XIII CONGRESSO NACIONAL DE PROCESSAMENTO DE DADOS

OUTUBRO 1980

SUCESU

RIO DE JANEIRO

XIII CONGRESSO NACIONAL DE PROCESSAMENTO DE DADOS

ANAIS DO XIII CNPD

Hotel Nacional-Rio
Outubro de 1980

COMISSÃO ORGANIZADORA

Raulino Carvalho de Oliveira
Salvador Perrotti
Sylvio de Carvalho S. Mattos
Carlos Eduardo C. Fonseca
Raul Isiris
Fernando Vianna
Miguel Stábile
Eduardo Jorge Caldas Pereira
Gabriel de Almeida
Ronaldo Latsch
Carlos Alberto Pontes
Maria Elzira C. Isiris
Marlene Caldeira

COMISSÃO TÉCNICA

Fernando Vianna
Artur Edson Dias Pereira
Francisco Antônio Dantas Monteiro
José Eduardo Thyrso de Lara
Paulo Frota Simas de Oliveira
Jorge Peñaranda Coimbra
Carlos Jorge Zimmermann
Luiz Carlos Pinna

SUCESU-Sociedade dos Usuários de Computadores e Equipamentos Subsidiários

PRESERVAÇÃO DE RESTRIÇÕES DE INTEGRIDADE E SEGURANÇA EM BANCOS DE DADOS

LUCAS MORTIMER MACEDO,
A. L. FURTADO

SUCESU–SOCIEDADE DOS USUÁRIOS DE COMPUTADORES
E EQUIPAMENTOS SUBSIDIÁRIOS – RIO DE JANEIRO (1980)

PRESERVAÇÃO DE RESTRIÇÕES DE INTEGRIDADE E SEGURANÇA EM BANCOS DE DADOS

Lucas Mortimer Macedo, A. L. Furtado

Pontifícia Universidade Católica do Rio de Janeiro

RESUMO:

Este trabalho apresenta, através de um exemplo, uma metodologia para projeto de banco de dados levando em consideração:

– integridade de dados

– segurança de acesso

– facilidades operacionais para os usuários.

No exemplo adotou-se o modelo relacional e foi utilizado o sistema HYADES de gerência de banco de dados, desenvolvido na PUC/RJ. Entretanto a metodologia independe da escolha do modelo e pode ser usada com qualquer sistema de gerência de banco de dados.

1 – INTRODUÇÃO

Em todo banco de dados devem ser consideradas regras que determinam se os dados estão ou não corretos. Essas regras são denominadas *restrições de integridade*. Uma restrição de integridade pode ser violada pela execução (criação, remoção ou modificação de registros).

Por outro lado, existem regras que definem quais usuários tem direito de ter acesso a quais partes do banco de dados. Essas regras são chamadas de *restrições de autorização.* Uma restrição de autorização pode ser violada pela execução de operações de consulta ou de atualização por usuários não autorizados. Da obediência a essas restrições depende a segurança do banco de dados.

Vários métodos tem sido propostos para impedir a violação dos dois tipos de restrições, alguns tendo sido incorporados aos sistemas de gerência de banco de dados (SGBDs), como o Sistema–R [AST]e o sistema INGRES [STO] . Ambos são de disponibilidade muito limitada. Os SGBDs de uso corrente no mercado dispõem de facilidades menos poderosas, em grau variado.

Quando não se pode ou não se deseja (por considerações de custo/benefício) impedir cada execução indevida de operações, uma segunda solução é a de auditorar o banco de dados periodicamente. Isso significa que algum programa para este fim é executado de tempos em tempos, examinando os dados e detetando as violações de restrições ocorridas. Esta orientação implica em permitir que o banco de dados esteja incorreto durante o intervalo de tempo entre duas auditorias, e também que manipulações por usuários não autorizados não sejam surpreendidas imediatamente.

Uma terceira solução, proposta neste trabalho, consiste em limitar o acesso ao banco de dados a um determinado número de funções (sub-programas, procedimentos, rotinas) pré-programadas, sendo permitido a cada usuário (ou classe de usuários) o uso de apenas um sub-conjunto especificado dessas funções.

Para que se avalie o que significa esta abordagem é interessante comparar dois tipos de SGBDs:

– Os sistemas *abertos,* tais como IMS, TOTAL, ADABAS, DMS–II, IDMS, etc., que contem comandos básicos através dos quais se pode compor qualquer transação (conjunto de operações) de atualização ou consulta;

– Os sistemas *fechados,* tais como os sistemas de reserva de passagens aéreas, reservas em hotéis, etc., que contem um conjunto fixo pré-determinado de transações.

Essencialmente a solução que propomos nos coloca em um ponto intermediário quanto a esses dois tipos de sistemas. Estaremos usando um SGBD aberto para nele programar funções correspondendo a um elenco pré-determinado de transações; entretanto sempre que houver necessidade é possível (embora demande a tomada de certas providências pelo *Administrador de Banco de Dados – ABD)* aumentar esse elenco ou reduzi-lo.

Devemos advertir ao leitor de que os problemas que surgem com o uso concorrente e com sistemas distribuídos não serão considerados neste trabalho.

A seção 2 discute a metodologia em detalhe, a seção 3 a apresenta através de um exemplo e a seção 4 contém as conclusões. O apêndice contém uma função de consulta e uma de atualização.

2 – DESCRIÇÃO DA METODOLOGIA

Para o projeto de banco de dados vamos seguir a arquitetura em três níveis de esquemas, conforme a proposta de padronização do grupo ANSI/X3/SPARC [ANS]:

– Esquema conceitual – descrição em termos lógicos de banco de dados completo;

– Esquemas externos – descrição em termos lógicos de banco de dados a que cada usuário (ou classe de usuários) tem acesso;

– Esquema interno – descrição em termos de implementação física de banco de dados completo.

A Figura F–2.1 representa essa arquitetura, sendo que as setas indicam que os esquemas respectivos devem ser exprimiveis um em termos do outro.

Como usaremos no exemplo o modelo relacional [COD], o esquema conceitual toma a forma de um conjunto de *relações-base.* No esquema interno cada relação base corresponde a um segmento (arquivo) de dados, sendo também utilizados segmentos de inversões para acesso mais rápido aos segmentos de dados; segmentos de dados e de inversões são disponíveis no sistema HYADES [PAS], utilizado para implementar o exemplo.

Cada esquema externo consiste de um conjunto de visões. Uma visão é obtida através de projeções, junções, uniões, restrições, partições ou de – partições [FKE] realizadas sobre relações base. No entanto, para um usuário, uma visão tem o mesmo aspecto de uma relação base já que os usuários não precisam ter conhecimento de como as visões são formuladas. Esse aspecto é o de uma simples tabela.
Agora vejamos como são especificadas as funções de consulta e de autorização.

A cada visão corresponde uma função de consulta, que essencialmente obtem a visão a partir das relações base e a exibe ao usuário (por impressão ou através da tela do terminal). Na realidade nossa implementação é um pouco mais flexível, permitindo que o usuário especifique um ou mais valores dos atributos da visão de modo que somente as tuplas (linhas da tabela de que consiste a visão) contendo esses valores sejam exibidas. Nas conclusões indicaremos como uma flexibilidade ainda maior poderia ser obtida.

As funções de atualização se aplicam também sobre visões. Ao programá-las devem ser previstos os seguintes elementos, de modo a garantir a preservação das restrições de integridade;

– Condições – que, se não forem verificadas, levam a função a simplesmente retornar um código de advertência avisando ao usuário de que a atualização não pode ser efetuada;

– Efeitos – que correspondem às operações de atualização que o usuário pretende executar;

– Efeitos colaterais – que são outras atualizações complementares, não diretamente pretendidas pelo usuário, mas que são requeridas para a preservação da integridade.

Enquanto para consulta o usuário tem acesso a todos os atributos das visões de seu esquema externo, para atualização ele poderá estar limitado a alguns atributos apenas.

Ao escrever um programa de aplicação, o usuário incluirá uma ou mais chamadas a uma ou mais funções de consulta e/ou atualização. Na implementação que efetuamos usando o sistema HYADES um programa de aplicação tem o aspecto de um programa qualquer em PL/I. Como consequência secundária da metodologia adotada ficam assim reduzidas as necessidades de treinamento dos programadores, os quais não precisam aprender o uso do SGBD utilizado, bastando que saibam como chamar as funções; a implementação também envolve um procedimento catalogado na linguagem de controle (JCL) que dispensa a declaração dos arquivos do HYADES e de outras informações requeridas pelo HYADES e pelo Sistema operacional.

Afora isso, o programador só deverá ter um cuidado especial, requerido para garantir as restrições de autorização; o primeiro comando executável deverá ser uma chamada ao módulo HYAFCIN. Esse módulo efetua uma série de inicializações, das quais a mais importante para a presente discussão é trazer para posições da memória principal compartilhadas por HYAFCIN e pelas funções (mas não pelo programa

de aplicação do usuário) informações sobre o usuário que estão gravadas em um segmento de dados especial constituindo a relação de autorizações.

Cada tupla (linha, registro) da relação de autorizações corresponde a um usuário, indicando para esse usuário:

—Nome do usuário;

—sua senha;

—seu cargo;

—funções que está autorizado a utilizar.

Este último elemento tem a forma de um vetor de bits em que cada conjunto de bits (um byte na implementação) corresponde a uma visão e cada bit do conjunto corresponde a uma função sobre a visão (na implementação sobraram bits nos diversos bytes, possibilitando expansões futuras por acréscimo de novas funções). O primeiro bit de cada conjunto designa a (única por visão) função de consulta, e os demais as de atualização. Se o bit tem valor 1, o usuário está autorizado a chamar a função.

Voltemos à execução dos programas de aplicação. O módulo HYAFCIN lê o nome e senha do usuário (fornecidos por cartão ou teclado) e os utiliza como chave para localizar a tupla do usuário na relação de autorizações, trazendo-a para a memória. Se não se tratar de usuário reconhecido pelo sistema a execução é interrompida. Caso contrário prossegue e, ao ser chamada cada função , testes programados no início da função verificam por consulta ao vetor de bits se o usuário está autorizado a chamá-la; se não estiver, a execução da função se interrompe sendo retornado um código de advertência.

Uma medida adicional de segurança adotada na implementação foi a codificação ("encryption") do nome e senha do usuário, efetuada pelo sistema, preliminar ao acesso à relação de autorizações.

Certamente as medidas adotadas não tornam totalmente impossível a violação à segurança, mas a dificultam a um nível considerável. O sucesso da metodologia depende ainda:

— De os usuários serem instruídos sobre o uso das funções;

— de a empresa impor como norma que tais instruções devam ser seguidas rigorosamente;

— de os comandos básicos do SGBD não serem disponíveis aos usuários de aplicações.

A metodologia dá ênfase ao papel do ABD, ao qual (ou cuja equipe) compete especificar e programar as funções usando o SGBD e criar e modificar a relação de autorizações; para este último fim a implementação prevê funções privilegiadas que só o ABD pode utilizar.

Para programar as funções o ABD precisa conhecer as relações base (esquema conceitual) e como estão implementadas (esquema interno). Em particular, cabe-lhe decidir que arquivos de inversões devem existir para otimizar o tempo de acesso, levando em conta a frequência das operações. Por outro lado, o programador de aplicações não tem conhecimento disso e, se o banco de dados for reestruturado, seus programas de aplicação não precisam ser reescritos, desde que as sequências de chamada de suas funções (nome da função e parâmetros) não tenham sido alteradas; isso é verdade mesmo que o corpo das funções tenha sido alterado (pelo ABD). Em consequência, um outro benefício indireto da metodologia é uma maior independência dos programas quanto à estrutura de dados ("data independence").

O ABD conhecerá também todos os esquemas externos, estando assim em condições de disciplinar a interferência entre usuários, fenômeno que ocorre quando dois ou mais esquemas externos compartilham partes do banco de dados.

3 — UM EXEMPLO

O exemplo implementado é o caso hipotético da área de pessoal de uma pequena companhia industrial [FSA].

Supõem-se, que feita a análise do sistema de informações, foram identificadas as seguintes entidades e seus atributos:

— Empregados;

N — Nome do empregado
S — Salário
C — Cargo
H — Habilitação

— Projetos

P — Nome do projeto
T — Tarefa
L — Lider do projeto
H — Habilitação requerida por tarefa

O esquema conceitual consiste das seguintes relações bases:

EMP (N. S, C) — nome, salário e cargo do empregado;
REQ (T, H) — requisito de habilitação para executar tarefa;
ATR (N, T, P) — atribuição de empregado em projeto

a tarefa;
SUP (P, L) – Supervisão de projeto por um lider;
CAP (N, H) – Capacitação (habilitações) possuídas ou adquiridas por empregados.

Os usuários do banco de dados são:

– Gerente de pessoal
– Gerente de Engenharia
– Gerente de Treinamento
– Líderes de projetos.

A análise do sistema de informações também deteta que dados deverão ter acesso autorizado a quais usuários para simples consulta ou para consulta e atualização, e que restrições de integridade serão impostas.

Os quadros Q–3.1 e Q–3.2 mostram:

– Os esquemas externos dos usuários;
– Para cada função de atualização, as condições, efeitos, e efeitos colaterais respectivos.

Para exemplificar, consideremos a visão V–LIVRE do esquema externo do gerente de pessoal. Para consulta existe a função livre com três parâmetros: N, S, C. Se for chamada por

COD = LIVRE ('joão da silva', –, –)

será exibida a tupla (única) correspondente ao empregado com o nome indicado. COD recebe o código indicativo de término normal ou de advertência. A chamada com

COD = LIVRE (–, –, 'motorista')

causa a exibição de todas as tuplas de empregados com o cargo indicado. A chamada com todos os parâmetros em branco ocasiona a exibição de toda a visão.

Usuário/Visão		Descrição
GPE	V–LIVRE	Empregados não associados a projetos. É obtida de EMP, selecionando as tuplas (n,s,c) tais que n não aparecem nenhuma tupla de ATR. Operações: ADMITA, DESPECA.
	V–OCUP	Empregados associados a pelo menos um projeto. É obtida de EMP selecionando as tuplas tais que n aparece em pelo menos uma tupla de ATR. Sem operações.
GEN	V–NECESS (T, H)	Habilitação necessária para desempenhar uma tarefa. É idêntica a REO. operações: REQUEIRA, REMOVA.
	V–QUAL (N, C, H)	Qualificação dos empregados. É obtida concatenando as tuplas de EMP e CAP com o mesmo n. Sem operações.
	V–PROJ (P, L)	Projetos e seus líderes. É idêntica a SUP. operações; INICIE, SUBSTITUA, SUSPENDA, REINICIE, ENCERRE.
	V–DISTR (N, T, P)	Distribuição de empregados a projetos e tarefas. É idêntica a ATR. operações: ASSOCIE, DESASSOCIE.
GRH	V–REC.HU (N, C, H)	Recursos humanos. É obtida de EMP e CAP concatenando tuplas com o mesmo n. operações: ADQUIRA, PERCA.
	V–UTIL (H)	Habilitações requeridas por pelo menos uma tarefa. É obtida tomando o domínio H de REO. Sem operações.
LPJ	V–EQUIPE–p (N, C, H)	Empregados no projeto p. É obtida de EMP e CAP concatenando tuplas com o mesmo n e tomando somente as tuplas tais que (n,–,p) está em ATR e (p, 1) está em SUP, sendo 1 o usuário (somente o líder do projeto está autorizado a ter tais informações referentes ao seu projeto). Sem operações.
	V–ALOC-p (N, T)	Alocação de empregados a tarefas no projeto p. É obtida de ATR tomando apenas as tuplas (n,–,p) tais que (p, 1) está em SUP, sendo 1 o usuário, e tomando apenas os domínios N e T. Operações: ENCARREGUE, LIBERE.

V–TAREFA–p (T, H)	Exigência de habilitação para tarefa. É idêntica a REQ. Sem operações.

Quadro Q–3.1 Visões dos usuários autorizados.

Sobre essa visão podem ser executadas as operações ADMITA (N, S, C), e DESPECA (N), evidentemente com o efeito pretendido de acrescentar ou remover uma tupla de V–LIVRE. As restrições de integridade que nos interessam aqui são:

a – Os salários devem ser pelo menos iguais ao salário mínimo;

b – Um empregado contratado deve ter um e apenas um salário e cargo;

c – Somente empregados contratados podem ter suas habilitações registradas no banco de dados;

d – Somente empregados que não estão no momento associados a projeto algum podem ser despedidos.

A visão V–LIVRE contém apenas empregados não associados a projetos (V–OCUP contém os que se ocupam de pelo menos um projeto). A função ADMITA verifica, como condição, se já há alguma tupla correspondente ao empregado que se quer contratar (restrição b) e se o salário a ser concedido é inferior ao mínimo (restrição a), casos em que ADMITA falha, retornando diferentes códigos de advertência. Satisfeitas as condições, a inserção da tupla é feita na relação base EMP (da qual V–LIVRE é derivada como visão).

DESPEÇA só se aplica se o empregado não está associado a projetos (restrição d) e tem como efeito colateral remover as tuplas da relação base CAP que contenham habilitação do empregado demitido (restrição c). O efeito pretendido é obtido pela remoção das tuplas do empregado da relação base EMP.

Os programas das funções LIVRE e DESPECA constam do Apêndice. Linhas de comentários permitem examinar como funcionam os programas sem requerer conhecimento do sistema HYADES; merece referência o uso do procedimento SELINV do HYADES, para consulta a segmentos de inversões.

A implementação também inclui um programa genérico para usuários de aplicações, conveniente para uso através de terminal por usuários, não programadores. O programa requer que o usuário indique a função a executar e, depois de chamá-la e exibir o resultado, volta a ficar disponível para nova chamada de função, até que o usúario encerre a sessão.

4 – CONCLUSÕES

A metodologia proposta pode ser usada com qualquer SGBD, não dependendo de que este disponha de recursos sofisticados.

As restrições de integridade e de autorização são preservadas às custas de uma certa perda de flexibilidade. Em especial, se novas funções de atualização se tornam necessárias mais tarde (ex: para modificar salários de empregados), é necessário que o usuário interessado solicite sua especificação e programação ao ABD.

Na prática será necessária uma análise de custo/benefício para determinar para que restrições é compensatório utilizar a presente metodologia (ou outra qualquer) para garantir sua preservação. Em certos casos pode ser conveniente a política de auditoria periódica, ou até mesmo a de permitir a ocorrência e permanência de erros de menor importância.

Quanto à presente implementação, vários aperfeiçoamentos podem ainda ser introduzidos. Um deles é a saída das funções de consulta, que poderia consistir na gravação de um arquivo (que assim seria um instântaneo – snapshot – da visão) ao invés de imprimir a visão ou exibi-la em terminal; também se poderia adicionar como parâmetro uma função lógica (predicado), a ser programada separadamente pelo usuário, impondo condições à mais para exibir ou não cada tupla da visão (ex: só exibir tuplas de empregados com salário maior do que um certo valor e cujo cargo seja motorista ou mecânico de veículos, etc.).

BIBLIOGRAFIA

[ANS] ANSI/X3/SPARC
"Interim Report ANSI/X3/SPARC Study Group on Data Base Management Systems" – boletim FDT da ACM (1975).

[AST] Astraham, M. M. e outros
System R: relational approach to data base management ACM transactions on Data Base Systems 1, 2 (1976).

[COD] Codd, E. F.
"A relational model of data for large shared data banks" – Communications of the ACM (1970).

[FKE] Furtado, A. L. e Kerschber, L.
"An algebra of quotient relations" – Anais da SIGMOD Conference (1977).
[FSA] Furtado, A. L. e Santos, C. S.

"Organização de Banco de Dados" – Ed. Campus (1979).

[PAS] Passos, S. M. M., Vasques, R. P. e Furtado A. L.
"Um Suporte Básico para Implementação de Sistemas de Gerência de Banco de Dados" – Anais do Panel Expodata V (1978).

[STO] Stonebraker, M. R. e outros
"The design and implementation of INGRES" – ACM transactions on Data Base Systems – 1, 3 (1976).

AGRADECIMENTOS

ESta pesquisa teve o apoio financeiro da Plandata Consultoria e Processamento Ltda., do CNPq e da FINEP. Os autores agradecem a Andrés Rubistein pela formatação do texto.

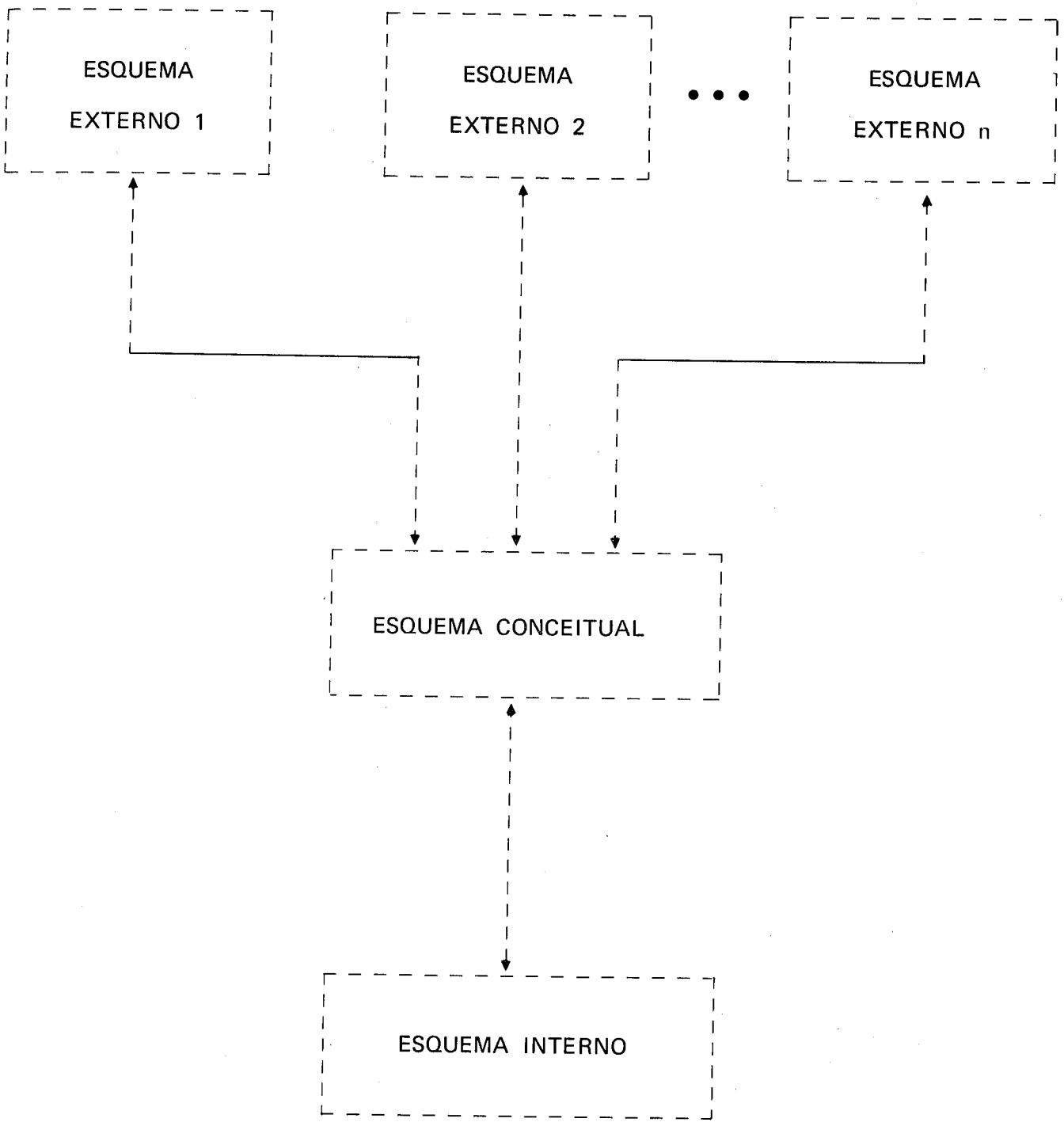

Figura F-2.1 – Arquitetura em três níveis de esquema.

VISÃO	OPERAÇÃO DE ATUALIZAÇÃO	DESCRIÇÃO	CONDIÇÕES	EFEITOS	EFEITOS COLATERAIS
V-LIVRE	admita(n,s,c)	contratar n com salario s e cargo c	não exista nenhuma tupla em EMP com n; s >= salario minimo	insere (n,s,c) em EMP	-----
	despeça(n)	demitir o empregado n	não exista nenhuma tupla em ATR com n	retira (n,-,-) de EMP	retira todas as (n,-) de CAP
V-NECESS	requeira(t,h)	exigir a habilitação h para a tarefa t	-----	insere (t,h) em REQ. retira (n,t,p) de ATR se não existe (n,h) em CAP	-----
	renova(t,h)	dispensar a exigência de h para executar t	-----	retira (t,h) de REQ	-----
V-PROJ	inicie(p,l)	iniciar o projeto p com o líder l	não exista nenhuma tupla em SUP com p	insere (p,l) em SUP	-----
	substitua(p,l)	substituir o líder atual do projeto p por l	-----	modifica (p,-) para (p,l) em SUP	-----
	suspenda(p)	suspender o projeto p retirando o líder	-----	modifica (p,l) para (p,*) em SUP	retira todas as (-,-,p) de ATR
	reinicie(p,l)	reiniciar um projeto p associando ao mesmo um líder l	exista (p,*) em SUP	modifica (p,l) para (p,l) em SUP	-----
	encerre(p)	encerrar o projeto p que estava suspenso	exista (p,*) em SUP	retira (p,*) de SUP	-----

Quadro Q-3.2a Operações de atualização.

V-DISIR	associe(n,p)	associar o empregado n ao projeto p	existe $(n,\neg,\neg)$ em EMP existe $(n,1)$ em SUP com $1 \neq *$; não existe $(n,-,p)$ em ATR	insere (n,*,p) em ATR	-----
	desassocie(n,p)	desassociar o empregado n do projeto p	-----	retira (n,-,p) de ATR	-----
V-PEC.EH	adquira(n,h)	habilitar o empregado n com h	existe a tupla com n em EMP	insere (n,h) em CAP	-----
	perca(n,h)	retirar habilitação h do empregado n	-----	retira(n,h) de CAP	para cada tupla (t,h) em REQ, retirar as tuplas (n,t,-) em ATR caso existam
T-ALOC-p	encarregue(n,t,p)	encarregar o empregado n da tarefa t	existe para cada h das tuplas (t,h) de REQ (n,h) de CAP, e pelo menos uma tupla (n,-,p) em ATR	se (n,*,p) está em ATR então modifica (n,*,p) para (n,t,p) senão insere (n,t,p) em ATR	-----
	libere(n,t,p)	liberar o empregado n da tarefa t	-----	retira (n,t,p) de ATR	-----

Quadro Q-3.2b Operações de atualização.

APÊNDICE

```
DESPECA: PROC (NOME)  RETURNS (BIN FIXED);                    /* 15.03.80 */
/***** EFEITO:
         RETIRA DOS ARQUIVOS OS DADOS REFERENTES A UM EMPREGADO.
         PARA QUE O EMPREGADO POSSA SER DESPEDIDO, E NECESSARIO
         QUE NAO ESTEJA VINCULADO A NENHUM PROJETO(S).
       PARAMETRO DA PROCEDURE:
         NOME: NOME DO EMPREGADO QUE DEVERA SER DESPEDIDO.
               ALFANUMERICO COM 10 POSICOES.
       CODIGOS DE RETORNO:
         CODIGO = 40 - FUNCAO BEM SUCEDIDA.
         CODIGO = 46 - TENTATIVA DE DEMITIR UM EMPREGADO INEXISTENTE.
         CODIGO = 51 - TENTATIVA DE DEMITIR UM EMPREGADO VINCULADO
                       A UM PROJETO.
         CODIGO = 62 - USUARIO NAO AUTORIZADO A UTILIZAR ESTA FUNCAO.
*****/
     /*DECLARACAO  DO  PARAMETRO  DA PROCEDURE                          */
     DCL  NOME                  CHAR (10);
     /*DECLARACOES DAS PROCEDURES EXTERNAS ATIVADAS                     */
     DCL  SELDAD                ENTRY (CHAR(7),PIC'9',PIC'(6)9',
                                       CHAR(502))
                                RETURNS (BIN FIXED);
     DCL  REMDAD                ENTRY (CHAR(7),PIC'(6)9')
                                RETURNS (BIN FIXED);
     DCL  SELINV                ENTRY(CHAR(7),PIC'9',CHAR(249),
                                      CHAR(7),PIC'(6)9')
                                RETURNS (BIN FIXED);
     /*DECLARACOES DOS PARAMETROS DAS PROCEDURES EXTERNAS               */
     /* DESCRICAO DE RELACAO   -  SISTEMA HYADES DE BANCO DE DADOS      */
     /*      SEGMENTO :  EMP   -  EMPREGADOS DA EMPRESA                 */
     /*      TUPLAS   : (N,S,C)-  NOME, SALARIO, CARGO                  */
     /*      MEMBRO   : REGEMP                                          */
     DCL  REG_EMP   EXTERNAL  CHAR (502);     /* REG_EMP - EXTERNAL    */
     DCL  P_EMP               POINTER;        /* POINTER               */
     DCL  1 REGISTRO_EMP      BASED (P_EMP), /* ESTRUTURA - 15 BYTES  */
            2 N               CHAR (10),      /* NOME DO EMPREGADO     */
            2 S               DEC FIXED (02),/* SALARIO                */
            2 C               CHAR (03);      /* CARGO                 */
     /*================================================================*/
     /* DESCRICAO DE RELACAO   -  SISTEMA HYADES DE BANCO DE DADOS     */
     /*      SEGMENTO :  ATR   -  ATRIBUICOES DE TAREFAS NOS PROJETOS  */
     /*      TUPLAS   : (N,T,P)-  NOME, TAREFA, PROJETO                */
     /*      MEMBRO   : REGATR                                         */
     DCL  REG_ATR   EXTERNAL  CHAR (502);     /* REG_ATR - EXTERNAL    */
     DCL  P_ATR               POINTER;        /* POINTER               */
     DCL  1 REGISTRO_ATR      BASED (P_ATR), /* ESTRUTURA - 35 BYTES  */
            2 N               CHAR (10),      /* NOME DO EMPREGADO     */
            2 T               CHAR (15),      /* NOME DA TAREFA        */
            2 P               CHAR (10);      /* NOME DO PROJETO       */
     /*----------------------------------------------------------------*/
     /* DESCRICAO DE RELACAO   -  SISTEMA HYADES DE BANCO DE DADOS     */
     /*      SEGMENTO :  CAP   -  HABILITACOES DOS EMPREGADOS          */
     /*      TUPLAS   : (N,H)  -  NOME, HABILITACAO                    */
     /*      MEMBRO   : REGCAP                                         */
     DCL  REG_CAP   EXTERNAL  CHAR (502);     /* REG_CAP - EXTERNAL    */
     DCL  P_CAP               POINTER;        /* POINTER               */
     DCL  1 REGISTRO_CAP      BASED (P_CAP), /* ESTRUTURA             */
            2 N               CHAR (10),      /* NOME DO EMPREGADO     */
            2 H               CHAR (15);      /* HABILITACAO           */
     /*----------------------------------------------------------------*/
     DCL (ID_REGEMP,
          ID_REGATR,
          ID_REGCAP)           PIC  '(6)9',
     ID_APODUM                 CHAR (7),
```

```
     DUM239                CHAR (239)   INIT(' ');
DCL  VET_AUT(96)           BIT(1) EXTERNAL, /* VETOR AUTORIZACOES */
     HEXDESP         BIN   FIXED  EXTERNAL; /* INDICE DE VET_AUT  */
/*DECLARACOES DE VARIAVEIS AUXILIARES                              */
DCL (COD,
     CODIGO)               BIN  FIXED;
/*ESTRUTURA PRINCIPAL                                              */
IF VET_AUT(HEXDESP) = '0'B
THEN CODIGO = 62;                          /* USUARIO NAO AUTORIZ.*/
ELSE DO;
P_EMP = ADDR (REG_EMP);
P_CAP = ADDR (REG_CAP);
P_ATR = ADDR (REG_ATR);
COD = SELINV ('NOMEEMP',4,NOME || DUM239,ID_ARQDUM,ID_REGEMP);
IF   COD ¬= 0
THEN CODIGO = 46;                          /* EMPREGADO N.EXISTE   */
ELSE DO;
     COD = SELINV ('NOMEATR',4,NOME||DUM239,ID_ARQDUM,ID_REGATR);
     IF   COD = 0
     THEN CODIGO = 51;                     /* N VINCULADO A PROJETO*/
     ELSE DO;
          IF   COD ¬= 8
          THEN CODIGO = COD;               /* ERRO HYADES          */
          ELSE DO;
               COD = REMDAD ('EMPT   ',ID_REGEMP);
               IF   COD ¬= 0
               THEN CODIGO = COD;          /* ERRO HYADES          */
               ELSE DO;
                    CODIGO = 40;           /* TERMINO NORMAL       */
                    COD = SELINV ('NOMECAP',4,NOME||DUM239,
                                  ID_ARQDUM,ID_REGCAP);
                    IF   COD = 0
                    THEN DO WHILE (COD = 0  &  ID_REGCAP ¬= 0);
                              COD = REMDAD ('CAPT   ',ID_REGCAP);
                              IF   COD ¬= 0
                              THEN CODIGO = COD;
                              ELSE COD = SELINV ('NOMECAP',5,NOME
                                   ||DUM239,ID_ARQDUM,ID_REGCAP);
                         END;
                    COD = SELINV ('NOMECAP',9,NOME||DUM239,
                                  ID_ARQDUM,ID_REGCAP);
                    END;
               END;
          END;
     COD = SELINV ('NOMEATR',9,NOME||DUM239,ID_ARQDUM,ID_REGATR);
     END;
COD = SELINV ('NOMEEMP',9,NOME || DUM239,ID_ARQDUM,ID_REGATR);
END;
RETURN(CODIGO);
END DESPECA;
```

```
LIVRE: PROC (NOME, SAL_CAP, CARGO) RETURNS (RIN FIXED);          /*23.02.80 */
/*      ROTINA DE CONSULTA DO EXEMPLO DE APLICAÇÃO DO SISTEMA HYADES
========================================================================
            VISÃO LIVRE DO GERENTE DE PESSOAL.
            FORNECE OS EMPREGADOS DA FIRMA QUE NÃO ESTÃO ASSOCIADOS A PROJETOS. É
            OBTIDA DE 'EMP' SELECIONANDO AS TUPLAS  (N, S, C)  TAIS QUE N NÃO APARECE  EM
            NENHUMA TUPLA (N, T, P) DE ATR UTILIZAÇÃO:
            = == = = ====

            EM PROGRAMAS ESCRITOS EM PLI=F, CHAMAR COM O COMANDO:
                CODIGO =LIVRE (NOME, SAL_CAR, CARGO)
            SENDO 'CODIGO' O VALOR DE RETORNO DA FUNÇÃO, QUE INDICARÁ O NÚMERO DA
            MENSAGEM DE SUCESSO OU FRACASSO NO USO DA MESMA.
        PARAMETROS DA PROCEDURE:
        ========== = =====
            NOME : NOME DO EMPREGADO                    ALFANUM. (10)
            SAL_CAR: VALOR DO SALÁRIO DO
                     EMPREGADO                          ALFANUM: (02)
            CARGO: INICIAIS DO CARGO DO EMPREGADO       ALFANUM: (03)
        VALORES DOS PARAMETROS:
        ====== == = = = = = ===
            OS PARAMETROS DESTA PROCEDURE PODERAO CONTER VALORES ESPECIFICADOS OU
            STAR EM BRANCO. QUANDO O VALOR DE UM PARAMETRO ESTÁ ESPECIFICADO NA
            CHAMADA ROTINA DE CONSULTA LIVRE, SOMENTE SERÃO EXIBIDAS AS TUPLAS CUJOS
            VALORES SATISFAÇAM OS VALORES PEDIDOS. SE O VALOR DE UM PARAMETRO ESTÁ
            EM BRANCO, TODAS AS TUPLAS PARA AQUELE PARAMETRO SERÃO EXIBIDAS.
        CODIGOS DE RETORNO:
        ======== =====
            CODIGOS =40 – FUNÇÃO BEM SUCEDIDA.
            CODIGOS =62 – TENTATIVA DE UTILIZAÇÃO DA FUNÇÃO POR USUÁRIO NÃO AUTORIZADO.
            CODIGO  =63 – NÃO EXISTEM TUPLAS PARA SEREM EXIBIDAS.
            CODIGO  =0 A 21 – ERRO NO SUPORTE HYADES DE BANCO DE DADOS.

        ESTRUTURA DA ROTINA:
        ==========
            1. TESTAR AUTORIZAÇÃO.
            2. INICIALIZAÇÃO DE VARIÁVEIS DE TRABALHO.
            3. SELEÇÃO DA PRIMEIRA TUPLA.
            4. VERIFICAÇÃO E TESTE DA TUPLA/SELEÇÃO DAS DEMAIS TUPLAS.
            5. LIBERAÇÃO DO SEGMENTOS (RELAÇÕES) UTILIZADAS.

*/      /*DECLARAÇÕES DOS PARAMETROS DA PROCEDURE.                          */
        DCL  NOME                  CHAR (10);
        DCL  SAL_CAR               CHAR (02),
           1 SAL_D      DEFINED    SAL_CAR,
           2 SALÁRIO               PIC'99';
        DCL  CARGO                 CHAR (3);
        /*DECLARAÇÕES DAS VARIÁVEIS EXTERNAS.                              */
        DCL  VET AUT (96)          BIT (1)  EXTERNAL; /*VET. AUTORIZAÇÕES   */
        DCL  HEXLIVR               BIN FIXED (15,0)   /*INDICE DO VETOR     */
                                   EXTERNAL;
        DCL  SYSPRINT              FILE;
        /* DECLARAÇÕES DAS PROCEDURES EXTERNAS ATIVADAS                     */
        DCL  SELDAD                ENTRY (CHAR (7), PIC'9', PIC'(6) 9',
                                   CHAR (502))
                                   RETURNS (BIN FIXED);
        DCL SELINV                 ENTRY (CHAR (7), PIC'9', CHAR (249),
                                   CHAR (7), PIC' (6) 9')
                                   RETURNS (BIN FIXED);
        /* DECLARAÇÕES DOS PARAMETROS DAS PROCEDURES EXTERNAS               */
        /* DESCRIÇÃO DE RELAÇÃO – SISTEMA HYADES DE BANCO DE DADOS          */
```

```
/*          TUPLAS   : (N, S, C) – NOME, SALÁRIO, CARGO                          */
/*          MEMBRO   : REGEMP                                                    */
DCL   REG_EMP      EXTERNAL A MAR (502);          /*REG_EMP – EXTERNAL          */
DCL   P_EMP                   POINTER;            /*POINTER                     */
DCL   1 REGISTRO_EMP          BASED (P_EMP),      /*ESTRUTURA – 15 BYTES        */
      2 N                     CHAR (10),          /*NOME DO EMPREGADO           */
      2 S                     DEC FIXED (02),     /*SALÁRIO                     */
      2 C                     CHAR (03);          /*CARGO                       */
/* ========================================== === ============                  */
/* DESCRIÇÃO DE RELAÇÃO – SISTEMA HYADES DE BANCO DE DADOS                       */
/*          SEGMENTO: ATR – ATRIBUIÇÕES DE TAREFAS NOS PROJETOS                  */
/*          TUPLAS   : (N, T, P) – NOME, TAREFA, PROJETO                         */
/*          MEMBRO   : REGATR                                                    */
DCL REG ATR     EXTERNAL CHAR (502);           /*REG_ATP – EXTERNAL            */
DCL P_ATR                POINTER;              /*POINTER                       */
DCL 1 REGISTRO_ATR       BASED (P_ATR)         /*ESTRUTURA – 35 BYTES          */
      2 N                CHAR (10);            /*NOME DO EMPREGADO             */
      2 T                CHAR (15),            /*NOME DA TAREFA                */
      2 P                C HAR (10);           /*NOME DO PROJETO               */
/* – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – */
DCL ID_REGEMP            PIC (6) 9',           /*IDENTIFICADOR REG. EMP        */
    ID_REGATR            PIC (6) 9',           /*IDENTIFICADOR REG. ATR        */
    ID ARODUM            CHAR (7),             /*IDENTIFICAÇÃO ARQUIVO         */
    TIP_SEL              PIC '9',              /*TIPO DE SELEÇÃO INV.          */
    DUM239               CHAR (239)      IMIT    (' ');
/* DECLARAÇÕES DE VARIÁVEIS AUXILIARES                                          */
DCL COD                  BIN FIXED (15),       /*COD RETORNO PRIMITIVOS        */
    CODIGO               BIN FIXED (15),       /*COD RETORNO FUNÇÕES           */
     X                   BIT (1),              /*VAR. ASSOCIADA A NOME         */
     Y                   BIT (1),              /*VAR. ASSOCI. A SALÁRIO        */
     Z                   BIT (1);              /*VAR. ASSOCI. A CARGO          */
/* ESTRUTURA PRINCIPAL                                                          */
/* 1. TESTE AUTORIZAÇÃO                                                         */
IF VET_AUT  (HEXLIVR) = 'O'B
THEN CODIGO = 62;                              /*USUÁRIO NÃO AUTORIZ.          */
ELSE DO;
/*2, INICIALIZAÇÃO DE VARIÁVEIS DE TRABALHO                                    */
     P_EMP = ADDR (REG EMP);
     P_ATR = ADDR (REG_ATR);
     IF NOME ¬=' '
     THEN X = '1 'B
     ELSE X = 'O'B
     IF SAL_CAR  ¬=' '
     THEN Y =' 1'B,
     ELSE Y  = 'O'B;
     IF CARGO ¬=' '
     THEN Z  = '1'B;
     ELSE Z  = 'O'B;
     IF X = '1'B
     THEN TIP_SEL = 4;
     ELSE TIP_SEL = 1;
     CODIGO = 63;                              /*NÃO EXISTE TUPLA              */
/*3, SELEÇÃO DA PRIMEIRA TUPLA EM 'EMP.T'  ATRAVÉS DE 'NOMEEMP'                */
     COD = SELINV ('NOMEEMP', TIP_SEL, NOME || DUM239, ID_ARODUM,
                   ID_REGEMP);
/*4, VERIFICAÇÃO E TESTE DA TUPLA LIDA/SELEÇÃO DEMAIS TUPLAS                   */
   DO WHILE (COD =0);                          /*EXISTE TUPLA (N, S, C)        */
          COD = SELDAD ('EMPT     ', 0, ID_REGEMP, REG_EMP);
          IF \ OD ¬=0
          THEN CODIGO = COD;                   /*ERRO HYADES                   */
          ELSE DO;
               IF (x = 'O'B  | NOME  = REGISTRO _EMP. N) &
                  (y = 'OB'  | SALÁRIO = REGISTRO_EMP, S) &
                  (z = 'O'B  | CARGO =REGISTRO_EMP. C)
               THEN DO;  /* TESTAR SE N EXISTE EM ATR                         */
```

```
COD =SFLIHV('NOMFATB',4,
            REGISTRO EMP.N||DUN239,
            ID_APODUM, ID_PECATR);
IF COD =8        / *(H, T, P) NÃO ESTA ATR */
THEN DO;         / *IMPRIMIR A TUPLA       */
     PUT SKIP (2) EDIT (' *LIVRE *NOME =',
                        REGISTRO_EMP. N,
                        ' /          SALÁRIO =',
                        REGISTRO_EMP.S,
                        ' /              CARGO =',
                        REGISTRO _EMP.C)

          (COL (1), A, A, A, P'99', A, A);
     CODIGO =40;
     COD      =0;
     END;
ELSE IF COD  =0
     THEN CODIGO =COD; / *ERRO  HYADES       */
IF   X                 / * SE NOME FOI ESPECIFIC. */
THEN COD =99;          / *PARA SAIR DO LOOPING */
END;
IF COD =0

THEN COD = SELINV ('NOMEEMP', 2, NOME ||DUM239,
                   ID_ ARODUM, ID_REGEMP);

END;

END;
/ *5. LIBERAÇÃO DOS SEGMENTOS (RELAÇÕES) UTILIZADAS */
     COD =SELINV ('NOMEEMP', 9, NOME ||DUM239, ID_ARODUM, ID_REGEMP);
     COD =SELINV ('NOMEATR',9, NOME ||DUM239, ID_ARODUM, ID_REGATR);
     COD =SELDAD ('EMPT     '9, ID_REGEMP, REG_EMP);
     END;
RETURN (CODIGO);
END LIVRE;
```