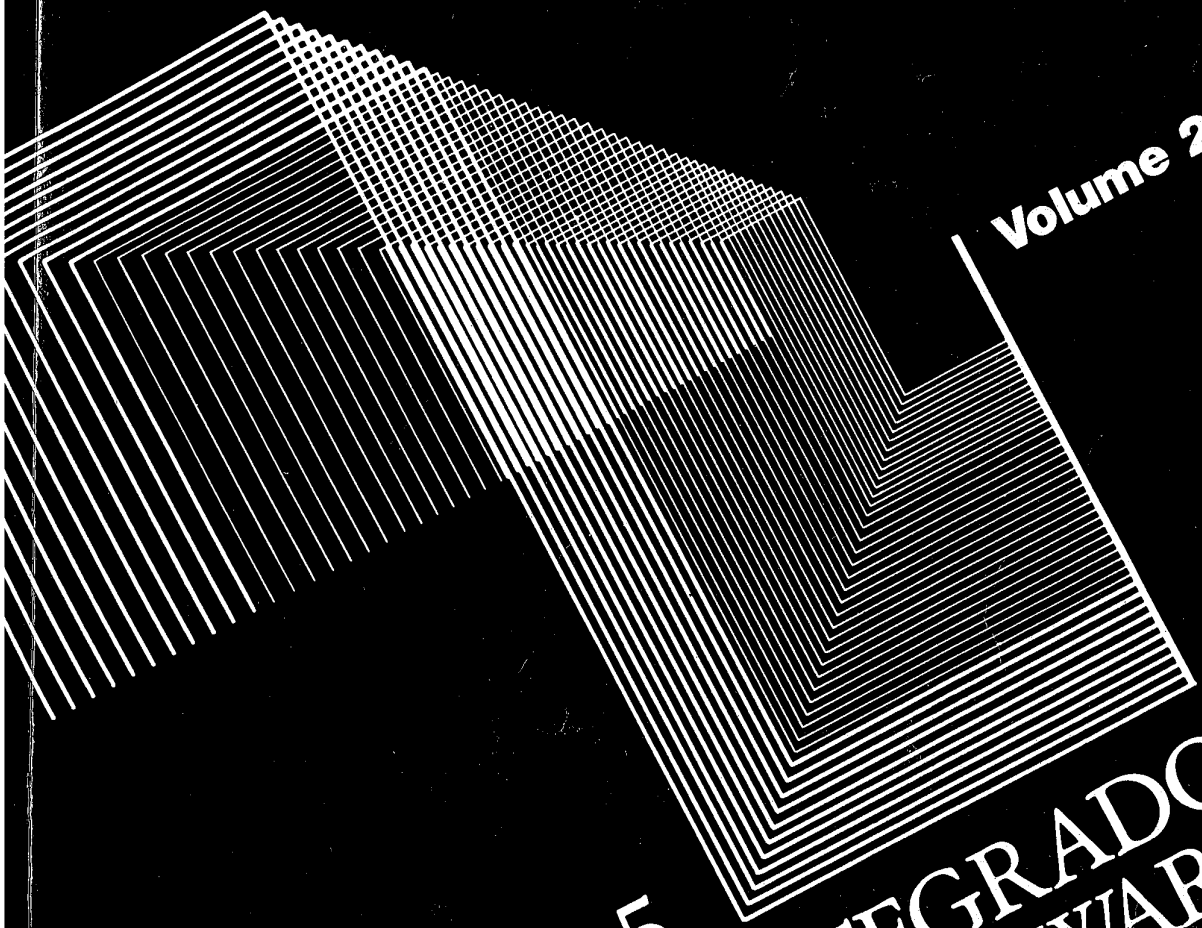


Volume 2

5 SEMINÁRIO INTEGRADO SOFTWARE E HARDWARE NACIONAIS

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Núcleo de Computação Eletrônica
CAPRE/FINEP
1978

TÉCNICAS DE INTEGRIDADE NO SISTEMA DE JACKDAW DE BANCOS DE DADOS

Conferencista e autor:

Michael F. Challis

Departamento de Informática da PUC/RJ

Resumo:

A parte mais importante deste trabalho, Seção II, é devotada à descrição de uma técnica para assegurar a integridade de um banco de dados. Esta técnica foi usada com sucesso pelo autor no sistema de banco de dados Jackdaw, e a primeira seção fornece uma breve introdução a este sistema. Finalmente, na Seção III, são sugeridas algumas aplicações da técnica em um ambiente onde vários usuários operam o sistema simultaneamente.

Seção 1

O "pacote" do banco de dados Jackdaw

1.1. - Status

O "pacote" Jackdaw foi escrito em uma linguagem de programação portátil, chamada BCPL [1], que está atualmente disponível no Brasil, no IBM 370/165 da PUC-RJ.

O núcleo do sistema fornece uma biblioteca de "procedimentos de interface" que podem ser usados por programas em BCPL, para perguntas e modificações sobre registros de um banco de dados. Vários programas de propósitos gerais também estão disponíveis para "não-programadores", podendo inclusive utilizar programas de perguntas/alterações projetados para uso interativo, e um gerador de relatórios para a produção de listagens formatadas.

Uma estrutura essencialmente de rede é adotada para a representação de ligação de dados, embora os conceitos envolvidos sejam um pouco mais objetivos do que os do CODASYL [2].

1.2. - Descrição estrutural

A maneira como a informação pode ser representada em um banco de dados Jackdaw será melhor ilustrada, através de exemplos (para maiores informações, veja [3]).

Os "comandos de definição" seguintes, definem um banco de dados com informações sobre lojas e sobre os seus itens à venda:

```
ADD   CLASS  LOJA
BEGIN
    STRING ENDEREÇO
    BOOL   ABREDOMINGO
END
ADD  CLASS  ITEM
BEGIN
    STRING ENDEREÇO
    INT PREÇO
END
ADD  LINK  (LOJAS,ESTOQUE) FROM ITEM TO LOJA
```

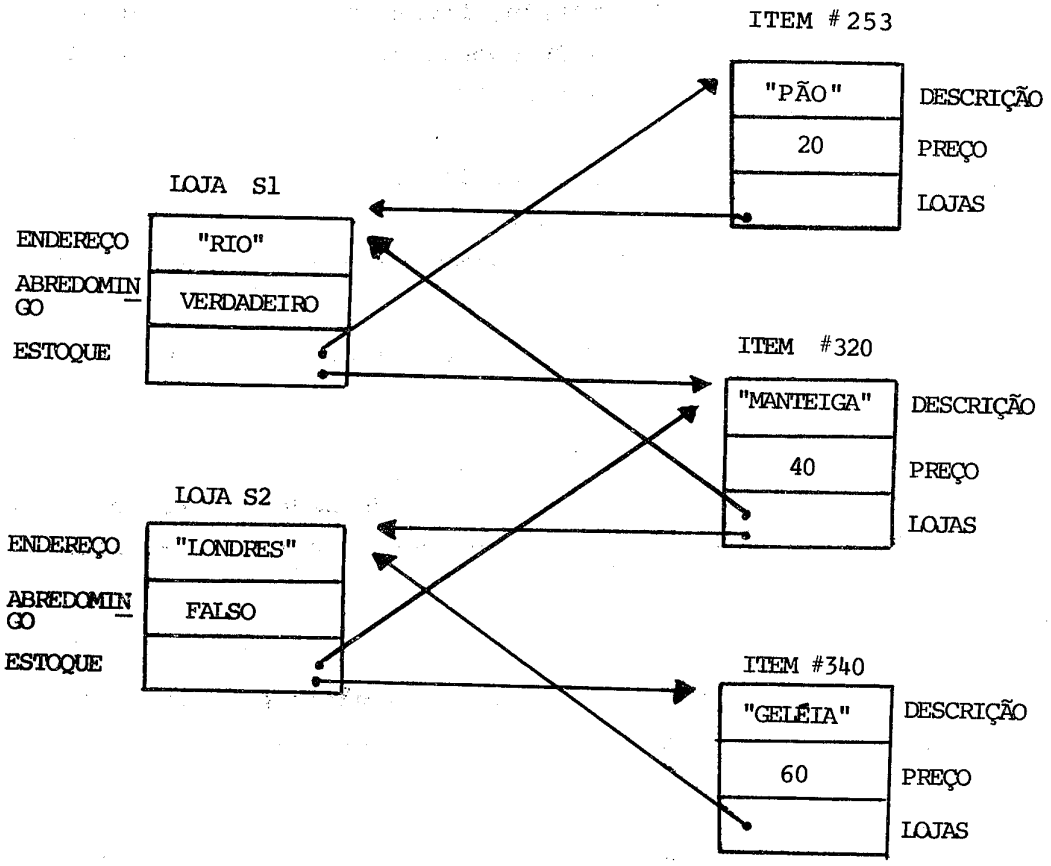
Os comandos "ADD CLASS" definem a estrutura básica dos diferentes tipos de registros que podem aparecer no banco de dados. Por exemplo, cada registro do tipo LOJA, tem um campo ENDEREÇO do tipo "STRING" e um campo lógico chamado ABREDOMINGO. Além disso, cada registro precisa ter um identificador (único na sua classe), pelo qual ele será referenciado; no caso dos registros do tipo ITEM, poderia ser o número de código do item, embora pudesse ser, da mesma forma, o nome do item, desde que fosse assegurado ser único.

O comando "ADD LINK" define uma ligação duplamente encadeada entre os registros do tipo ITEM e LOJA. Diferentemente do conceito CODASYL de ligação, o relacionamento é feito entre vários registros (vide [4]); um item pode ser estocado por várias lojas e uma loja pode estocar vários itens. A informação sobre quais lojas possuem um particular item X está guardada em um campo de ligação chamado LOJAS dentro do registro pertencente ao item X. Da mesma forma, os itens estocados pela loja Y estão contidos no campo de ligação chamado ESTOQUE no registro pertencente à loja Y.

Quando uma nova ligação é criada de um registro X ,

para outro registro Y, a correspondente ligação de Y para X é automaticamente inserida pelo "pacote", e assim isenta o programador de preocupações sobre o sentido da ligação. Ele pode adicionar ITEMs ao campo ESTOQUE, de um registro tipo LOJA ou adicionar LOJAs ao campo LOJAS de um registro tipo ITEM, conforme lhe seja mais conveniente.

O seguinte diagrama ilustra dois registros do tipo LOJA, relacionados a três registros do tipo ITEM:



S1 tem itens #253 e #320

S2 tem itens #320 3 #340

Concluimos esta seção com alguns exemplos de comandos na linguagem de perguntas/alterações, ilustrando como registros e seus campos podem ser manipulados.

Comando

```
TYPE  LOJA  S1   ENDEREÇO              / imprime    "RIO"
TYPE  LOJA  S1   ESTOQUE               / imprime"#253, #230"
ITEM  #320  PREÇO = 50                 /altera o preço
NEW   LOJA  S3   (ENDEREÇO="BRASILIA"
                  ABREDOMINGO=FALSE
                  ESTOQUE=(#340,#253)
                                       / cria e inicializa
                                       / um novo registro do ti
                                         po
                                       / LOJA
TYPE  ITEM  #340  LOJAS                / imprime  "S2,S3"
```

Seção II

Integridade do banco de dados

2.1. - Sua importância

Um banco de dados, especialmente em um ambiente "on - line", é um objeto muito valioso, e é essencial cercá-lo de tanta proteção quanto possível, dos efeitos de programas de aplicação imperfeitos.

Um banco de dados, por sua própria natureza, está sujeito a muitas aplicações distinitas, e um banco de dados corrompido por um programa, provavelmente não será utilizável por outro. Isto contrasta com o clássico método de processamento de dados, onde um arquivo mestre corrompido, só atinge aplicações de dependentes daquele arquivo.

Outra diferença importante está na maneira de se fazerem alterações. Na situação classica, um novo arquivo mestre é normalmente uma cópia modificado do original, que pode ser reinstalado se algo não dá certo durante o processamento do conjunto de alterações. Um banco de dados, contudo, será alterado dinâmicamente, possivelmente por muitas aplicações simultâneas, e é normalmente uma tarefa difícil, se não impossível, fazer uma recuperação do banco de dados original e dos programas de aplicação, no momento em que uma aplicação corrompe o banco de dados. (Esta situação é semelhante a um sistema operacional que utilize multi-programação, onde a falha de um programa de um usuário não pode afetar outros usuários independentes).

Para os propósitos do sistema Jackdaw, um banco de dados corrompido é definido como aquele que não pode ser processado corretamente pelo pacote. Exemplos de corrupção seriam índices inconsistentes ou uma ligação entre dois registros repre

sentada por um ponteiro em um único sentido.

As técnicas descritas abaixo para assegurar integrida de, protegem um banco de dados Jackdaw contra programas de a plicação imperfeitos e contra falhas no sistema operacional , mas não preve solução de problemas causados por ataques delibe rados de usuários maliciosos.

2.2. - Erros em programas de aplicação

Um banco de dados Jackdaw é constituido de uma se quência de blocos de tamanho fixo, em disco, que são trazidos à memória principal, modificados e escritos novamente em disco. Na atual implementação (como programa padrão de utilização nas séries IBM 360/370), o código do pacote, buffers e o código do programa de aplicação (PA), compartilham a mesma região da me mória principal, e assim, erros no PA podem resultar tanto no código do pacote quanto nos buffers terem sua capacidade exce-dida. No primeiro caso, podemos esperar que o pacote termina-rá anormalmente se o código corrompido chegar a ser utilizado, e assim o efeito será semelhante ao da queda de um sistema ope racional. O último caso é mais difícil de ser tratado, mas na prática parece que simples vistorias nos buffers antes que e les sejam regravados em disco, são adequados. (Por exemplo, ca da buffer inclui seu próprio número de bloco, que é examinado antes que ele seja gravado).

A possibilidade de escrever nos buffers destruindo seu conteudo anterior é também minimizada, por nunca se suprir o PA de ponteiros para áreas dentro dos buffers. De fato não existe qualquer necessidade de um PA ler ou escrever diretamen te de uma área do buffer, desde que toda informação passada en tre o PA e o banco de dados é sempre copiada de ou para uma á rea do PA, pelo próprio pacote. Endereços para tal transferên-cia de dados são naturalmente examinados pelo pacote, para as-

segurar que eles estão dentro da região pertencente ao PA.

Outros tipos de parâmetros transferidos do PA para o pacote, podem ser diretamente conferidos, desde que eles sejam valores previamente criados pelo pacote. Por exemplo, os três procedimentos de interface abaixo, do Jackdaw, podem ser usados para ler o valor do campo PREÇO do registro ITEM "#340":

```
X : = FINDENTRY(ITEM, "#340")
N : = READWORD (X, PREÇO)
RELEASEENTRY (X)
```

O valor atribuído a X, usado para representar o registro "#340", que foi localizado, é o endereço de uma pequena estrutura de dados criada pelo pacote, para descrever a localização e a estrutura do registro e para permitir que referências subsequentes aos campos do registro sejam processadas rapidamente. Portanto o pacote sabe que o primeiro parâmetro de "READWORD" tem que ser o endereço de tal estrutura de dados, e assim pode facilmente conferir sua validade.

É desnecessário dizer que se o código do pacote e os buffers estão em diferentes regiões de armazenamento, inacessíveis ao PA, exceto por chamada de procedimentos, então, muitos dos exames descritos acima seriam desnecessários. Infelizmente, o OS/MVT não provê este tipo de facilidade para programas de utilização normal.

2.3. - Quedas do Sistema

O caso de uma falha no sistema difere da falha do PA, porque não é possível ao pacote descobrí-la e tentar consertar mais tarde, e portanto, a proteção de um banco de dados contra tais casos não pode depender do conhecimento do que estava sendo executado imediatamente antes da queda. Em verdade, a úni-

ca versão existente do banco de dados, após uma falha do sistema, será a do disco, e a solução para o problema de integridade é assegurar que o banco de dados no disco esteja sempre consistente.

2.4. - Banco de dados físicos e lógicos

O arquivo em disco contendo um banco de dados Jackdaw é organizado como uma sequência de blocos de tamanho fixo (que podem ser acessados randomicamente), numerados de Ø a um valor máximo, digamos NF. A isto chamamos de banco de dados físico.

O banco de dados lógico é organizado de maneira semelhante, com blocos numerados de Ø a NL (< NF), e um mapeamento dos blocos lógicos para os blocos físicos, descreve aonde os blocos do banco de dados lógico aparecem dentro do banco de dados físico.

Ponteiros no banco de dados (representando campos de ligação, por exemplo), funcionam como pares número/afastamento de blocos lógicos, e a maior parte do código do pacote opera em termos desses endereços lógicos: somente o nível mais baixo (aonde paginação e reserva de espaço para os buffers é executada) precisa do conhecimento dos endereços físicos, e uma interface repara este nível do resto do pacote.

O mapeamento lógico para físico (LF) reside de tal maneira no banco de dados físicos, que pode ser trazido à memória principal, sem o conhecimento dele próprio: o primeiro bloco do mapeamento reside numa localização padrão do arquivo físico e contém o número do bloco físico do próximo bloco de mapeamento e assim por diante.

2.5. - O processo de modificação

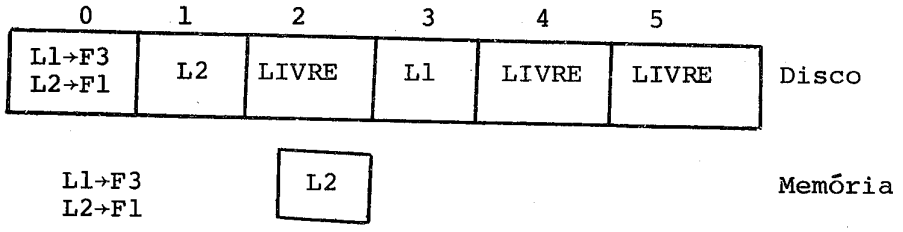
Quando o banco de dados é aberto inicialmente, o mapeamento LF é trazido para a memória principal e é usado para transformar endereços lógicos em endereços físicos, quando se faz necessária a leitura de blocos lógicos para buffers na memória principal. Quando um desses buffers é modificado pela primeira vez, um bloco físico livre é reservado para o correspondente bloco lógico e o mapeamento LF da memória principal é modificado para refletir tal mudança. Quando se faz necessário levar o buffer de volta para disco, ele virá para uma nova localização do banco de dados físico em vez de ser regravado sobre a versão original do correspondente bloco lógico. O efeito é que o banco de dados físico contém agora, dois bancos de dados lógicos: o original, descrito pelo mapeamento LF contido em disco e o alterado, descrito pelo mapeamento LF da memória principal.

Em algum momento adequado (vide seção seguinte) o mapeamento LF da memória principal é levado para o disco de tal maneira que o banco de dados lógico modificado está agora referenciado pelos mapeamentos existentes em disco e na memória principal. Se nós certificamos de que somente mapeamentos LF referidos a bancos de dados lógicos consistentes são gravados em disco, então somente bancos de dados consistentes serão definidos pelo banco de dados físico que estará portanto, protegido dos efeitos da terminação arbitrária de programa de aplicação.

O exemplo seguinte ilustra esta técnica para um banco de dados lógico de três blocos, em um banco de dados físico de seis blocos:

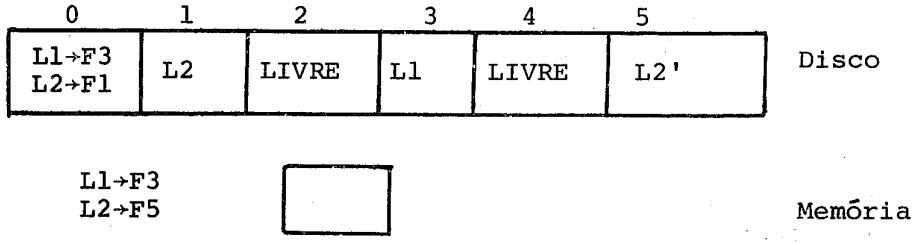
1) Abre o banco de dados e copia o mapeamento LF (no caso contido inteiramente no bloco físico $\emptyset$) para a memória princi-

pal; a seguir copia o bloco lógico 2 em um buffer:

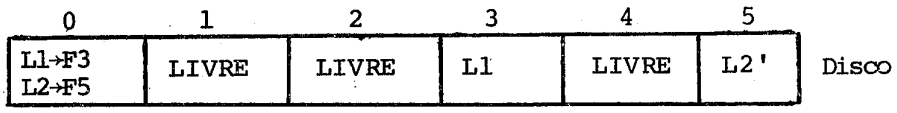


2) Altera L2 para L2' e associa um bloco físico livre a L2 no mapeamento da memória:

3) Quando precisarmos do buffer para algum outro propósito , torna-se necessário gravar o bloco alterado em disco. Note que neste estágio o arquivo em disco contém duas versões do mesmo bloco lógico (L2 e L2'):



4) Finalmente o mapeamento LF da memória é gravado em disco e os blocos físicos referenciados pelo mapeamento antigo, mas não pelo novo, estão liberados:



Existe um pequeno mas importante detalhe acerca da gravação de um novo mapeamento LF da memória para o disco. O novo mapeamento deveria sempre ser gravado em blocos livres do arquivo físico, de tal forma que se o sistema falhar no meio do processo de gravação, o mapeamento antigo ainda está disponível. Esta técnica também aumenta a probabilidade de uma recuperação bem sucedida, no caso de partes do arquivo em disco se tornarem ininteligíveis.

2.6. - Consistência lógica

O ato de gravar um novo mapeamento LF da memória em disco para definir um novo banco de dados lógico é chamado de "reconstrução" do arquivo em disco, e deve sempre ser feita antes do banco de dados ser fechado, após uma série de alterações. Durante a execução de um PA, só será necessário refazer o arquivo, se o número de blocos livres se tornar perigosamente baixo, ou se o PA pedí-lo explicitamente, a fim de executar uma "conferência" a partir da qual ele possa ser reiniciado.

Por outro lado, existem momentos em que não podemos reconstruir o arquivo, de modo a evitar a aparição de bancos de dados lógicos inconsistentes em disco. Por exemplo, se uma nova ligação deve ser criada entre dois registros, o pacote precisa alterar ambos os registros (provavelmente em blocos lógicos distintos), antes que o banco de dados esteja consistente.

A maior parte das situações pode ser contornada com a verificação da disponibilidade de um pequeno número de blocos físicos livres antes do processo de alteração ser iniciado: se

um número suficiente não estiver disponível, no momento, o arquivo em disco é reconstruído, a fim de liberar blocos referenciados pelo antigo mapeamento, mas não mais necessários no novo mapeamento.

Certas situações, contudo, podem requerer um maior número de blocos livres: o pior caso possível seria a deleção de um registro que tenha ligação com pelo menos um registro em cada bloco do banco de dados e portanto necessitando alterar todos os blocos lógicos, antes que o banco de dados se torne consistente. Em tais casos, o pacote fornece informação extra no disco, quando o arquivo é refeito, para indicar que alguma operação (ainda incompleta) está em andamento; se o sistema falhar antes de uma nova reconstrução, existe informação suficiente em disco, para permitir a complementação da operação, quando o banco de dados for novamente aberto.

Com o uso cuidadoso dessas técnicas, a implementação atual do pacote Jackdaw não apenas assegura que somente bancos de dados consistentes existirão em disco, mas também que cada procedimento de interface é "indivisível". Em outras palavras, qualquer chamada de um procedimento de interface, ou completará a ação especificado ou não fará nada.

Seção III

Conclusões e trabalhos futuros

3.1. - Desenvolvimentos futuros

A atual implementação do Jackdaw permite que somente um PA faça alterações no banco de dados de cada vez, embora vários usuários possam lê-lo simultaneamente. Versões futuras permitirão modificações paralelas, realizadas por diversos usuários, e novos procedimentos precisarão ser definidos, para permitir cooperação e interação controlada entre os usuários.

Um problema em um ambiente de bancos de dados "on-line" com um ou mais usuários modificando o banco de dados, é a obtenção de relatórios consistentes (vide [5]). Por exemplo, considere o caso de um programa que produz um resumo dos itens em estoque, seguido por um relatório detalhado mostrando a localização desses itens por armazém. Se os artigos estocados são alterados enquanto este programa está sendo executado, os totais dos dois relatórios estarão imcompatíveis.

Este problema pode ser resolvido pelo congelamento do mapeamento LP usado pelo gerador de relatórios, durante a sua execução. Durante este período, qualquer modificação realizada por processos concorrentes continuará a se refletir no novo mapeamento LF da maneira usual, exceto que os blocos físicos referenciados pelo mapeamento "congelado" não são reutilizados. O efeito é que o banco de dados físico passa a possuir tanto o banco de dados lógico "congelado", quanto o atual banco de dados lógico, até que os relatórios tenham terminado.

Uma técnica semelhante pode ser usada para testar com segurança novos programas de aplicação em um banco de dados "vivo". Quando o banco de dados é aberto em caráter de teste,

uma cópia do atual mapeamento LF é feita para o programa em teste. Quaisquer alterações feitas serão executadas nessa cópia, e não no mapeamento real. Ao final da execução, essa cópia poderia ser normalmente desprezada, mas poderia também ser conservada para permitir que programas "post-mortem" verifiquem como os testes se processaram. Uma vez mais, o efeito é que o banco de dados físico contém mais do que um banco de dados lógico: neste caso, o banco de dados "real" e o modificado pelo programa em teste.

3.2. - Sumário

Este artigo descreveu uma técnica geral para a proteção de um banco de dados em disco dos efeitos de erros do sistema e de programas de aplicação, essencialmente dependente de um único banco de dados físico ser capaz de refletir mais de um banco de dados lógico.

Futuras aplicações dessa técnica em um processo de multi-utilização foram sugeridos, inclusive a facilidade para a obtenção de relatórios consistentes durante um processo de mudanças constantes e a capacidade de depurar novas aplicações contra um banco de dados "vivo".

Esta técnica é empregada pelo pacote de banco de dados Jackdaw, que tem sido usado pelo banco de dados administrativo do Serviço de Computação da Universidade de Cambridge, durante quase quatro anos. Como indicação do sucesso dessa técnica, pode-se notar que nunca foi necessário proteger o banco de dados administrativo com uma cópia em fita magnética, embora alterações sejam feitas interativamente e todos os dias.

3.3. - Agradecimentos

Parte da pesquisa descrita neste artigo foi realizada enquanto o autor trabalhava para o Serviço de Computação da Universidade de Cambridge, na Inglaterra, e outros desenvolvimentos do pacote Jackdaw estão sendo parcialmente financiados pelo CNPq na Pontifícia Universidade Católica do Rio de Janeiro.

A tradução deste artigo foi feita por Celso Teixeira Souza, a quem agradeço.

Referências

1) Richards,M., 1974. "The BCPL Programming Manual", Computer Laboratory, University of Cambridge, Cambridge, England.

2) CODASYL, 1971. "CODASYL Data Base Task Group Report", April 1971, ACM, New York.

3) Challis,M.F., 1974. "The Jackdaw Databese Package", TR1 , Computer Laboratory, University of Cambridge, Cambridge, England.

4) Date,C.J., 1975. "An Introduction to Database Systems" , pp231-241, Addison-Wesley Publishing Company London.

5) Palmer,I., 1975 "Database Systems: A Practical Reference", pp2.29-30, C.A.C.I. Inc., London.

MFC/jrp

MICHAEL CHALLIS

Curriculum

O autor concluiu seus estudos de graduação em Matemática em 1967 e obteve um doutorado em Ciência da Computação na Universidade de Cambridge na Inglaterra.

Ele trabalhou no projeto e implementação de linguagens de programação assim como no desenvolvimento do pacote de banco de dados intitulado Jackdaw. Este último sistema teve a sua versão preliminar completada quando o autor ainda trabalhava no Centro de Computação da Universidade de Cambridge.

Desde 1976 o autor é professor associado do Departamento de Informática da PUC do Rio de Janeiro. Durante este tempo ele vem trabalhando em pesquisas na área de banco de dados, em particular no aprimoramento do sistema Jackdaw.