ao mesmo detalhamento dos operandos de operações unárias. Por exemplo, no caso de uma operação LIGAR onde ambos os operandos são folhas, podem estar disponíveis arquivos de inversões sobre ARQ1 e ARQ2, definidos sobre os atributos de ligação, que permitirão uma execução mais rápida dessa operação.

7.3 Caminhamento "bottom-up"

7.3.1 Princípio básico

O caminhamento "bottom-up" da árvore de operações é feito analisando-se a adequação dos algoritmos a cada operação e escolhendo-se o algoritmo mais conveniente (ver tabelas 6.2 a 6.20), determinando-se, ao mesmo tempo, as opções de ordem de classificação que este algoritmo oferece.

São levados em conta:

1º) aproveitamento da ordem de classificação dos operandos envolvidos;
2º) aproveitamento das estruturas de dados existentes sobre os mesmos.

Só em último caso são escolhidos algoritmos necessitando efetuar classificações, devido ao número de operações de entrada e saída e ao espaço de armazenamento exigido para uma operação de classificação.

7.3.2 Estrutura das tabelas utilizadas

Cada operação é apresentada em uma tabela contendo as seguintes informações:

- condições a serem analisadas (CONDIÇÕES);
- algoritmo escolhido (ALGORITMO);
- opções de ordem de saída oferecidas (OPÇÕES DE ORDEM);
- situação dos operandos na árvore de operações.

Para as operações ligacionais, a coluna OPÇÕES DE ORDEM se encontra subdividida em LIGANTES (contendo as opções de ordem de saída para a lista de ligantes produzida) e LIGADOS (contendo as opções de ordem de saída para cada grupo de ligados associado a um mesmo ligante, na lista de ligados).

Algumas das operações estudadas não devolvem,como resultado, listas de dados, e sim um valor booleano(caso das operações SUB, SUB= e ELEM). Outras devolvem apenas um ou nenhum registro de dados ou associação (operações COLRU e COLAU). Para tais operações, a coluna OPÇÕES DE ORDEM foi deixada em branco.

A tabela 7.1 mostra os termos utilizados como opções de ordem de saída e seu significado (valem aqui, também,os termos utilizados no capítulo 6, nas tabelas de adequação dos algoritmos).

TABELA 7.1: Termos utilizados nas tabelas de caminhamento "bottom-up"

TERMO	SIGNIFICADO
T	Tabela de ligados de uma associação.
LIGANTES	Lista de ligantes mantida na zona intermediária ou canal auxiliar.
Or_{ARQ}	Ordem de classificação de ARQ (quando ARQ é folha).
$CP_{INV_{ARQ}}$	Ordem de classificação segundo a chave primária do arquivo de inversões sobre ARQ que está sendo utilizado por um determinado algoritmo (nas operações unárias é dada, simplesmente, por CP_{INV}).
CP_{ASS}	Ordem de classificação segundo a chave primária do arquivo de associações ASS (esse termo só é utilizado nos casos em que ASS é um arquivo de associações por item ou por idreg).
CP_T	Ordem de classificação segundo a chave primária da tabela de ligados de uma associação envolvida na operação.
Or_T	Ordem de classificação segundo a tabela de ligados de uma associação envolvida na operação (usado quando esta ordem não é conhecida de antemão).

TABELA 7.1: Continuação

TERMO	SIGNIFICADO
$Or_{LIGANTES}$	Ordem de classificação da lista de dados de ligantes usada como entrada para a operação.
$Adic_{ARQ}$	Opção de ordem definida apenas para as operações AGRUPAR e LIGAR; diz respeito à ordem de classificação das tabelas de ligados produzidas por tais operações (ver algoritmos para as operações AGRUPAR e LIGAR).
Não	Nenhuma ordem de classificação.
C_{ARQ}	Conjunto das opções de ordem oferecidas pela operação que gera ARQ (no caso de ARQ representar um nodo não-folha da árvore de operações).
P(X)	Conjunto das permutações possíveis do conjunto X.
X	Pode ser um dos seguintes conjuntos: M_{ARQ} - conjunto dos atributos mantidos por uma operação ESTR(ARQ) ou ESTRND(ARQ). M_T - conjunto dos atributos mantidos por uma operação ESTREITT(AS), sendo T a tabela de ligados da associação AS. A_{ARQ} - conjunto dos atributos de agrupamento para uma operação AGRUPAR (ARQ). A_T - conjunto dos atributos de agrupamento para uma operação AGRUPART (AS), sendo T a tabela de ligados da associação AS.

TABELA 7.1: Continuação

TERMO	SIGNIFICADO
	ATR_{ARQ} - conjunto dos atributos que constituem os campos de um registro de ARQ.
	L - conjunto dos atributos de ligação para uma operação LIGAR (ARQ1, ARQ2).
	J - conjunto dos atributos de junção para uma operação JUNTAR (ARQ1, ARQ2).

São ainda usadas as formas compactadas C(L), para representar a expressão $[C_{ARQ} \cap P(L)]$, ou C(J), para representar $[C_{ARQ} \cap P(J)]$, onde ARQ representa um dos operandos ARQ1 ou ARQ2 de uma operação JUNT(ARQ1,ARQ2) ou LIG (ARQ1, ARQ2), que significam: "o conjunto de opções de ordem onde os 'primeiros' atributos são os atributos de ligação/junção, entre todas as opções de ordem oferecidas pela operação que gera ARQ".

Algumas das operações merecem atenção especial quanto às opções de ordem de saída que oferecem, a saber:

AGRUPAR(ARQ) -

No caso de ser escolhido o algoritmo AGRUPAR1, a ordem de saída dos ligantes é dada por CP_{INV}. Como os ligantes provenientes de uma operação AGRUPAR são constituídos, apenas, dos atributos de agrupamento, quando for utilizado um arquivo de inversões onde os atributos de agrupamento são "os primeiros" componentes da chave de inversão, a ordem de saída, na realidade, é dada pelos atributos de agrupamento mantidos nessa chave de inversão.

AGRUPART(AS) -

No caso de ser escolhido o algoritmo AGRUPART2, a ordem de saída dos ligantes é dada por CP_T. Como esses ligantes são constituídos, apenas, dos atributos de agrupamento mantidos na chave primária da tabela de ligados, então a ordem de saída, na realidade, é dada pelos atributos de agrupamento mantidos na chave primária da tabela de ligados da associação AS.

LIG(ARQ1,ARQ2) -

No caso de os atributos de ligação serem "os primeiros" de uma chave de inversão sobre ARQ1 ou ARQ2, a opção de ordem de saída dada por $CP_{INV_{ARQ1}}$ ou $CP_{INV_{ARQ2}}$, efetivamente, se constitui no conjunto de atributos de ligação utilizados, dentro desta chave de inversão.

JUNT(ARQ1,ARQ2) -

No caso de os atributos de junção serem "os primeiros" de uma chave de inversão sobre ARQ1 ou ARQ2, a opção de ordem de saída dada por $CP_{INV_{ARQ1}}$ ou $CP_{INV_{ARQ2}}$, efetivamente, pode-se constituir no conjunto de atributos que virão a fazer parte dos registros resultantes da junção e que pertencem à chave de inversão.

ESTR(ARQ)
ESTRND(ARQ) -

Quando escolhido o algoritmo ESTREIT2, a ordem de saída CP_{INV} deve representar apenas os atributos mantidos pelo estreitamento ocorrentes nesta chave de inversão sobre ARQ, e não toda a chave, já que os demais atributos que a compõem não aparecerão no resultado dessa operação.

ESTREITT(ARQ) -

Quando escolhido o algoritmo ESTREITT2, a ordem de saída CP_T deve representar apenas os atributos mantidos pelo ESTREITT ocorrentes na chave primária dessa tabela de ligados, já que os demais atributos que a compõem não apare

cerão no resultado dessa operação.

7.3.3 Tabelas de caminhamento"bottom-up"

As tabelas 7.2 a 7.20 mostram as regras de caminhamento"bottom-up"da árvore de operações, para as operações estudadas. A leitura das mesmas deve ser feita de cima para baixo, utilizando os mesmos critérios já descritos no capítulo 6, item 6.3.1, conforme as tabelas de adequação.

TABELA 7.2: COLR(ARQ), COLRU(ARQ) - Caminhamento "bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ARQ É FOLHA	ARQ possui um arquivo de inversões definido sobre atributos que estão envolvidos em uma subexpressão de seleção, e as subexpressões de seleção estão todas ligadas por ET(em primeiro nível).	COLR2	CP_{INV}
	Em caso contrário	COLR1	Or_{ARQ}
ARQ NÃO É FOLHA		COLR1	C_{ARQ}

TABELA 7.3: COLA(ASS) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
ASS É	ASS é um arquivo de associações por item e pelo menos uma das subexpressões de primeiro nível, que estão todas ligadas por ET, é do tipo C A $\geq$ valor, onde A é chave primária de ASS.	COLA2	CP_{ASS}	CP_T
FOLHA	ASS é um arquivo de associações por item ou por idreg, e existe pelo menos uma subexpressão de primeiro nível, do tipo C A = valor envolvendo atributos dos ligantes sobre os quais existe definido um arquivo de inversões (as subexpressões de primeiro nível estão todas ligadas por ET).	COLA3	CP_{INV}	CP_T
	ASS é um arquivo de associações por idreg e existe pelo menos uma subexpressão de primeiro nível envolvendo dados do ligante. As subexpressões de primeiro nível estão todas ligadas por ET. A subexpressão envolvendo dados do ligante pode conter outros conectores lógicos.	COLA4	Não (nenhuma ordem)	CP_T
	ASS é um arquivo de associações mantido no canal auxiliar.	COLA5	$Or_{ligantes}$	Or_T
	Nenhuma das anteriores.	COLA1	CP_{ASS}	CP_T

TABELA 7.3: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
ASS NÃO É FOLHA	(Observação - Neste caso, ASS é um arquivo de associações mantido na zona intermediária.)	COLA5	C_{ASS}	C_T

TABELA 7.4: COLAU(ASS) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ASS	ASS é um arquivo de associações por item.	COLA2	-
É	ASS é um arquivo de associações por idreg.	COLA3	-
FOLHA	ASS é um arquivo de associações mantido no canal auxiliar.	COLA5	-
ASS NÃO É FOLHA	(Observação - Neste caso, ASS é um arquivo de associações mantido na zona intermediária.)	COLA5	-

TABELA 7.5: COLL(ASS) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ASS É	ASS é um arquivo de associações por item e a primeira expressão de seleção contém subexpressões todas ligadas por ET, em primeiro nível. Uma das subexpressões é do tipo C A $\stackrel{>}{<}$ valor, onde A é chave primária de ASS.	COLL3	CP_{ASS}
FOLHA	ASS é um arquivo de associações por item e a primeira expressão de seleção contém pelo menos uma subexpressão de primeiro nível, do tipo C A = valor, envolvendo atributos dos ligantes sobre os quais existe definido um arquivo de inversões (as subexpressões de primeiro nível estão todas ligadas por ET).	COLL4	CP_{ASS}
	ASS é um arquivo de associações por idreg e a primeira expressão de seleção contém pelo menos uma subexpressão de primeiro nível, do tipo C A = valor, envolvendo atributos do ligante sobre os quais existe definido um arquivo de inversões (as subexpressões de primeiro nível estão todas ligadas por ET).	COLL5	$P(ATR_T)$

TABELA 7.5: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
	ASS é um arquivo de associações por idreg e a primeira expressão de seleção contém pelo menos uma subexpressão de primeiro nível envolvendo dados do ligante. As subexpressões de primeiro nível estão todas ligadas por ET. A subexpressão envolvendo dados do ligante pode conter outros conectores lógicos.	COLL6	$P(ATR_T)$
	ASS é um arquivo de associações por item.	COLL1↑	CP_{ASS}
	ASS é um arquivo de associações por idreg.	COLL2	$P(ATR_T)$
	ASS é um arquivo de associações mantido no canal auxiliar.	COLL7	$P(ATR_T)$
ASS NÃO É FOLHA	ASS é um arquivo de associações mantido na zona intermediária.	COLL7	$P(ATR_T)$

TABELA 7.6: ESTR(ARQ) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ARQ É FOLHA	ARQ possui um arquivo de inversões definido exatamente sobre os atributos mantidos.	ESTREIT1	CP_{INV}
	ARQ possui um arquivo de inversões onde "os primeiros" atributos da chave de inversão são os mantidos.	ESTREIT2	CP_{INV}
	ARQ é uma lista de dados classificada segundo alguma ordem dos atributos mantidos.	ESTREIT4	Or_{ARQ}
	Nenhuma das anteriores.	ESTREIT5	$P(M_{ARQ})$
ARQ NÃO É FOLHA	ARQ é uma lista de dados classificada segundo alguma ordem dos atributos mantidos.	ESTREIT4	$C_{ARQ} \cap P(M_{ARQ})$
	Em caso contrário	ESTREIT5	$P(M_{ARQ})$

TABELA 7.7: ESTRND(ARQ) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ARQ É FOLHA	ARQ possui um arquivo de inversões definido exatamente sobre os atributos mantidos.	ESTREIT1	CP_{INV}
	ARQ possui um arquivo de inversões onde "os primeiros" atributos da chave de inversão são os mantidos.	ESTREIT2	CP_{INV}
	Nenhuma das anteriores.	ESTREIT3	Or_{ARQ}
ARQ NÃO É FOLHA	(Nesse caso, ARQ é uma lista de dados qualquer)	ESTREIT3	C_{ARQ}(todas as opções de ordem oferecidas pela operação anterior)

TABELA 7.8: DIF(ARQ1,ARQ2) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ARQ1 E ARQ2 SÃO FOLHAS	ARQ1 e ARQ2 são listas de dados classificadas sob mesmos atributos.	DIF1	Or_{ARQ1} (= Or_{ARQ2})
	ARQ1 e ARQ2 são arquivos de dados com mesma chave primária.	DIF2M	CP_{INV}
	ARQ1 e ARQ2 são listas classificadas sob ordens diferentes.	DIF1	Or_{ARQ1}, Or_{ARQ2}
	Apenas ARQ1 está classificado sob alguma ordem.	DIF1	Or_{ARQ1}
	Apenas ARQ2 está classificado sob alguma ordem.	DIF1	Or_{ARQ2}
	Nem ARQ1 nem ARQ2 estão classificados.	DIF1	$P(ATR_{ARQ1})$ (=$P(ATR_{ARQ2})$)
CASO CONTRÁRIO*	ARQ1 e ARQ2 são listas de dados classificadas sob mesmos atributos.	DIF1	$C_{ARQ1} \cap C_{ARQ2}$
	ARQ1 e ARQ2 são listas classificadas sob ordens diferentes.	DIF1	$C_{ARQ1} \cup C_{ARQ2}$
	Apenas ARQ1 está classificado sob alguma ordem.	DIF1	C_{ARQ1} (Se ARQ1 for folha, é oferecida apenas Or_{ARQ1})
	Apenas ARQ2 está classificado sob alguma ordem.	DIF1	C_{ARQ2} (Se ARQ2 for folha, é oferecida apenas Or_{ARQ2})
	Nem ARQ1 nem ARQ2 estão classificados.	DIF1	$P(ATR_{ARQ1})$ (=$P(ATR_{ARQ2})$)

* Ou apenas ARQ1 é folha, ou apenas ARQ2 é folha, ou nem ARQ1 nem ARQ2 são folhas.

TABELA 7.9: UNI(ARQ1,ARQ2) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
	ARQ1 e ARQ2 estão classificados sob mesma ordem.	UNI1	$C_{ARQ1} \cap C_{ARQ2}$
	ARQ1 e ARQ2 estão classificados sob ordens diferentes.	UNI1	$C_{ARQ1} \cup C_{ARQ2}$
	Apenas ARQ1 está classificado sob alguma ordem.	UNI1	C_{ARQ1} (Se folha, a ordem oferecida é apenas Or_{ARQ1})
*	Apenas ARQ2 está classificado sob alguma ordem.	UNI1	C_{ARQ2} (Se folha, a ordem oferecida é apenas Or_{ARQ2})
	Nem ARQ1 nem ARQ2 estão classificados.	UNI2	$P(ATR_{ARQ1})(=P(ATR_{ARQ2}))$

* Todas as situações possíveis quanto às posições de ARQ1 e ARQ2 são resolvidas da mesma forma (como detalhado acima).

TABELA 7.10: ISEC(ARQ1,ARQ2) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ARQ1 E ARQ2 SÃO FOLHAS	ARQ1 e ARQ2 são listas de dados classificadas sob mesmos atributos.	ISEC1	Or_{ARQ1} (=Or_{ARQ2})
	ARQ1 e ARQ2 são arquivos de dados com mesma chave primária.	ISEC2M	CP_{INV}
	ARQ1 e ARQ2 são listas classificadas sob ordens diferentes.	ISEC1	Or_{ARQ1}, Or_{ARQ2}
	Apenas ARQ1 está classificado segundo alguma ordem.	ISEC1	Or_{ARQ1}
	Apenas ARQ2 está classificado segundo alguma ordem.	ISEC1	Or_{ARQ2}
	Nem ARQ1 nem ARQ2 estão classificados.	ISEC3	$P(ATR_{ARQ1})$(=$P(ATR_{ARQ2})$)
CASO CONTRÁRIO*	ARQ1 e ARQ2 são listas de dados classificadas sob mesmos atributos.	ISEC1	$C_{ARQ1} \cap C_{ARQ2}$
	ARQ1 e ARQ2 são listas classificadas sob ordens diferentes.	ISEC1	$C_{ARQ1} \cup C_{ARQ2}$
	Apenas ARQ1 está classificado segundo alguma ordem.	ISEC1	C_{ARQ1} (Se ARQ1 for folha, é oferecida apenas Or_{ARQ1})
	Apenas ARQ2 está classificado segundo alguma ordem.	ISEC1	C_{ARQ2} (Se ARQ_2 for folha, é oferecida apenas Or_{ARQ2})
	Nem ARQ1 nem ARQ2 estão classificados.	ISEC3	$P(ATR_{ARQ1})$(=$P(ATR_{ARQ2})$)

* Ou apenas ARQ1 é folha, ou apenas ARQ2 é folha, ou nem ARQ1 nem ARQ2 são folhas.

TABELA 7.11: AGRUPAR(ARQ) - Caminhamento "bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
ARQ É FOLHA	ARQ possui um arquivo de inversões sobre os atributos de agrupamento ou estes são "os primeiros" de uma chave de inversão sobre ARQ.	AGRUPAR1	CP_{INV}	$Adic_{ARQ}$
	ARQ é uma lista de dados classificada segundo alguma ordem dos atributos de agrupamento.	AGRUPAR2	Or_{ARQ}	$Adic_{ARQ}$
	Nenhuma das anteriores.	AGRUPAR2	$P(A_{ARQ})$	Não (nenhuma ordem)
ARQ NÃO É FOLHA	ARQ está classificado segundo alguma ordem dos atributos de agrupamento.	AGRUPAR2	$C_{ARQ} \cap P(A_{ARQ})$	$Adic_{ARQ}$
	Em caso contrário	AGRUPAR2	$P(A_{ARQ})$	Não (nenhuma ordem)

TABELA 7.12: DESAGRUPAR(ASS) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ASS É FOLHA	ASS é um arquivo de associações por item cuja chave primária é constituída por todo o ligante.	DESAGRUPAR1	CP_{ASS}
	ASS é um arquivo de associações por item cuja chave primária não é constituída por todo o ligante, ou ASS é um arquivo de associações por idreg.	DESAGRUPAR2	CP_{ASS}
	ASS é um arquivo de associações mantido no canal auxiliar.	DESAGRUPAR3	$Or_{LIGANTES}$
ASS NÃO É FOLHA	(Nesse caso, ASS é um arquivo de associações mantido na zona intermediária).	DESAGRUPAR3	$C_{LIGANTES}$

TABELA 7.13: SUB(ARQ1,ARQ2), SUB=(ARQ1,ARQ2) - Caminhamento "bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ARQ1 E ARQ2 SÃO FOLHAS	ARQ1 e ARQ2 estão classificados sob mesma ordem.	SUB2	-
	ARQ1 é uma lista de dados classificada segundo a chave primária de ARQ2, e ARQ2 é um arquivo de dados.	SUB1	-
	ARQ1 é um arquivo de dados e ARQ2 é uma lista de dados classificada segundo a chave primária de ARQ1.	SUB4	-
	ARQ1 e ARQ2 são arquivos de dados com mesma chave primária.	SUB3	-
	ARQ1 e ARQ2 estão classificados sob diferentes ordens.	SUB2	-
	ARQ1 é uma lista de dados e ARQ2 é um arquivo de dados.	SUB1	-
	ARQ1 é um arquivo de dados e ARQ2 é uma lista de dados.	SUB4	-
	Nenhuma das anteriores.	SUB2	-

TABELA 7.13: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
APE-NAS	ARQ1 e ARQ2 estão classificados sob mesma ordem.	SUB2	-
ARQ1 É FOLHA	ARQ1 é um arquivo de dados e ARQ2 é uma lista de dados classificada segundo a chave primária de ARQ1.	SUB4	-
	ARQ1 e ARQ2 estão classificados sob diferentes ordens.	SUB2	-
	ARQ1 é um arquivo de dados e ARQ2 é uma lista de dados.	SUB4	-
	Nenhuma das anteriores.	SUB2	-
APE-NAS	ARQ1 e ARQ2 estão classificados sob mesma ordem.	SUB2	-
ARQ2 É FOLHA	ARQ1 é uma lista de dados classificada segundo a chave primária de ARQ2, e ARQ2 é um arquivo de dados.	SUB1	-
	ARQ1 e ARQ2 estão classificados sob diferentes ordens.	SUB2	-
	ARQ1 é uma lista de dados e ARQ2 é um arquivo de dados.	SUB1	-
	Nenhuma das anteriores.	SUB2	-
NEM ARQ1 NEM ARQ2 SÃO FO-LHAS		SUB2	-

TABELA 7.14: ELEM(REGDADOS,ARQ) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ARQ É FOLHA	ARQ é um arquivo de dados.	ELEM1	-
	ARQ é uma lista de dados classificada segundo sua chave primária.	ELEM2	-
	Nenhuma das anteriores.	ELEM3	-
ARQ NÃO É FOLHA	ARQ é uma lista de dados classificada segundo sua chave primária.	ELEM2	-
	Caso contrário	ELEM3	-

TABELA 7.15: ELEM(REGASS,ASS) - Caminhamento "bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ASS É FOLHA	ASS é um arquivo de associações por item.	ELEM4	-
	ASS é um arquivo de associações por idreg.	ELEM5	-
	ASS é um arquivo de associações mantido no canal auxiliar.	ELEM6	-
ASS NÃO É FOLHA	(ASS é um arquivo de associações mantido na zona intermediária).	ELEM6	-

TABELA 7.16: JUNT(ARQ1,ARQ2) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ARQ1 E ARQ2 SÃO FOLHAS	Ambos ARQ1 e ARQ2 classificados por uma mesma ordem dos atributos de junção.	JUNT3	Or_{ARQ1} (=Or_{ARQ2})
	Existe sobre ARQ1 e ARQ2 um arquivo de associações por item cuja chave primária é constituída exatamente pelos atributos de junção.	JUNT5	CP_{ASS}
	Existe sobre ARQ1 e ARQ2 um arquivo de associações por item cuja chave primária é constituída exatamente por um subconjunto dos atributos de junção.	JUNT4	CP_{ASS}
	ARQ1 possui um arquivo de inversões definido exatamente sobre os atributos de junção, e ARQ2 está classificado segundo esta ordem dos atributos de junção.	JUNT2A	Or_{ARQ2}
	ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de junção e ARQ1 está classificado segundo esta ordem dos atributos de junção.	JUNT2A	Or_{ARQ1}
	ARQ1 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão, e ARQ2 está classificado segundo esta ordem dos atributos de junção.	JUNT2B	Or_{ARQ2}

TABELA 7.16: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ARQ1 E ARQ2 SÃO FOLHAS	ARQ2 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão, e ARQ1 está classificado segundo esta ordem dos atributos de junção.	JUNT2B	Or_{ARQ1}
	ARQ1 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de junção, sendo este um arquivo de inversões sobre chave primária, e ARQ2 está classificado segundo esta ordem.	JUNT2A	Or_{ARQ2}
	ARQ2 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de junção, sendo este um arquivo de inversões sobre chave primária, e ARQ1 está classificado segundo esta ordem.	JUNT2A	Or_{ARQ1}
	ARQ1 e ARQ2 possuem arquivos de inversões definidos exatamente sobre os atributos de junção.	JUNT1A	$CP_{INV_{ARQ1}}$
	ARQ1 e ARQ2 possuem arquivos de inversões onde os atributos de junção são "os primeiros" das chaves de inversão.	JUNT1B	$CP_{INV_{ARQ1}}$
	ARQ1 e ARQ2 possuem arquivos de inversões definidos exatamente sobre mesmos subconjuntos dos atributos de junção, e estes são, ambos, arquivos de inversões sobre chave primária.	JUNT1A	$CP_{INV_{ARQ1}}$ $(=CP_{INV_{ARQ2}})$

TABELA 7.16: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
ARQ1 E ARQ2 SÃO FOLHAS	ARQ1 possui um arquivo de inversões exatamente sobre os atributos de junção.	JUNT2A	Or_{ARQ2}
	ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de junção.	JUNT2A	Or_{ARQ1}
	ARQ1 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão.	JUNT2B	Or_{ARQ2}
	ARQ2 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão.	JUNT2B	Or_{ARQ1}
	ARQ1 e ARQ2 estão classificados por diferentes ordens dos atributos de junção.	JUNT3	Or_{ARQ1}, Or_{ARQ2}
	Apenas ARQ1 está classificado segundo alguma ordem dos atributos de junção.	JUNT3	Or_{ARQ1}
	Apenas ARQ2 está classificado segundo alguma ordem dos atributos de junção.	JUNT3	Or_{ARQ2}
	Nem ARQ1 nem ARQ2 classificados segundo os atributos de junção.	JUNT3	$P(J_{ARQ1}) = P(J_{ARQ2})$

TABELA 7.16: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
APE- NAS ARQ1 É FOLHA	Ambos ARQ1 e ARQ2 classificados por uma mesma ordem dos atributos de junção.	JUNT3	Or_{ARQ1}
	ARQ1 possui um arquivo de inversões definido exatamente sobre os atributos de junção, e ARQ2 está classificado segundo esta ordem dos atributos de junção.	JUNT2A	$CP_{INV_{ARQ1}}$
	ARQ1 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave primária de inversão, e ARQ2 está classificado segundo esta ordem dos atributos de junção.	JUNT2B	$CP_{INV_{ARQ1}}$
	ARQ1 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de junção, sendo este um arquivo de inversões sobre chave primária, e ARQ2 está classificado segundo esta ordem.	JUNT2A	$CP_{INV_{ARQ1}}$
	ARQ1 possui um arquivo de inversões definido exatamente sobre os atributos de junção.	JUNT2A	C_{ARQ2}
	ARQ1 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão.	JUNT2B	C_{ARQ2}

TABELA 7.16: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
APENAS ARQ1 É FOLHA	ARQ1 e ARQ2 estão classificados por diferentes ordens dos atributos de junção.	JUNT3	$\{Or_{ARQ1}\} \cup C(J_{ARQ2})$
	Apenas ARQ1 está classificado segundo alguma ordem dos atributos de junção.	JUNT3	Or_{ARQ1}
	Apenas ARQ2 está classificado segundo alguma ordem dos atributos de junção.	JUNT3	$C(J_{ARQ2})$
	Nem ARQ1 nem ARQ2 estão classificados segundo os atributos de junção.	JUNT3	$P(J_{ARQ1})(=P(J_{ARQ2}))$
APENAS ARQ2 É FOLHA	Ambos ARQ1 e ARQ2 classificados por uma mesma ordem dos atributos de junção.	JUNT3	Or_{ARQ2}
	ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de junção, e ARQ1 está classificado segundo esta ordem dos atributos de junção.	JUNT2A	$CP_{INV_{ARQ2}}$
	ARQ2 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão e ARQ1 está classificado segundo esta ordem dos atributos de junção.	JUNT2B	$CP_{INV_{ARQ2}}$

TABELA 7.16: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
APE- NAS ARQ2 É FOLHA	ARQ2 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de junção, sendo este um arquivo de inversões sobre chave primária, e ARQ1 está classificado segundo esta ordem.	JUNT2A	$CP_{INV_{ARQ2}}$
	ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de junção.	JUNT2A	C_{ARQ1}
	ARQ2 possui um arquivo de inversões onde os atributos de junção são "os primeiros" da chave de inversão.	JUNT2B	C_{ARQ1}
	ARQ1 e ARQ2 estão classificados por diferentes ordens dos atributos de junção.	JUNT3	$C(J_{ARQ1}) \cup \{Or_{ARQ2}\}$
	Apenas ARQ1 está classificado segundo alguma ordem dos atributos de junção.	JUNT3	$C(J_{ARQ1})$
	Apenas ARQ2 está classificado segundo alguma ordem dos atributos de junção.	JUNT3	Or_{ARQ2}
	Nem ARQ1 nem ARQ2 estão classificados segundo os atributos de junção.	JUNT3	$P(J_{ARQ1})$ $(=P(J_{ARQ2}))$

TABELA 7.16: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
NEM ARQ1 NEM ARQ2 SÃO FOLHAS	Ambos ARQ1 e ARQ2 classificados por uma mesma ordem dos atributos de junção.	JUNT3	$C(J_{ARQ1}) \cap C(J_{ARQ2})$
	ARQ1 e ARQ2 estão classificados por diferentes ordens dos atributos de junção.	JUNT3	$C(J_{ARQ1}) \cup C(J_{ARQ2})$
	Apenas ARQ1 está classificado segundo alguma ordem dos atributos de junção.	JUNT3	$C(J_{ARQ1})$
	Apenas ARQ2 está classificado segundo alguma ordem dos atributos de junção.	JUNT3	$C(J_{ARQ2})$
	Nem ARQ1 nem ARQ2 estão classificados segundo os atributos de junção.	JUNT3	$P(J_{ARQ1})$ $(=P(J_{ARQ2}))$

TABELA 7.17: LIG(ARQ1,ARQ2) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
ARQ1 E ARQ2 SÃO FOLHAS	Ambos ARQ1 e ARQ2 classificados por uma mesma ordem dos atributos de ligação.	LIG4	Or_{ARQ1}	$Adic_{ARQ2}$
	ARQ1 possui um arquivo de inversões definido exatamente sobre os atributos de ligação,e ARQ2 está classificado segundo esta ordem dos atributos de ligação.	LIG2A	$CP_{INV_{ARQ1}}$	$Adic_{ARQ2}$
	ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de ligação, e ARQ1 está classificado segundo esta ordem dos atributos de ligação.	LIG3A	Or_{ARQ1}	Não (Nenhuma ordem)
	ARQ1 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão, e ARQ2 está classificado segundo esta ordem dos atributos de ligação.	LIG2B	$CP_{INV_{ARQ1}}$	$Adic_{ARQ2}$
	ARQ2 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão, e ARQ1 está classificado segundo esta ordem dos atributos de ligação.	LIG3B	Or_{ARQ1}	$Adic_{ARQ2}$

TABELA 7.17: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
ARQ1 E ARQ2 SÃO FOLHAS	ARQ1 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de ligação, sendo este um arquivo de inversões sobre chave primária, e ARQ2 está classificado segundo esta ordem.	LIG2A	$CP_{INV_{ARQ1}}$	$Adic_{ARQ2}$
	ARQ2 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de ligação, sendo este um arquivo de inversões sobre chave primária, e ARQ1 está classificado segundo esta ordem.	LIG3A	Or_{ARQ1}	$Adic_{ARQ2}$
	ARQ1 e ARQ2 possuem arquivos de inversões definidos exatamente sobre os atributos de ligação.	LIG1A	$CP_{INV_{ARQ1}}$	Não (nenhuma ordem)
	ARQ1 e ARQ2 possuem arquivos de inversões onde os atributos de ligação são "os primeiros" da chave de inversão.	LIG1B	$CP_{INV_{ARQ1}}$	$Adic_{ARQ2}$
	ARQ1 e ARQ2 possuem arquivos de inversões definidos sobre mesmos subconjuntos dos atributos de ligação, e estes são arquivos de inversões sobre chave primária.	LIG1A	$CP_{INV_{ARQ1}}$	Não (nenhuma ordem)
	ARQ1 possui um arquivo de inversões exatamente sobre os atributos de ligação.	LIG2A	$CP_{INV_{ARQ1}}$	Não (nenhuma ordem)

TABELA 7.17: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
ARQ1 E ARQ2 SÃO FOLHAS	ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de ligação.	LIG3A	Or_{ARQ1}	Não (nenhuma ordem)
	ARQ1 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão.	LIG2B	$CP_{INV_{ARQ1}}$	Não (nenhuma ordem)
	ARQ2 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão.	LIG3B	Or_{ARQ1}	$Adic_{ARQ2}$
	ARQ1 e ARQ2 estão classificados segundo diferentes ordens dos atributos de ligação.	LIG4	Or_{ARQ1}, Or_{ARQ2}	$Adic_{ARQ2}$
	Apenas ARQ1 está classificado segundo alguma ordem dos atributos de ligação.	LIG4	Or_{ARQ1}	Não (nenhuma ordem)
	Apenas ARQ2 está classificado segundo alguma ordem dos atributos de ligação.	LIG4	Or_{ARQ2}	$Adic_{ARQ2}$
	Nem ARQ1 nem ARQ2 classificados segundo os atributos de ligação.	LIG4	$P(L_{ARQ1})$ $(=P(L_{ARQ2}))$	Não (nenhuma ordem)

TABELA 7.17: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
APE- NAS ARQ1 É FOLHA	Ambos ARQ1 e ARQ2 classificados por uma mesma ordem dos atributos de ligação.	LIG4	Or_{ARQ1}	$Adic_{ARQ2}$
	ARQ1 possui um arquivo de inversões definido exatamente sobre os atributos de ligação,e ARQ2 está classificado segundo esta ordem dos atributos de ligação.	LIG2A	$CP_{INV_{ARQ1}}$	$Adic_{ARQ2}$
	ARQ1 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão, e ARQ2 está classificado segundo esta ordem dos atributos de ligação.	LIG2B	$CP_{INV_{ARQ1}}$	$Adic_{ARQ2}$
	ARQ1 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de ligação, sendo este um arquivo de inversões sobre chave primária, e ARQ2 está classificado segundo esta ordem.	LIG2A	$CP_{INV_{ARQ1}}$	$Adic_{ARQ2}$
	ARQ1 possui um arquivo de inversões definido exatamente sobre os atributos de ligação.	LIG2A	$CP_{INV_{ARQ1}}$	Não (nenhuma ordem)
	ARQ1 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão.	LIG2B	$CP_{INV_{ARQ1}}$	Não (nenhuma ordem)

TABELA 7.17: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
APENAS ARQ1 É FOLHA	ARQ1 e ARQ2 estão classificados por diferentes ordens dos atributos de ligação.	LIG4	$\{Or_{ARQ1}\} \cup C(L_{ARQ2})$	$Adic_{ARQ2}$
	Apenas ARQ1 está classificado segundo alguma ordem dos atributos de ligação.	LIG4	Or_{ARQ1}	Não (nenhuma ordem)
	Apenas ARQ2 está classificado segundo alguma ordem dos atributos de ligação.	LIG4	$C(L_{ARQ2})$	$Adic_{ARQ2}$
	Nem ARQ1 nem ARQ2 estão classificados segundo os atributos de ligação.	LIG4	$P(L_{ARQ1})$ $(P(L_{ARQ2}))$	Não (nenhuma ordem)
APENAS ARQ2 É FOLHA	Ambos ARQ1 e ARQ2 classificados por uma mesma ordem dos atributos de ligação.	LIG4	Or_{ARQ2}	$Adic_{ARQ2}$
	ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de ligação, e ARQ1 está classificado segundo esta ordem dos atributos de ligação.	LIG3A	$CP_{INV_{ARQ2}}$	Não (nenhuma ordem)
	ARQ2 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão, e ARQ1 está classificado segundo esta ordem dos atributos de ligação.	LIG3B	$CP_{INV_{ARQ2}}$	$Adic_{ARQ2}$

TABELA 7.17: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
APE-NAS ARQ2 É FOLHA	ARQ2 possui um arquivo de inversões definido exatamente sobre um subconjunto dos atributos de ligação, sendo este um arquivo de inversões sobre chave primária, e ARQ1 está classificado segundo esta ordem.	LIG3A	$CP_{INV_{ARQ2}}$	Não (nenhuma ordem)
	ARQ2 possui um arquivo de inversões definido exatamente sobre os atributos de ligação.	LIG3A	$C(L_{ARQ1})$	Não (nenhuma ordem)
	ARQ2 possui um arquivo de inversões onde os atributos de ligação são "os primeiros" da chave de inversão.	LIG3B	$C(L_{ARQ1})$	$Adic_{ARQ2}$
	ARQ1 e ARQ2 estão classificados sob diferentes ordens dos atributos de ligação.	LIG4	$C(L_{ARQ1}) \cup \{Or_{ARQ2}\}$	$Adic_{ARQ2}$
	Apenas ARQ1 está classificado segundo alguma ordem dos atributos de ligação.	LIG4	$C(L_{ARQ1})$	Não (nenhuma ordem)
	Apenas ARQ2 está classificado segundo alguma ordem dos atributos de ligação.	LIG4	Or_{ARQ2}	$Adic_{ARQ2}$
	Nem ARQ1 nem ARQ2 classificados segundo os atributos de ligação.	LIG4	$P(L_{ARQ1})$ $(=P(L_{ARQ2}))$	Não (nenhuma ordem)

TABELA 7.17: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
NEM ARQ1 NEM ARQ2 SÃO FOLHAS	Ambos ARQ1 e ARQ2 classificados por uma mesma ordem dos atributos de ligação.	LIG4	$C(L_{ARQ1}) \cap C(L_{ARQ2})$	$Adic_{ARQ2}$
	ARQ1 e ARQ2 classificados sob diferentes ordens dos atributos de ligação.	LIG4	$C(L_{ARQ1}) \cup C(L_{ARQ2})$	$Adic_{ARQ2}$
	Apenas ARQ1 classificado segundo alguma ordem dos atributos de ligação.	LIG4	$C(L_{ARQ1})$	Não (nenhuma ordem)
	Apenas ARQ2 classificado segundo alguma ordem dos atributos de ligação.	LIG4	$C(L_{ARQ2})$	$Adic_{ARQ2}$
	Nem ARQ1 nem ARQ2 classificados segundo os atributos de ligação.	LIG4	$P(L_{ARQ1})$ $(=P(L_{ARQ2}))$	Não (nenhuma ordem)

TABELA 7.18: ESTREITT(AS) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
AS PROVÉM DE UMA FOLHA	AS é um registro de associação por item ou por idreg e os atributos mantidos constituem exatamente a chave primária da tabela de ligados.	ESTREITT1	CP_T
	AS é um registro de associação por item ou por idreg e os atributos mantidos são "os primeiros" da chave primária da tabela de ligados.	ESTREITT2	CP_T
	AS é um registro de associação por item ou por idreg e entre os atributos mantidos, está a chave primária da tabela de ligados.	ESTREITT3	CP_T
	AS é um registro de associação por item ou por idreg.	ESTREITT4	$P(M_T)$
	AS é um registro de associação no canal auxiliar e entre os atributos mantidos está a chave primária da tabela de ligados.	ESTREITT5	Or_T
	AS é um registro de associação no canal auxiliar e sua tabela de ligados está classificada segundo alguma ordem dos atributos mantidos, ou estes são "os primeiros" na ordem de classificação.	ESTREITT6	Or_T
	Nenhuma das anteriores.	ESTREITT6	$P(M_T)$

TABELA 7.18: Continuação

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
AS NÃO PROVÉM DE UMA FOLHA	Entre os atributos mantidos está a chave primária da tabela de ligados.	ESTREITT5	C_T
	A tabela de ligados de AS está classificada segundo alguma ordem dos atributos mantidos, ou os atributos mantidos são "os primeiros", na ordem de classificação.	ESTREITT6	$C_T \cap P(M_T)$
	Nenhuma das anteriores.	ESTREITT6	$P(M_T)$

TABELA 7.19: COLECT(AS) - Caminhamento"bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM
AS PROVÉM DE UMA FOLHA	AS é um registro de associação por item ou por idreg e a expressão seletora envolve apenas atributos da chave primária da tabela de ligados.	COLECT1	CP_T
	AS é um registro de associação por item ou por idreg.	COLECT2	CP_T
	AS é um registro de associação mantido no canal auxiliar.	COLECT3	Or_T
AS NÃO PROVÉM DE UMA FOLHA	(Neste caso, AS é um registro de associação mantido na zona intermediária).	COLECT3	C_T

TABELA 7.20: AGRUPART(AS) - Caminhamento "bottom-up"

	CONDIÇÕES	ALGORITMO	OPÇÕES DE ORDEM	
			LIGANTES	LIGADOS
AS PROVÉM DE UMA FOLHA	AS é um registro de associação por item ou por idreg e os atributos de agrupamento constituem exatamente a chave primária da tabela de ligados, ou são "os primeiros" da chave primária da tabela de ligados.	AGRUPART1	CP_T	$Adic_T$
	AS é um registro de associação por item ou por idreg.	AGRUPART2	$P(A_T)$	Não (nenhuma ordem)
	AS é um registro de associação mantido no canal auxiliar e os atributos de agrupamento constituem toda ou a parte inicial da chave de classificação da tabela de ligados.	AGRUPART3	Or_T	$Adic_T$
	Nenhuma das anteriores.	AGRUPART4	$P(A_T)$	Não (nenhuma ordem)
AS NÃO PROVÉM DE UMA FOLHA	Os atributos de agrupamento constituem toda ou a parte inicial da chave de classificação da tabela de ligados.	AGRUPART3	C_T	$Adic_T$
	Em caso contrário	AGRUPART4	$P(A_T)$	Não (nenhuma ordem)

7.4 Caminhamento "Top-down"

7.4.1 Importância das ordens de classificação

Após efetuado o caminhamento "bottom-up" da árvore de operações, a cada nodo não folha se encontram associados: um algoritmo para resolução desta operação e um conjunto de opções de ordem de saída, segundo as quais pode ser fornecido seu resultado.

A ordem de classificação de uma lista de dados intermediária, que representa o resultado de outra operação, deve ser escolhida de forma a melhorar a performance geral de execução da árvore de operações /SMI 75/. Se os resultados são fornecidos numa ordem interessante às operações que os utilizam, é diminuído o número geral de classificações e, conseqüentemente, é diminuído o número de acessos a um meio externo de armazenamento, melhorando-se o tempo total de resolução, além de se economizar espaço de armazenamento.

O exemplo a seguir mostra uma situação onde se pode avaliar a importância da ordem de saída de uma operação (os arquivos envolvidos são provenientes da figura 7.1).

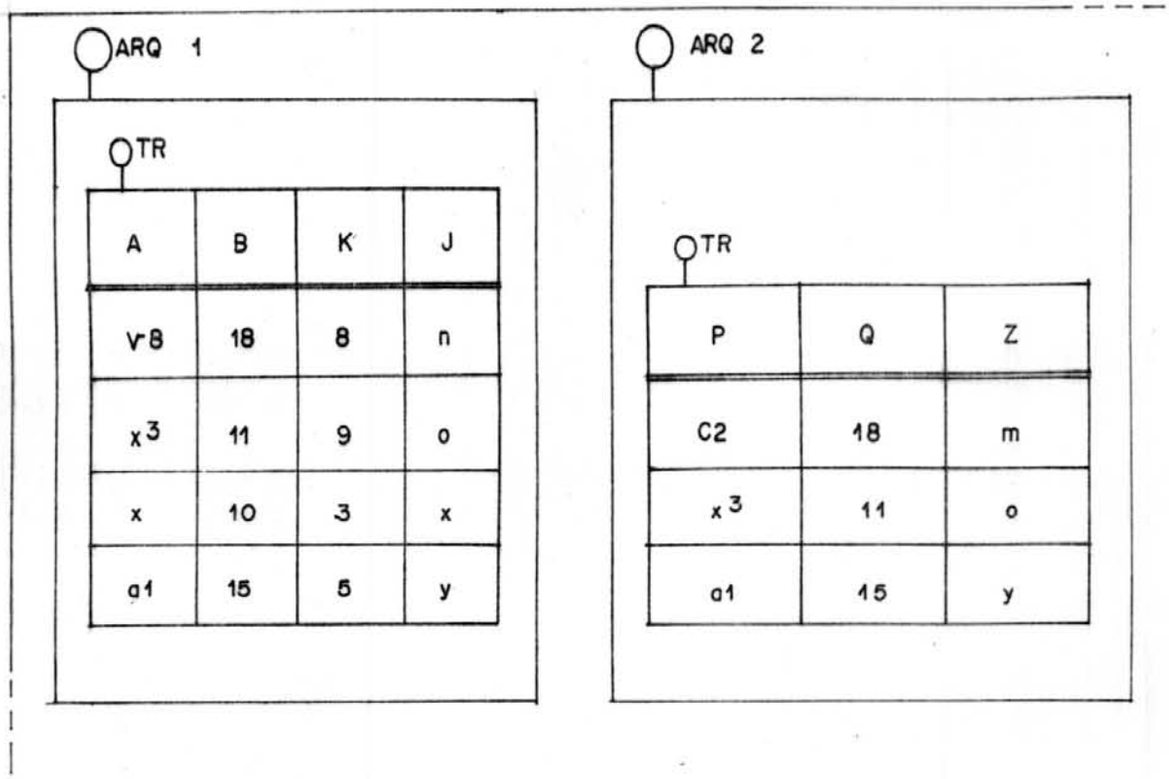


FIGURA 7.1: Acervo setorial

Exemplo:

```
JUNTAR
      ESTREITAR   C   ARQ1.TR
      PARA            B,K,J

COM C ARQ2.TR

POR   C  B  =  C  Q   ET
      C  J  =  C  Z
```

Supondo que a chave primária do arquivo relacional ARQ1 é composta dos itens A-B então a operação ESTREIT deve se preocupar com a eliminação de duplicatas (ESTR).

Como o operando a ser estreitado é uma constru-

ção da base de dados, o mesmo não se encontra classificado. Supondo que não existam arquivos de inversões disponíveis para esta operação, a resolução deve ser feita através de um algoritmo que classifique o arquivo de entrada segundo alguma ordem dos atributos mantidos pelo estreitamento e devolva registros já estreitados, eliminando as duplicatas.

Considerando que o resultado da operação ESTREITAR será dado conforme alguma ordem dos atributos mantidos (B,K,J), então as ordens de saída possíveis para a operação ESTREITAR são:

B - K - J
B - J - K
K - B - J
K - J - B
J - K - B
J - B - K

Considerando que não há arquivos de inversões ou arquivos de associações a utilizar, para a operação JUNTAR, então é interessante que a ordem de saída da operação ESTREITAR seja alguma ordem dos atriburos de junção: no caso, B-J-K ou J-B-K. Assim, apenas o segundo operando terá que ser classificado, enquanto que, sendo oferecida qualquer outra ordem de saída pela operação ESTREITAR, ambos os operandos teriam que ser classificados!

7.4.2 Método utilizado

Para determinação da ordem de saída mais interessante a cada operação, é utilizado o método introduzido em/SMI 75/, que prevê um caminhamento"top-down"da árvore de operações.

O caminhamento"top-down"visa decidir em que ordem de classificação devem ser fornecidos os resultados interme

diários, de forma a servirem mais eficientemente às operações que os utilizam. Sabendo-se qual o algoritmo escolhido para resolução de uma certa operação, é facilmente determinada a ordem de entrada mais interessante ao mesmo (ou exigida pelo mesmo). Se possível, esta deve ser a ordem de classificação do resultado da operação anterior, e assim por diante, até serem atingidas as folhas da árvore de operações. Desta forma, a ordem de entrada mais interessante a uma certa operação se propaga através da descida na árvore, influenciando não apenas o resultado da operação anterior mas, possivelmente, toda uma subárvore que a precede.

O caminhamento"top-down"da árvore de operações é mostrado em alto nível, na forma de um procedimento recursivo para visitação dos nodos da árvore e determinação das ordens de saída, denominado TOP-DOWN, que faz uso de dois argumentos de entrada: a indicação do nodo a ser visitado (NODO) e indicação de seu antecessor na visitação (PAI). Após a determinação da ordem de saída para uma certa operação, nada mais falta para que se proceda à geração de código, pois o algoritmo a usar já se encontra determinado.

No procedimento TOP-DOWN são introduzidos alguns nomes cujo significado é dado a seguir:

- ORDSAI(NODO): é a ordem de saída associada a cada nodo que representa uma operação (NODO).

- ORDENT(NODO): é a ordem de entrada necessária à operação dada em NODO. Pode ser determinada conhecendo-se o algoritmo a ser utilizado e a ordem de saída a ser fornecida pela operação dada em NODO.

- ALGORITMO(NODO): é o algoritmo a ser utilizado para resolução da operação dada em NODO.

- OPÇÕES(NODO): é o conjunto das opções de ordem de saída da operação dada em NODO.

- FILHO-ESQ(NODO): é a operação que produzirá o primeiro operando da operação dada em NODO (caso NODO represente uma operação binária) ou o único operando desta operação (caso NODO represente uma operação unária).

- FILHO-DIR(NODO): é a operação que produzirá o segundo operando da operação dada em NODO (caso NODO represente uma operação binária).

Alguns algoritmos, mesmo oferecendo apenas uma opção de ordem de saída, estão aptos a trabalhar do modo que lhes seja mais interessante, caso a ordem que oferecem não seja aquela preferida pelo algoritmo seguinte. Estão incluídos neste caso o COLR2, DESAGRUPAR2, JUNT1A e JUNT1B, que efetuam classificação de idregs a fim de diminuir o número de acessos para recuperação de registros de dados, caso a ordem que oferecem não possa ser aproveitada pela operação seguinte.

Na existência de mais de uma opção de ordem de saída, a escolha Indiferente (feita pelo algoritmo seguinte) faz com que a decisão da ordem de saída seja tomada internamente, pelo próprio algoritmo, optando por uma das ordens de classificação oferecidas.

Nos algoritmos de resolução das operações, a ordem Indiferente é tratada como uma opção de ordem Inválida, já que não se encontra no conjunto de opções de ordem oferecidas (o tratamento dado a uma opção de ordem inválida é próprio a cada algoritmo que permite opções de ordem, conforme detalhamento visto no capítulo 5).

No método aqui proposto, a raiz da árvore de operações tem sempre a ordem de saída escolhida como Indiferente, possibilitando que o algoritmo que execute esta operação decida, internamente, qual a opção de ordem que lhe é mais interessante.

A seguir é mostrado o procedimento recursivo de caminhamento"top-down"da árvore de operações. As tabelas referenciadas são parte do próximo item deste capítulo.

```
PROCEDIMENTO TOP-DOWN(NODO,PAI);

1 -       Se NODO não é uma folha, então:
          /* Supõe-se que as folhas são construções da base
          de dados ou do canal auxiliar. Portanto, não ne-
          cessitam ser "resolvidas" */
1.1 -       Se NODO é raiz, então ORDSAI(NODO)←Indiferente.
1.2 -       Do contrário, fazer:
1.2.1 -       Se em OPÇÕES(NODO) é oferecida uma única or-
              dem de saída, então:
1.2.1.1 -       Se ORDENT(PAI) ≠ ordem oferecida, então:
1.2.1.1.1 -       Se ALGORITMO(NODO) é um COLR2,
                                        DESAGRUPAR2,
                                        JUNT1A ou
                                        JUNT1B,
                  então ORDSAI(NODO) ← Indiferente.
1.2.1.1.2 -       Do contrário, ORDSAI(NODO) ← ordem ofere-
                                                 cida
                  /* tabela 7.21 */
1.2.1.2 -       Do contrário, ORDSAI(NODO) ← ordem oferecida
1.2.2 -       Do contrário, fazer:
1.2.2.1 -       Se ORDENT(PAI) está entre as ordens oferecidas em OPÇÕES(NODO), então
                ORDSAI(NODO) ← ORDENT(PAI)
1.2.2.2 -       Do contrário, ORDSAI(NODO) ← Indiferente.
                /* Tabelas 7.22 e 7.23 */
1.3 -       Gerar código.para a operação dada em NODO.
1.4 -       Estabelecer ORDENT(NODO)  /* tabelas 7.24 a 7.26*/
1.5 -       Se NODO tem filho à esquerda então
            TOP-DOWN(FILHO-ESQ(NODO),NODO)
1.6 -       Se NODO tem filho à direita então
            TOP-DOWN(FILHO-DIR(NODO),NODO).
```

7.4.3 Tabelas de ordem

São apresentadas aqui as tabelas utilizadas pelo procedimento recursivo de caminhamento "top-down" na determinação das ordens de saída e de entrada das operações.

As tabelas 7.21 a 7.23 mostram qual a ordem de saída fornecida em cada nodo, quando ORDSAI(NODO) tem valor Indiferente.

Na tabela 7.21 são apresentadas as ordens de classificação resultantes quando uma única opção de ordem é oferecida, mas a mesma não é interessante à operação seguinte (a escolha feita é indiferente). A tabela 7.22 apresenta as ordens de classificação resultantes nos casos em que várias opções de ordem são oferecidas, e a escolha feita é indiferente, para as operações relacionais, e a tabela 7.23 apresenta esta mesma situação junto às operações ligacionais.

TABELA 7.21: Opção única/escolha indiferente

ALGORITMO	ORDEM DE CLASSIFICAÇÃO DO RESULTADO
COLR2	Nenhuma ordem
DESAGRUPAR2	
JUNT1A	
JUNT1B	
Todos os demais	Ordem oferecida

TABELA 7.22: Várias opções/escolha indiferente - operações relacionais

OPERAÇÃO	ALGORITMO	ORDEM DE CLASSIFICAÇÃO DO RESULTADO
ESTR	ESTREITT5	Ordem em que os atributos mantidos pelo estreitamento aparecem nos registros de entrada.
ESTREITT	ESTREITT4	Ordem em que os atributos mantidos pelo estreitamento aparecem nos registros de entrada.
	ESTREITT6	
COLL	COLL2	Chave primária do arquivo/lista de dados dos ligados.
	COLL5	
	COLL6	
	COLL7	
JUNTAR	JUNT3	Ordem em que os atributos de junção aparecem nos registros do primeiro operando, se nenhum dos operandos está classificado segundo alguma ordem dos atributos de junção.
		Ordem de classificação do operando de maior cardinalidade, se ambos os operandos já estão classificados sob diferentes ordens dos atributos de junção.
UNI/ ISEC/ DIF	UNI1/ ISEC1/ DIF1	Chave primária do primeiro operando, se nenhum dos operandos se encontra classificado.
		Ordem de classificação do operando de maior cardinalidade, se ambos os operandos estão classificados sob diferentes ordens.
	UNI2/ ISEC3	Chave primária do primeiro operando.

TABELA 7.23: Várias opções/escolha indiferente - operações ligacionais

OPERAÇÃO	ALGORITMO	ORDEM DE CLASSIFICAÇÃO DO RESULTADO	
		LIGANTES	LIGADOS
AGRUPAR	AGRUPAR2	Ordem em que os atributos de agrupamento aparecem nos registros de entrada.	Não (nenhuma ordem)
AGRUPART	AGRUPART2	Ordem em que os atributos de agrupamento aparecem nos registros de entrada.	Não (nenhuma ordem)
	AGRUPART4		
LIG	LIG4	Ordem em que os atributos de ligação aparecem nos registros do primeiro arquivo/lista de entrada, se nenhum dos arquivos/listas estiver classificado segundo os atributos de ligação.	Não (nenhuma ordem)
		Ordem de classificação do arquivo/lista de dados de maior cardinalidade, se ambos os arquivos/listas já estão classificados sob diferentes ordens dos atributos de ligação.	Atributos adicionais sob os quais está classificado o segundo arquivo/lista de dados, além dos atributos de ligação.

Observe-se que, quando executado o passo 1.2.1.1.1 do procedimento TOP-DOWN, deve ser utilizada a tabela 7.21 para determinar qual a ordem de saída efetivamente fornecida. Já na execução do passo 1.2.2.2, devem ser utilizadas as tabelas 7.22 e 7.23, que determinam a ordem de saída fornecida quando são oferecidas várias opções de ordem e a escolha é indiferente.

Quando determinada a ordem de saída para uma certa operação, considerando que o algoritmo que a vai resolver já foi determinado (caminhamento"bottom-up"), é possível determinar-se a ordem de entrada para a mesma, ou seja, a ordem de classificação que devem apresentar, de preferência, seus operandos.

As tabelas 7.24 a 7.26 (utilizadas no passo 1.4 do algoritmo proposto) mostram as ordens de entrada preferenciais para as operações estudadas, conhecendo-se o algoritmo a ser utilizado e a ordem de classificação escolhida como saída. Os algoritmos cujos operandos só podem ser provenientes de folhas são omitidos nas tabelas, pois nesse caso a ordem de entrada é determinada pelo próprio contexto (ORDENT(NODO) ← ordem oferecida), sem oportunidade de opção.

As operações(LOBAN)JUNTAR e LIGAR constituem casos um pouco mais complexos, pois permitem que alguns dos atributos de junção/ligação sejam eliminados no resultado. Para estas operações, a determinação das ordens de entrada deve ser feita não apenas conhecendo-se a ordem de saída mas, além disso, relacionando-se os atributos das tabelas originais. A função T aparece na determinação das ordens de entrada para as operações CI3 JUNT e LIG, com o objetivo de representar esse relacionamento.

Por exemplo, considerando o acervo setorial da figura 7.1, a operação

```
JUNTAR C ARQ1.TR
COM    C ARQ2.TR

POR        C   A   EXCL  =  C   P          ET
           C   B         =  C   Q    EXCL  ET
           C   J         =  C   Z
```

... produz como resultado a tabela relacional mostrada na figura 7.2.

Caso não fosse permitida, numa operação JUNTAR, a opção EXCL, nenhum dos atributos dos operandos seria omitido na tabela resultante. No exemplo dado, se não fossem excluídos os atributos A e Q, e se a resolução fosse feita classificando-se os dois operandos pelos atributos de junção, então as opções de ordem de saída seriam dadas por:

$$P(\{\{\overset{A}{P}\},\{\overset{B}{Q}\},\{\overset{J}{Z}\}\})$$

Isto equivale a todas as permutações possíveis entre os atributos $\{\overset{A}{P}\},\{\overset{B}{Q}\},\{\overset{J}{Z}\}$ (a representação entre chaves dada por $\{\overset{a1}{a2}\}$ significa que, indiferentemente, qualquer um desses atributos, a_1 ou a_2, pode aparecer na solicitação de uma certa ordem de saída, numa certa posição, já que os mesmos têm valores iguais, num mesmo registro).

Considerando que o exemplo proposto seja resolvido pela classificação dos dois operandos por alguma ordem dos atributos de junção, vemos que as opções de ordem de saída são dadas por:

$$P(\{\{P\},\{B\},\{\overset{J}{Z}\}\})$$

Neste caso, apenas os atributos J e Z podem aparecer, indiferentemente, numa mesma posição, na ordem de saída, já que serão ambos mantidos nos registros resultantes.

TABELA 7.24: Ordem de entrada - operações unárias sobre listas de dados, operação ELEM

OPERAÇÃO	ALGORITMO	ORDEM DE ENTRADA (Operando: ARQ)
COLR,COLRU	COLR1	Ordem de saída
ESTR	ESTREIT4	Ordem de saída
	ESTREIT5	Indiferente
ESTRND	ESTREIT3	Ordem de saída
AGRUPAR	AGRUPAR2	Ordem de saída dos ligantes
ESTREITT	ESTREITT5	Ordem de saída
	ESTREITT6	Ordem de saída
COLECT	COLECT3	Ordem de saída
AGRUPART	AGRUPART3	Ordem de saída
	AGRUPART4	Indiferente
ELEM reg. dados	ELEM2	CP_{ARQ}
	ELEM3	Indiferente

B	K	J	P	Z
11	9	o	x^3	o
15	5	y	a1	y

FIGURA 7.2: Tabela relacional obtida pela operação JUNTAR do exemplo

Fato semelhante pode ocorrer com a operação LIG, que permite a exclusão de atributos de ligação nos registros de ligados formados.

A função T visa determinar, a partir da ordem de saída de uma operação JUNT ou LIG (correspondendo, em CI3, a um JUNTAR ou LIGAR), a ordem de entrada de cada um de seus operandos. Uma forma de implementá-la pode ser através de uma tabela mantendo a correspondência entre os nomes que constituem os atributos de junção ou ligação.

TABELA 7.25: Ordem de entrada - operações unárias sobre arquivos de associações, operação ELEM

OPERAÇÃO	ALGORITMO	ORDEM DE ENTRADA	
		LIGANTES	LIGADOS
COLA,COLAU	COLA5	Ordem de saída dos ligantes	Ordem de saída dos ligados
COLL	COLL7	Indiferente	Indiferente
DESAGRUPAR	DESAGRUPAR3	Ordem de saída	Indiferente
ELEM reg.ass.	ELEM6	Indiferente	Indiferente

TABELA 7.26: Ordem de entrada - operações binárias

OPERAÇÃO	ALGO-RITMO	ORDEM DE ENTRADA	
		PRIMEIRO OPERANDO (ARQ1)	SEGUNDO OPERANDO (ARQ2)
UNI,ISEC, DIF	DIF1/ UNI1/ ISEC1	Ordem de saída	Ordem de saída
	UNI2/ ISEC3	Indiferente	Indiferente
SUB,SUB=	SUB2	Se $C_{ARQ1} \cap C_{ARQ2} \neq \{\}$ então escolher uma opção do conjunto $C_{ARQ1} \cap C_{ARQ2}$.	
		Do contrário: - Se ambos classificados sob diferentes ordens,escolher uma opção do conjunto C_{ARQ}, onde ARQ é o operando de maior cardinalidade. - Se apenas um classificado, escolher uma opção do conjunto C_{ARQ}, onde ARQ é o operando classificado.	
	SUB1	$CP_{INV_{ARQ2}}$, se $\{CP_{INV_{ARQ2}}\} \cap C_{ARQ1} \neq \{\}$ ou indiferente, do contrário	-
	SUB4	-	$CP_{INV_{ARQ1}}$
JUNT	JUNT3	T(Ordem de saída)	T(Ordem de saída)
	JUNT2A	-	Se $T(CP_{INV_{ARQ1}}) \in C_{ARQ2}$ então $T(CP_{INV_{ARQ1}})$ Senão,indiferente
		Se $T(CP_{INV_{ARQ2}}) \in C_{ARQ1}$ então $T(CP_{INV_{ARQ2}})$ Senão, indiferente	

TABELA 7.26: Continuação

OPERAÇÃO	ALGO-RITMO	ORDEM DE ENTRADA	
		PRIMEIRO OPERANDO (ARQ1)	SEGUNDO OPERANDO (ARQ2)
JUNT	JUNT2B	-	Se $T(CP_{INV_{ARQ1}}) \varepsilon C_{ARQ2}$ então $T(CP_{INV_{ARQ1}})$ Senão, indiferente
		Se $T(CP_{INV_{ARQ2}}) \varepsilon C_{ARQ1}$ então $T(CP_{INV_{ARQ2}})$ Senão, indiferente	-
LIG	LIG4	Ordem de saída dos ligantes	T(Ordem de saída dos ligantes)
	LIG2A, LIG2B	-	$T(CP_{INV_{ARQ}})$
	LIG3A, LIG3B	Se $T(CP_{INV_{ARQ2}}) \varepsilon C_{ARQ1}$ então $T(CP_{INV_{ARQ2}})$ Senão, indiferente	-

8. CONCLUSÃO

Foi abordado, nesta dissertação, o problema da geração de código para operações de consulta ao banco de dados, em específico aquelas operações que manipulam tabelas inteiras (relacionais ou ligacionais) como operandos. Foram desenvolvidos algoritmos para resolução de cada operação, e foi estabelecido um processo de escolha do algoritmo mais adequado a cada situação, levando em consideração a organização física dos dados. Foram estudadas as ordens de classificação dos resultados produzidos pelas operações, sendo proposto um método para reduzir o número de operações de classificação e, conseqüentemente, o número de arquivos intermediários gerados, através do caminhamento na árvore de operações, em duas etapas: "bottom-up" e "top-down". Na primeira etapa (caminhamento "bottom-up") são determinadas as ordens possíveis em que os resultados das operações podem ser gerados, e na segunda etapa (caminhamento "top-down") é selecionada, para cada algoritmo, a melhor das opções oferecidas.

Uma opção de ordem de saída é escolhida sempre em função da ordem de entrada mais interessante à operação que utiliza esse resultado; logo, uma certa ordem escolhida pode-se propagar através da árvore, influenciando não só o resultado cuja ordem está sendo determinada mas, possivelmente, toda uma ramificação da árvore de operações.

No estudo das vantagens e desvantagens da classificação de listas de identificadores de registros, foi observado que este assunto ainda pode ser bem mais explorado, chegando-se até à determinação do ponto a partir do qual é interessante a geração de uma lista de identificadores e classificação da mesma, para posterior recuperação dos registros de dados. Determinado este ponto, e tendo-se em mãos alguns dados suplementares, como a cardinalidade do arquivo de dados e o fator de restrição da operação, pode ser visualizado um primeiro passo em direção à inclusão de métodos decisórios utilizando estatísticas /NOT 72, AST 76,

OLN 79/, na escolha do algoritmo mais adequado a cada situação.

Além do enfoque aqui utilizado na elaboração dos algoritmos, existem ainda outros enfoques para resolução de consultas ao banco de dados, como, por exemplo, o uso de processos concorrentes na resolução de operações independentes, ou o fornecimento dos resultados no modo "uma-tupla-por-vez" (ou "pipeline"), permitindo sua manipulação pelos próximos operadores, mesmo antes de produzido todo o arquivo de saída. A modalidade de resolução utilizada vai ao encontro das características do sistema L, para o qual deverá servir o estudo realizado.

Cumpre salientar, entretanto, que a forma de resolução aqui proposta não representa a meta final, na geração de código para as operações estudadas. A partir dos métodos apresentados, pode ser explorada a resolução conjunta de seqüências de operações, buscando diminuir o número de arquivos intermediários criados. Por exemplo, uma operação de seleção (dada pelo operador COLEC) tende a ser sempre embutida na operação que utiliza seu resultado, como um filtro a ser aplicado sobre os registros utilizados.

Esta forma adicional de otimização não-algébrica virá em parte a ampliar o enfoque utilizado na elaboração dos algoritmos propostos, permitindo a manipulação de registros por mais de uma operação a cada vez e, conseqüentemente, introduzindo uma modalidade de resolução em "pipeline".

BIBLIOGRAFIA

/AST 75/ ASTRAHAN, M.M. & CHAMBERLIN, D.D. Implementation of a structured english query language. Communications of the ACM, New York, 18(10):580-8, Oct. 1975.

/AST 76/ ASTRAHAN, M.M. et alii. System R: Relational approach to database management. ACM Transactions on Database Systems, New York, 1(2):97-137, Jun.1976.

/BLA 77/ BLASGEN, M.W. & ESWARAN, K.P. Storage and access in relational data bases. IBM Systems Journal, New York, 16(4):363-75, 1977.

/CAR 81/ CARVALHO, M.A. de. Sort; projeto MINIBAN. Porto Alegre, PGCC da UFRGS, 1981. (Relatório interno)

/CAS 79/ CASTILHO, J.M. & RICHTER, G. Uma interface de sistemas de informação: LOBAN - Linguagem de Operação de Bancos de Dados. In: CONGRESSO NACIONAL DE PROCESSAMENTO DE DADOS, 11., Rio de Janeiro, out.1978. Anais. Rio de Janeiro, SUCESU, 1978. p.347-54.

/COD 70/ CODD, E.F. A relational model of data for large shared data banks. Communications of the ACM, New York, 13(6):377-97, Jun. 1970.

/DUR 76/ DURCHHOLZ, R. & RICHTER, G. Information Management Concepts (IMC) for use with DBMS interfaces. In: NIJSSEN, ed. Modelling in data base management systems. Amsterdam, North-Holland, 1976.

/GOL 80a/ GOLENDZINER, L.G. Uma metodologia para melhorar a eficiência de instruções de recuperação em banco de dados. Porto Alegre, Universidade Federal do Rio Grande do Sul, 1980. 130p.

/GOL 80b/ _____. & HEUSER, C.A. Transformações para otimizar instruções em banco de dados. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 7., Campinas, Jul. 21-25, 1980. Anais. s.n.t. p.45-52.

/GOL 81a/ _____. Banco de Dados: uma linguagem unificada. Porto Alegre, PGCC da UFRGS, 1981. (não publicado).

/GOL 81b/ _____. Executor do Código L. Porto Alegre, PGCC da UFRGS, 1981. (Relatório MINIBAN/UFRGS 15).

/GOT 75/ GOTLIEB, L.R. Computing joins of relations. In: ACM SIGMOD - International Conference on Management of Data, San Jose, California, May 14-16, 1975. Proceedings. San Jose, Calif., IBM Research Laboratory, s.d. p.55-63.

/HEU 81/ HEUSER, C.A.; GOLENDZINER, L.G.; OLIVEIRA, J.P.M. Sistema L: uma implementação da linguagem LOBAN. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 8., Florianópolis, jul. 27-31, 1981. Anais. Florianópolis, UFSC, 1981. p.169-86.

/KIM 79/ KIM, W. Relational database systems. Computing surveys. New York, 11(3):185-211, Sept.1979.

/NOT 72/ NOTLEY, M.G. The Peterlee IS/1 System. Winchester, IBM United Kingdom Scientific Centre,1972. 20p.

/OLN 79/ OLNHOFF, T. & PAEUSCH, W. Efficient retrieval operations for a relational database system. Stuttgart, University of Stuttgart, 1979. 23p.

/RIC 77/ RICHTER, G. O sentido e o valor do banco de dados. Dados e Idéias. Rio de Janeiro, 2(6): 2-14, jun./jul. 1977.

/RIC 78/ _____.; PEREIRA FILHO, J.C.; CASTILHO, J.M.V. Projeto MINIBAN - Relatório final da segunda etapa. Rio de Janeiro, DIGIBRAS, 1978.

/SAN 80/ SANTOS, A.C. et alii. Especificação da interface LOBAN. Rio de Janeiro, COPPE/UFRJ, 1980. (Projeto MINIBAN/COPPE - Relatório técnico).

/SMI 75/ SMITH, J.M. & CHANG, P.Y. Optimizing the performance of a relational algebra database interface. Communications of the ACM, New York, 18(10):568-79, Oct.1975.

/TOD 74/ TODD, S.J.P. Implementation of join operator in relational data bases. Winchester, IBM United Kingdom Scientific Centre, 1974. 28p.

/TOD 76/ _____. The Peterlee Relational Test Vehicle - a system overview. IBM Systems Journal, New York, 15(4):286-308, 1976.

/VER 76/ VERHOFSTAD, J.S.M. The PRTV Optimiser: the current state. Winchester, IBM United Kingdom Scientific Centre, 1974. 48p.

/YAO 79/ YAO, S.B. Optimization of query evaluation algorithms. ACM Transactions on Database Systems, New York, 4(2):133-55, June 1979.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Pós-Graduação em Ciência da Computação

Um estudo sobre resolução de operações de consulta a Banco de Dados

Dissertação apresentada aos Srs.

Visto e permitida a impressão.
Porto Alegre, 19/03/1982

Coordenador do Curso de Pós-Graduação em Ciência da Computação