PUC

Série: Monografias em Ciência da Computação
Nº 5/80

UM ESTUDO SOBRE TÉCNICAS DE RECUPERAÇÃO DE ERROS EM BANCOS DE DADOS

Oscar E. Landes
Daniel A. Menascé

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 - CEP 22453
Rio de Janeiro — Brasil

Series: Monografias em Ciência da Computação No 5/80
Editor: Daniel A. Menascé Março 1980

UM ESTUDO SOBRE TÉCNICAS DE RECUPERAÇÃO DE ERROS EM BANCOS DE DADOS

OSCAR E. LANDES

e

DANIEL A. MENASCÉ

Departamento de Informática

ABSTRACT

This work is a survey of several different techniques used for crash recovery purposes in modern database management systems and file systems. These techniques restore the DB to a consistent state after a failure occurs. Different recovery mechanisms have to be used for different kinds of failures - transaction, system and secondary storage failures. These techniques also differ with respect to the state of integrity of the DB after the occurrence of a crash. Some of them are fault tolerant, others restore the DB to any previous consistent state, and others restore the DB to the most recent consistent state. Finally, we present some techniques which consider the transaction as the unit of recovery and consistency and implement the atomic ("all or none") property for transaction.

RESUMO

Este trabalho apresenta as diferentes técnicas e mecanismos de recuperação de erros utilizados hoje em dia pelos diferentes sistemas de arquivos e de bancos de dados. Estas técnicas de recuperação são utilizadas para restaurar os dados do sistema, após a ocorrência de uma falha, para um estado consistente. Existem vários tipos de falhas (de sistema, de transação e de memória secundária) e em consequência vários tipos de recuperação. As técnicas apresentadas neste trabalho diferem com relação ao estado de integridade do BD após a ocorrência de uma falha. Algumas delas são utilizadas para tolerar a ocorrência de falhas, outras para restaurar o BD para um estado consistente previamente existente e, outras são utilizadas para restaurar o BD para o último estado consistente existente antes da ocorrência da falha. Algumas técnicas apresentadas aqui consideram a transação como a unidade de recuperação e consistência e implementam a propriedade atômica para transações: "ou todas ou nenhuma das modificações de uma transação ocorrem no sistema".

1 - INTRODUÇÃO

Este trabalho tem por objetivo apresentar as diferentes técnicas e mecanismos de recuperação de erros utilizados hoje em dia pelos diferentes sistemas de arquivos e de bancos de dados. A intenção deste trabalho não é a de exaurir o assunto, mas sim a de apresentar de forma simples e clara as diferentes idéias utilizadas para restaurar os dados danificados após a ocorrência de uma falha.

Com o propósito de tolerar a ocorrência de falhas, componentes e algorítmos adicionais podem ser incluidos no sistema. Esses componentes e algorítmos tentam garantir que a ocorrência de estados errados não resultem em futuras falhas de sistema. Em termos ideais, os componentes e algorítmos removem os erros e restauram o BD para um estado correto, a partir do qual o processamento normal pode continuar. Esses componentes e algorítmos adicionais constituem as chamadas técnicas de recuperação [1].

As técnicas de recuperação são utilizadas para restaurar os dados de um sistema para um estado aceitável pelos usuários desse sistema. A noção de aceitável é diferente para diferentes ambientes . Em geral, aceitável significa correto, válido ou consistente.

Existem vários tipos de falhas (de sistema, de transação e de memória secundária) e em consequência vários tipos de recuperação. Para Verhofstad [1], existe sempre um limite para o tipo de recuperação que pode ser providenciado. Se a fa-

lha não apenas destrói os dados ordinários, mas também os dados de recuperação (dados redundantes mantidos pelo sistema para tornar a recuperação possível), a recuperação completa pode se tornar impossível.

Diferentes tipos de falhas são, em geral, tratados por mecanismos de recuperação distintos. É possível ainda, que o sistema não inclua mecanismos para certos tipos de falhas tais como: falhas raras, falhas que implicam em um elevado custo de recuperação (maior do que o valor da informação perdida), ou ainda, falhas não identificadas como possíveis durante o projeto do sistema.

A fim de que a recuperação de falhas seja possível torna-se necessário armazenar dados que permitam restaurar o estado em que o BD se encontrava por ocasião da ocorrência da falha. Estes dados, chamados de dados de recuperação, por sua vez também podem ser danificados devido a ocorrência de falhas. Por exemplo, uma falha em um cabeçote de leitura/gravação em disco pode destruir não apenas os dados ordinários mas também os dados de recuperação. Seria preferível manter os dados de recuperação em outro dispositivo de entrada/saída. Todavia, existem outras falhas que podem afetar esse dispositivo, por exemplo, falhas no mecanismo que grava os dados de recuperação para aquela memória auxiliar.

Os dados de recuperação, por sua vez, também podem ser protegidos de falhas pela manutenção de outros dados de recuperação que permitam restaurar os dados de recuperação origi-

mais no caso de sua destruição. Essa progressão poderia seguir indefinidadmente. Na prática, é claro, deve existir a aceitação de que dados de recuperação não podem ser totalmente dignos de confiança. Deve-se partir de alguns dados de recuperação básicos, aos quais atribui-se uma probabilidade pequena de destruição, para então, construir-se o mecanismo de recuperação.

Neste trabalho, são descritas técnicas que podem ser usadas para recuperação, para tolerância a falhas e para a manutenção da consistência depois da ocorrência de uma falha. A tolerância a falhas é conseguida se os algorítmos normais do sistema operam com os dados de tal maneira que após a ocorrência de certa falha o sistema estará sempre em um estado correto, isto é, o estado em que o sistema estava antes da ultima operação realizada sobre os dados (ou possivelmente a ultima série de operações).

Para Verhofstad [1], um BD está em um estado correto se a informação que o BD armazena consiste das cópias mais recentes dos dados introduzidas no BD pelos usuários e não contem dados deletados pelos usuários. Um BD está em um estado válido se suas informações são parte das informações existentes em um estado correto. E por ultimo, um BD está em um estado consistente se ele está em um estado válido, e as informações que o BD mantem satisfazem as restrições de integridade dos usuários.

Verhofstad [1] faz a seguinte classificação dos processos de recuperação, com relação ao seu tipo:

1) recuperação para um estado correto.

2) recuperação para um estado correto que existia em algum momento no passado.

3) recuperação para um possível estado prévio, isto é, restauração de um conjunto de estados de arquivos previamente existentes que podiam não ter existido simultaneamente antes.

4) recuperação para um estado válido.

5) recuperação para um estado consistente.

6) resistência contra falhas.

A tolerância a falhas difere dos outros tipos de recuperação. Enquanto os outros tipos de recuperação restauram explícitamente estados, a tolerância a falhas mantem estados corretos pela maneira como os dados são manipulados e mantidos durante o processamento normal. Pode-se dizer que a tolerância a falhas restaura implícitamente estados. No final desse trabalho é feita uma relação entre as diferentes técnicas que serão apresentadas e os diferentes tipos de recuperação.

Dentro de um único sistema, podem existir diferentes mecanismos de recuperação correspondendo a diferentes tipos de falhas. Cada técnica de recuperação possui suas vantagens e desvantagens, mas cada uma delas possibilita ao sistema tolerar a ocorrência de diferentes tipos de falhas em diferentes ambientes. Isso implica em dizer, que todas as técnicas tem a

sua validade e não podemos dizer que esta é melhor que aquela sem considerarmos o ambiente em que estão atuando.

Para alguns sistemas, a transação é considerada a unidade de recuperação; ela portanto aparece para o usuário como uma ação atômica, isto é, "ou todas ou nenhuma das modificações de uma transação devem ser executadas". As técnicas consideradas nesse trabalho podem ser usadas para implementar essa propriedade atômica.

2 - TÉCNICAS DE RECUPERAÇÃO

As diferentes técnicas de recuperação conhecidas e usadas no presente que serão descritas a seguir, estão agrupadas em 7 categorias [1]:

1) Programa Restaurador
2) Dump
3) Trilha Auditora
4) Copia e Versão Corrente
5) Duplicação
6) Arquivos Diferenciais
7) Substituição Cuidadosa

2.1 - Programa Restaurador

Um programa restaurador é executado após a ocorrência de uma falha para recuperar o sistema para um estado válido. Esse programa não utiliza dados de recuperação. (Essa é a unica técnica considerada aqui que não utiliza dados de recuperação). O programa restaurador é utilizado depois da ocorrência de uma falha se outras técnicas de recuperação, que utilizam dados de recuperação, falham, ou não são usadas, ou ainda se o sistema não é tolerante a falhas. Esse programa examina cuidadosamente o banco de dados após a ocorrência de uma falha, para avaliar os danos ocorridos e para restaurar o BD para algum estado válido. O programa restaurador tem o objetivo de salvar as informações que se mantem reconhecíveis após a falha.

O programa restaurador examina as estruturas de dados e tenta reconstruir o BD ou restaurar a consistência, possivelmente ao custo da deleção de alguns arquivos ou dados.

Se o sistema não possui resistência contra falhas, após a ocorrência de uma falha a memória secundária pode tornar-se inconsistente caso ocorra perda das informações armazenadas nos "buffers" da memória principal. Um programa restaurador pode ser utilizado então, para tentar salvar o conteúdo dos "buffers". De qualquer forma, se o conteúdo da memória principal é perdido, a restauração para um estado correto não é possível.

2.2 - Dump

Uma das técnicas mais utilizadas para prover a recuperação do BD face a ocorrência de uma falha é o dump. Um dump é uma cópia da versão corrente de um BD armazenada em memória "on-line" para uma memória secundária removível (geralmente fita). Juntamente com o dump deve-se manter um diário à medida que as transações executam e atualizam o BD. O diário contem informações suficientes para determinar as mudanças realizadas pelas transações no BD.

Para alguns sistemas, o diário contem a identificação de cada transação e os dados de entrada utilizados pela execução daquela transação. Para outros sistemas, o diário contem para cada atualização de uma entidade do BD, o novo valor da entidade (e algumas vezes o valor antigo). Em caso da ocorrência de uma falha, a mais recente cópia do BD é obtida, e as modificações indicadas pelo diário são então realizadas sobre essa cópia do BD. Se o diário contem a identificação e os dados de entrada para cada transação, as transações são reexecutadas para reconstruir o BD como ele existia antes da falha. Se o diário contem as atualizações feitas pelas transações, essas atualizações são usadas para modificar a versão copiada do BD.

Às vezes ao gerar um dump aproveita-se para reorganizar o BD e recarregá-lo na sua nova organização (Ex: consolidação do espaço livre deixado pela deleção de registros).

O processo de criar um dump pode ser:

a) ESTÁTICO

b) DINÂMICO

a) Dump Estático

Usualmente, o dump do BD é criado em uma maneira estática. Enquanto o dump está sendo criado, transações que atualizam o BD não são permitidas de executarem. Com isso,o sistema de BD não fica disponível para uso (por transações que atualizam) durante a crição do dump. A criação de dumps frequentes aumenta a quantidade de tempo em que o sistema de BD não pode ser utilizado por transações de atualização. No entanto, à medida que o tempo entre a geração de dumps aumenta, também aumenta o tempo necessário para reconstruir o BD depois da ocorrência de uma falha. Isto se deve ao fato de que o tempo para reconstruir o BD depende do tamanho do diário desde a ultima criação do dump.

As seguintes estratégias podem ser utilizadas para geração de um dump estático:

a.1) Copiar sequencialmente todo o BD.

a.2) Permitir cópias paralelas de partes do BD, tal que muitas partes do BD sejam copiadas ao mesmo tempo.

a.3) Permitir que apenas algumas partes do BD sejam copiadas a cada vez. Essa estratégia é conhecida como "dumping" incremental e é utilizada para copiar as entidades do BD que foram atualizadas, criando pontos de cheque para

essas entidades. Um ponto de cheque representa um estado em que não há nenhuma transação em progresso atualizando a entidade.

b) Dump Dinâmico

Não é necessário que o sistema de BD não possa estar disponível durante a criação do dump. O dump pode ser criado dinamicamente, enquanto transações são executadas e estão atualizando o BD. O dump criado dinamicamente deve representar um estado consistente do BD em algum instante de tempo. Para isso o dump precisa incluir todas as atualizações feitas pelas transações que tenham terminado em um tempo t, e nenhuma atualização feita por transações que terminaram depois daquele tempo t [2].

Existem tres tipos de métodos de geração de dumps dinâmicos [2]:

b.1) Dump da versão inicial: o dump criado representa o BD que existia no começo do processo de geração do dump.

b.2) Dump da versão final: o dump criado representa o BD que existe no momento do término do processo de geração do dump.

b.3) Dump da versão intermediária: o dump criado representa a versão do BD que existia em algum tempo t durante a execução do processo de geração do dump.

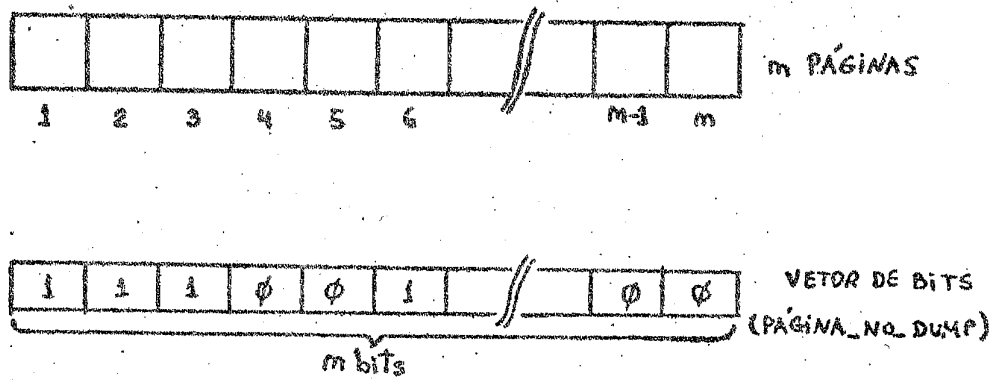
Para ilustrar, apresentaremos aqui, um algorítmo que permite obter dinamicamente um dump da versão inicial.

Algoritmo da Versão Inicial

O banco de dados está dividido em n páginas físicas P1,P2,...,Pn. Existe um vetor de n bits (PÁGINA_NO_DUMP[1..n]) mapeando essas páginas tal que:

$$\text{PÁGINA\_NO\_DUMP}[i] = \begin{cases} 1 \text{ se Pi já foi copiada para o dump.} \\ \emptyset \text{ se Pi ainda não foi copiada para o dump.} \end{cases}$$

A figura 1 mostra esse mapeamento.



Nesse exemplo as páginas P1,P2,P3 e P6 já foram copiadas para o dump

Figura 1 - mapeamento das páginas físicas do BD com relação ao dump dinâmico.

A variável X = i indica que as páginas P1,P2,...,Pi já foram copiadas para o dump.

O processo de geração do dump varre o BD da esquerda para a direita (da página P1 até a página Pn) copiando cada uma delas para o dump. No entanto, se o SGBD tenta atualizar a página Pj durante o processo de gereção do dump, o seguinte

algorítmo, descrito em "PASCAL-like", deve ser seguido:

```
If j < X
then ignore transação
else if PÁGINA_NO_DUMP[j] = 1
     then ignore transação
     else begin
          copie Pj para o dump;
          PÁGINA_NO_DUMP[j] := 1
          end
```

Se a página Pj já foi copiada para o dump; seja pela varredura ou porque ela foi atualizada previamente durante o processo de geração do dump, a transação não afeta em nada o dump. Se no entanto, Pj ainda não está no dump, ela deve ser copiada para o dump antes de ser atualizada.

Esse método de geração de dump garante que ao final do processo, o dump refletirá o estado do BD no início da geração do dump.

Recuperar um banco de dados muito grande, a partir de um dump pode levar muito tempo. Neste caso, é conveniente recarregar apenas as porções relevantes do BD. A fim de preservar a consistência interna do BD a escolha das porções a serem recarregadas deve levar em conta as transações que atualizam o BD. Um algorítmo que permite selecionar as porções do BD que devem ser recarregadas a fim de obter um BD num estado consistente está apresentado em [3].

2.3 - Trilha Auditora

A trilha auditora, também conhecida como "log" ou diário, é um arquivo sequencial que contem um histórico das atividades desenvolvidas no sistema (sequência de ações executadas nas entidades).

A trilha auditora pode ser usada para diferentes propósitos, tais como:

a) Corrigir erros e reconstruir o BD: se uma falha ocorre, o dump mais recente do BD é reinstalado e a trilha auditora pode ser usada para refazer no BD copiado todas as atualizações executadas desde o momento da execução do dump até o momento da falha. Com isso, é restaurado o estado em que o BD se encontrava no momento da falha. (reveja a técnica de recuperação dump: o diário está fazendo o papel da trilha auditora).

b) Desfazer atualizações executadas no BD: se o sistema falha, sem danificar a memoria secundária, as atualizações feitas por transações que estavam ativas no momento da ocorrência da falha devem ser desfeitas, restaurando o BD para o estado em que estava antes do início dessas transações. Para isso, a trilha auditora pode ser processada de trás para frente desfazendo-se todas as atualizações realizadas no BD. O efeito de uma transação sobre os dados do BD pode ter que ser desfeito no caso de ocorrência de impasses ou de falhas de transação. Em ambos os casos os dados afetados pela transação podem ser restaurados para seus estados antes do começo da transação.

c) Assegurar a segurança e a integridade do sistema através da trilha auditora pode-se verificar se todas as restrições de integridade definidas pelos usuários (regras e políticas ditadas pela lei, acordos comerciais, etc..., bem como, regras de consistência interna do sistema estabelecidas pelos projetistas do mesmo) estão sendo respeitadas. Nesse sentido a trilha auditora está sendo usada com o propósito de segurança e auditoria. Através das informações armazenadas na trilha procura-se detetar e descobrir as origens dos erros no BD.

As informações mantidas na trilha auditora podem ser

classificadas, com relação a sua utilização como técnica de recuperação, em informações relativas a:

- transações
- modificações dos dados do BD
- pontos de cheque

Informações sobre transações

A trilha auditora precisa manter para cada transação os seguintes dados:

. número sequencial da transação (identificação única da transação)
. Identificação do terminal fonte
. tipo da transação
. data e hora do início da transação
. texto da transação

Informações sobre modificações dos dados

A trilha auditora precisa manter as seguintes informações sobre cada atualização efetuada no BD:

. Número sequencial da transação
. Identificação do terminal fonte
. Tipo da transação
. Data e hora do início da transação
. Valor inicial da(s) entidade(s) que serão atualizadas
. Valor final da(s) entidade(s) atualizadas

Informações sobre pontos de cheque

Um ponto de cheque representa um estado do sistema em que não há nenhuma transação em progresso. Dentre as informações que compõem um ponto de cheque podemos citar:

- Informações sobre terminais ativos
- Ponteiros para filas de mensagens por tipo de mensagem, terminal e prioridade
- Conteúdo das filas em memória principal
- Tabelas de alocação de recursos e de bloqueios
- Informações sobre o BD: alocação de espaço, cabeças de cadeias, posições de arquivos sequenciais, etc...

É interessante salientar que os sistemas que utilizam essa técnica de recuperação devem implementar o "PROTOCOLO DA GRAVAÇÃO ADIANTADA DA TRILHA AUDITORA" [4]: o registro da trilha que contem as informações relativas às modificações feitas em uma entidade do BD, deve ser escrito no arquivo sequencial que contem as informações da trilha (geralmente fita) antes que a entidade seja escrita em memória secundária (geralmente disco). Isto se deve ao fato que a memória principal (memória de semicondutores) é volátil enquanto a memória secundária é não-volátil. Assim, se a entidade é escrita em memória não-volátil antes do registro da trilha auditora correspondente, uma falha pode tornar impossível desfazer uma ação.

Gray [4] salienta também, que para sistemas que necessitam de grande confiabilidade, é importante duplicar a gravação da trilha auditora, isto é, manter dois arquivos sequen-

ciais em unidades de entrada/saída distintas. Se a trilha auditora não é duplicada, uma falha de memória secundária na unidade da trilha pode ocasionar a perda de alguns dados de recuperação.

2.4 - Cópia da Versão Corrente

Essa técnica consiste em manter cópias do BD a fim de poder realizar uma possível restauração do BD para um estado prévio. Nessa técnica as transações são combinadas com a versão corrente do BD para produzir uma nova versão. A partir daí, a versão anterior do BD e as transações que acessaram os dados dessa versão são mantidos como cópias. No caso da ocorrência de uma falha, o estado atual do BD pode ser recuperado através da cópia (versão anterior) e das transações que acessaram essa cópia.

A técnica de dump pode ser utilizada para manter uma versão copiada do BD. Cópias de arquivos de alterações podem ser usadas para atualizar a versão copiada e obter a nova versão.

2.5 - Duplicação

A técnica de duplicação consiste em aplicar cada atualização ao BD, paralelamente a duas cópias do mesmo. A implementação dessa técnica é feita através da manutenção de duas cópias com "flags" para indicar "atualização em progresso".

diferencial será sempre pesquisado primeiro. Se o dado não for encontrado no arquivo diferencial ele deve ser pesquisado no arquivo principal. O algorítmo descrito na figura 2 representa a leitura de um registro.

Essa técnica apresenta as seguintes desvantagens [6]:

a) requer dois acessos ao BD por consulta.

b) eventualmente o arquivo diferencial deve ser fundido com o arquivo principal (uma operação que pode ser lenta)

c) tendo em vista que uma atualização pode afetar um dado já modificado, os arquivos diferenciais devem ser convenientemente organizados.

A fim de reduzir o número de acessos ao arquivo diferencial, Severance e Lohman [5] propõem uma técnica de filtros, descrita a seguir.

Elementos do filtro:

- Vetor de bits; B[1..M]
  Quando o arquivo diferencial está vazio, todos os bits de B são iguais a zero.

- X funções de hash
  Cada uma destas funções gera, a partir de uma identificação de um registro, um endereço no vetor B. (veja figura 3)

A figura 4 ilustra o funcionamento do filtro no armazenamento de um registro no arquivo diferencial.

Aplica-se as X funções de hash à identificação do registro. Obtem-se os endereços bi1,bi2,...,bix. Faz-se B[bij] = 1 para j = 1,2,...,X. Ao recuperar-se um registro R, pode-se de imediato saber se o registro não está no arquivo diferencial. Para isto, aplica-se as X funções à identificação de R e faz-

Uma cópia inconsistente é sempre reconhecida como inconsistente por causa dos "flags" utilizados.

Com a técnica de duplicação, as duas cópias do BD sempre possuem o mesmo valor exceto durante a atualização dos dados. Quando uma das cópias está sendo atualizada, o "flag" associado a essa cópia é setado para indicar que a atualização dessa cópia está em progresso. Se o sistema falha durante a atualização o "flag" continuará setado após a ocorrência da falha. Nesse sentido, a técnica de duplicação tem a característica de prover resistência a falhas. Uma cópia consistente sempre pode ser recuperada depois do reinício das atividades do sistema após a ocorrência de uma falha. Essa cópia terá, ou o valor que ela tinha antes da atualização que estava em progresso durante a falha, ou o novo valor. Sendo assim, em caso de falha de uma das duas cópias a outra poderá manter o sistema em funcionamento. A cópia inconsistente pode sempre ser reconhecida e ignorada.

Essa técnica de duplicação é utilizada em sistemas que exigem uma alta disponibilidade, tais como sistemas de reservas de passagens aéreas.

2.6 - Arquivos Diferenciais

Na técnica de arquivos diferenciais, os arquivos principais se mantêm inalterados até a reorganização dos mesmos. Todas as atualizações que seriam feitas em um arquivo principal são registradas em um arquivo diferencial [5]. O arquivo

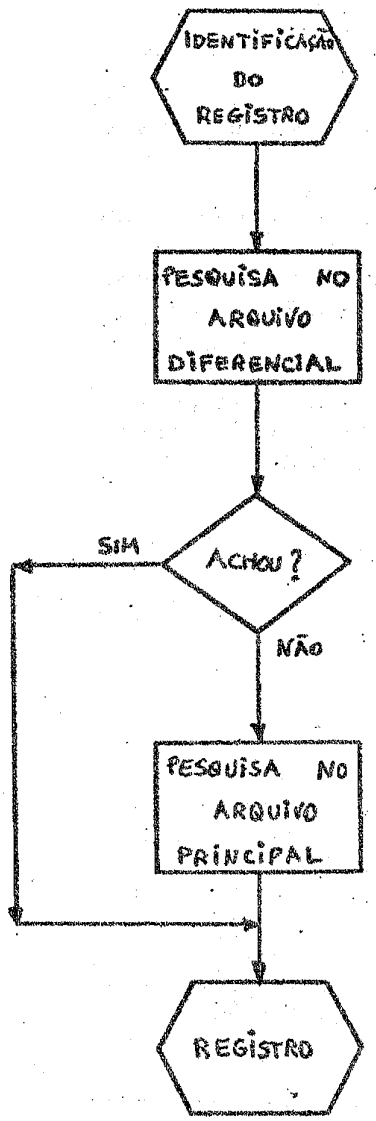


Figura 2 - Algoritmo de leitura de um registro em sistemas que utilizam arquivos diferenciais.

Figura 3 - Aplicação de uma função hash sobre a identificação de um registro.

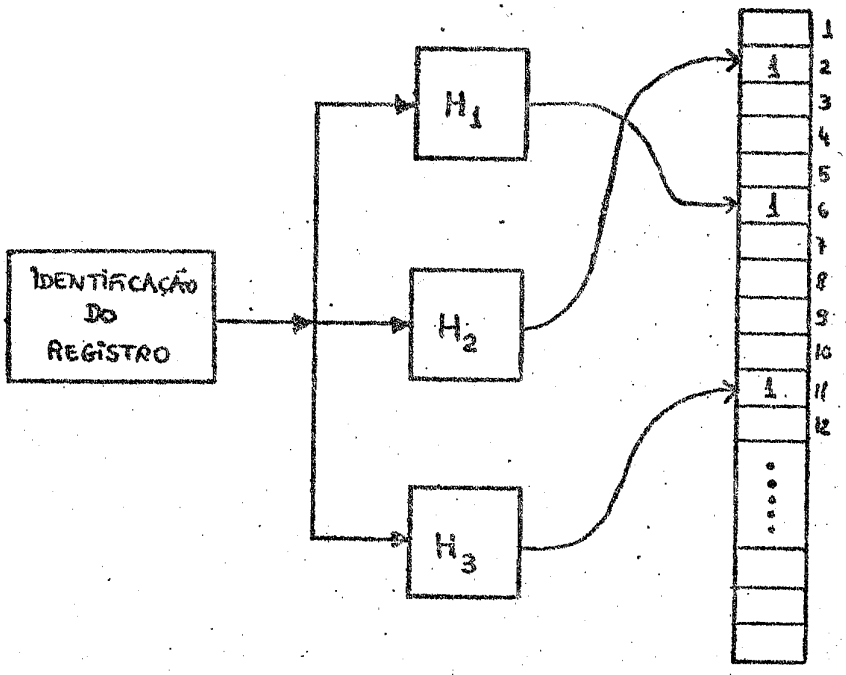


Figura 4 - Filtro com funções hash

se a operação de conjunção lógica (AND) com os X bits resultantes de B. Se o resultado for ∅ o registro não está no arquivo diferencial e se for 1 talvez esteja.

Assim, a recuperação de um registro R é ilustrada pelo diagrama da figura 5.

Severance [5] descreve detalhadamente as vantagens da técnica de arquivos diferenciais. Dentre elas podemos citar:

- redução do custo de dumping
- facilita dumping incremental
- permite dumping dinâmico
- acelera recuperação de perda de dados
- simplifica o desenvolvimento de software

2.7 - Substituição Cuidadosa

O objetivo da substituição cuidadosa é permitir a atualização das estruturas de dados "no local". A atualização é executada em uma cópia da entidade que está sendo modificada, a qual substituirá a entidade original somente no caso da atualização ter sido executada com sucesso. A cópia é mantida até que a substituição da entidade tenha ocorrido completamente.

Na técnica da substituição cuidadosa as duas cópias "virtuais" da mesma entidade, só são mantidas durante a atualização. Isso torna a atualização ou a sequência de atualiza-

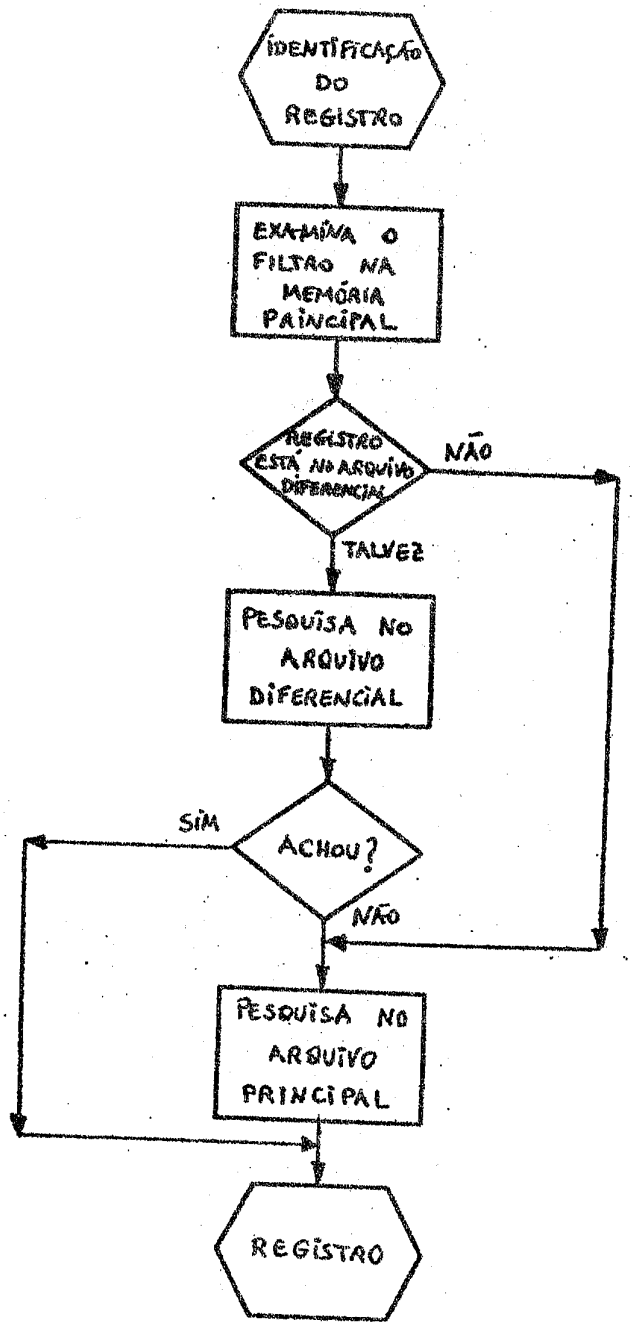


Figura 5 - Recuperação de um registro R

ções tão confiável quanto possível. Essa técnica elimina as desvantagens da técnica de arquivos diferenciais mencionadas no item anterior. Entretanto ela implica em custos adicionais decorrentes de um "overhead" de acessos a disco. Por esta razão os arquivos e as estruturas de dados necessárias para a recuperação devem ser cuidadosamente projetadas a fim de não comprometer o desempenho do sistema.

Serão apresentadas aqui, duas idéias que implementam essa técnica de substituição cuidadosa:

1) Mecanismo de Lampson e Sturgis [7] e

2) Mecanismo utilizado pelo sistema R [6].

2.7.1 - Mecanismo de Lampson e Sturgis [7]

Antes de descrevermos o mecanismo, vamos introduzir o conceito de lista de intenções. Uma lista de intenções é a lista de ações necessárias para efetivar os comandos de atualização de uma transação. Esta lista é armazenada em uma ou mais páginas físicas com propriedade atômica forte, ou seja, no momento da escrita de uma página física que armazena parte de uma lista de intenções associa-se duas páginas físicas livres, digamos i e j. A página é escrita na página física i e caso esta escrita ocorra sem erros, a página física i é copiada sobre a página física j. A lista de intenções deve ser idempotente ,isto é, o resultado da execução ininterrupta da lista de intenções é equivalente ao resultado da execução da

concatenação de listas parciais com uma lista completa. Assim, se L = A1,A2,A3,A4 é uma lista de intenções, temos que:

A1,A2,A3,A4 <=> A1,A2,A3,A1,A2,A1,A1,A2,A3,A4

A fim de que as operações ou ações que compõem uma lista de intenções possam ser repetidas um número qualquer de vezes sem que o resultado da execução da lista de intenções seja alterado, é suficiente que cada ação seja do tipo:

ESCREVA UM VALOR V NO ENDEREÇO E

Para Lampson e Sturgis [7] o BD consiste de um conjunto de arquivos. Um arquivo é composto de uma página de ponteiros e de um conjunto de páginas de dados. A página de ponteiros contêm os endereços das páginas de dados que compõem o arquivo. A figura 6 ilustra essa representação.

Vejamos então, como funciona o mecanismo de Lampson e Sturgis [7] para implementar a técnica de substituição cuidadosa. Considere uma transação que atualiza as páginas de endereço 250 e 900 do arquivo K. Em vez de atualizar diretamente no local as páginas 250 e 900, duas páginas livres são obtidas e os novos valores ou novas versões das páginas a serem atualizadas são construidas nas duas páginas novas. Sejam 300 e 450 os endereços das novas páginas correspondentes as páginas 250 e 900. Nesse ponto, para que a transação seja efetivada, é necessário que as seguintes ações sejam executadas:

A1 = escrever o valor 300 na terceira palavra da página Ø.

A2 = escrever o valor 450 na segunda palavra da página Ø.

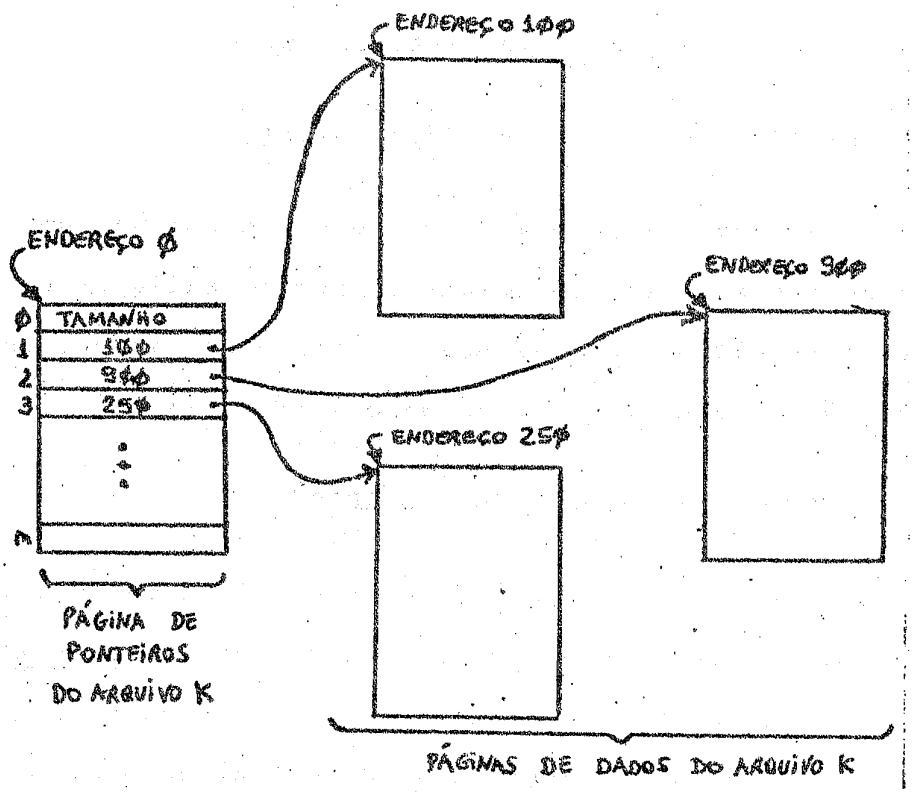


Figura 6 - Representação do arquivo K. Na figura, a página de ponteiros do arquivo K que está no endereço zero, está mapeando tres páginas de dados localizadas nos endereços 100,900 e 250 respectivamente.

As ações A1 e A2 constituem a lista de intenções para a transação em questão.

De uma maneira geral os seguintes passos devem ser seguidos para executar uma transação:

P1: para cada transação escreva páginas de dados modificadas usando páginas livres, à medida que comandos de atualização são recebidos.

P2: ao término da transação

P2.1: crie a lista de intenções

P2.2: escreva a lista de intenções em memória estável com propriedade atômica forte

P2.3: execute a lista de intenções

P2.4: apague a lista de intenções

Observe que após o passo P2.2 a transação pode ser recuperada mesmo que ocorram falhas. Note também, que até a execução do passo P2.3 o BD está refletindo o estado imediatamente anterior ao início da transação.

Em um ambiente de banco de dados distribuído cada nó da rede só executa uma lista de intenções relativa a uma determinada transação se e somente se todos os nós envolvidos com essa transação já tiverem criado suas listas de intenções e se já tiverem escrito essas listas em memória estável com propriedade atômica forte. Seguindo-se essa estratégia garante-se a consistência do banco de dados distribuido, ou seja, ou todas ou nenhuma atualização será executada nos nós da rede envolvidos com a transação, fazendo com que todas as cópias dos dados distribuídas pela rede possuam o mesmo valor.

2.7.2 - Mecanismo Utilizado pelo Sistema R [6]

O mecanismo utilizado pelo sistema R é bastante semelhante ao mecanismo de Lampson e Sturgis [7].

Para Lorie o BD consiste de um conjunto de segmentos (mesma noção de arquivo) Sk com $1 \leq k \leq n$. Cada segmento é composto de um número de páginas lógicas que serão armazenadas em páginas físicas de tamanho idêntico.

Para cada segmento o mapeamento de páginas lógicas em páginas físicas é feito usando-se um vetor VkØ, geralmente chamado de tabela de páginas, que nada mais é do que uma sequência de endereços de páginas físicas. O i-ésimo elemento do vetor VkØ indica que página física está sendo utilizada para armazenar o conteúdo da página lógica i do segmento Sk. Um valor nulo indica uma página indefinida. A figura 7 ilustra essa técnica de mapeamento.

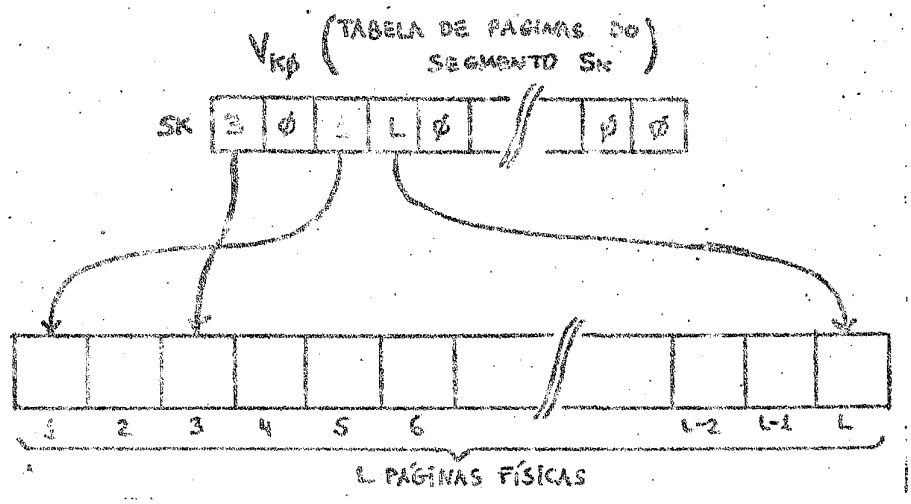


Figura 7 - Mapeamento entre páginas lógicas e páginas físicas.

No mecanismo de Lorie [6], as novas versões das páginas físicas modificadas também são construídas em páginas livres. Para encontrar essas páginas livres, o mecanismo utiliza-se de um vetor MAPØ de L bits (se existe um máximo de L páginas físicas disponíveis no sistema) tal que:

MAPØ[j] = 1 se a página física j está ocupada

MAPØ[j] = Ø se a página física j está livre

Vejamos então, como o mecanismo de Lorie implementa a técnica de substituição cuidadosa. Para poder realizar a atualização no local, o mecanismo utiliza-se de uma segunda tabela de páginas para cada segmento Sk (Vetor Vk1), que serve de cópia da tabela original; e de mais dois vetores de bits (MAP1 e MAP_CORRENTE) com propósitos idênticos ao vetor MAPØ. O vetor MAP1 serve de cópia do vetor MAPØ e o outro é utilizado para representar a situação atual do mapeamento das páginas físicas. (a utilização dos vetores MAPØ, MAP1 e MAP_CORRENTE é explicada adiante com mais detalhes). O mecanismo utiliza ainda um registro de n+1 bits chamado de MASTER com o seguinte formato em "PASCAL-like":

```
type bit = (Ø,1);
     MASTER = record
                  ESTADO: array [1..N] of bit;
                  MAP_CHAVE: bit
              end;
```

Os bits de estado indicam se um segmento está aberto ou fechado. Um segmento Sk está aberto se existe pelo menos uma transação atualizando alguma página lógica do segmento Sk (ESTADO[k] = 1). O segmento Sk está fechado caso contrário (ESTADO[k] = Ø). O bit MAP_CHAVE indica qual vetor de bits está sendo utilizado para representar o último estado de integridade do BD (MAP_CHAVE = Ø indica que o vetor MAPØ está representando o último estado de integridade e MAP_CHAVE = 1 indica que o vetor MAP1 está representando o último estado de integridade).

A figura 8 mostra a representação do BD, com todos os segmentos fechados, utilizando todos esses novos elementos.

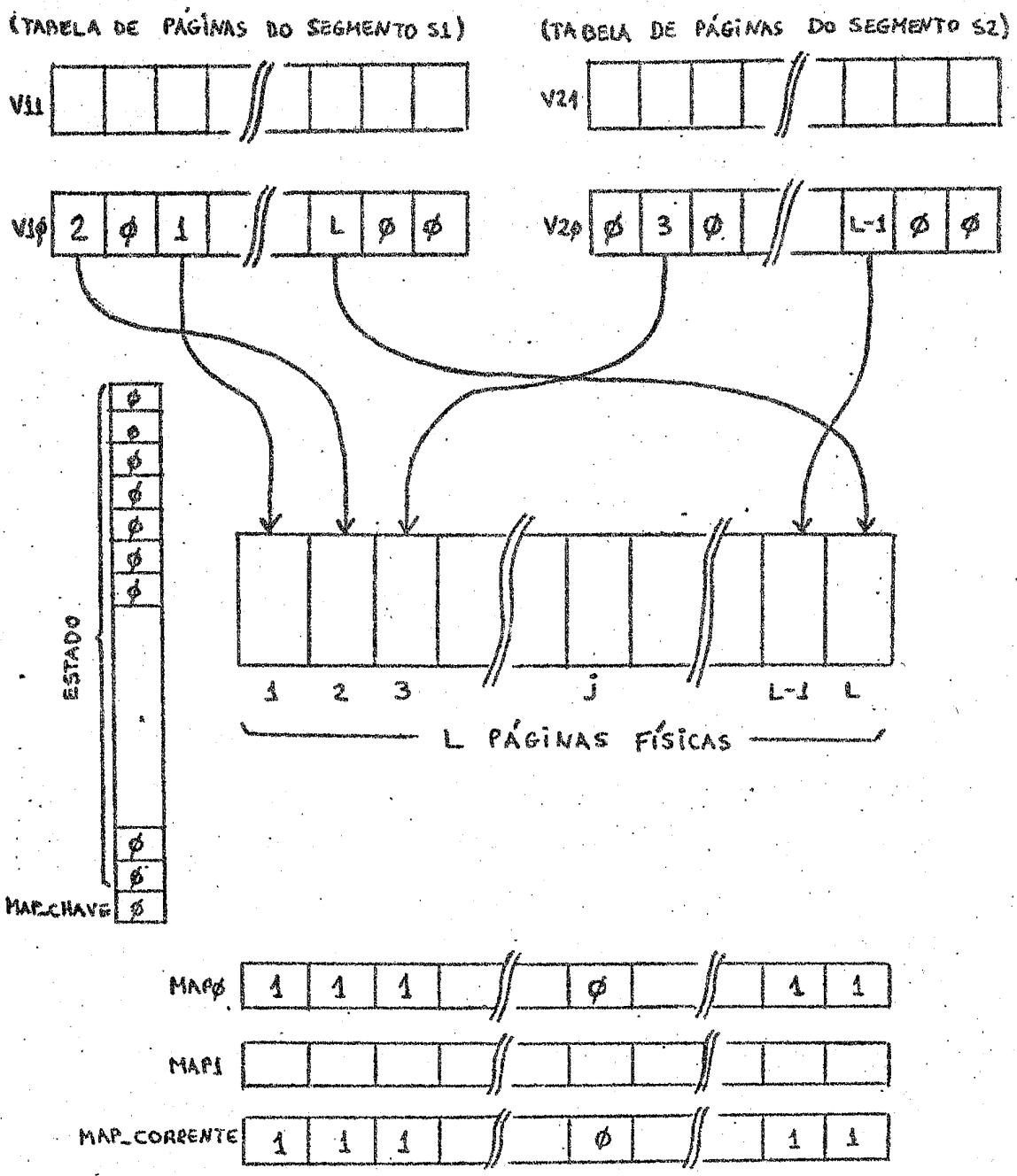


Figura 8 - Representação do BD (Todos os segmentos fechados). Visualização do mapeamento do segmento S1 e S2.

Suponha agora, que queremos modificar a página lógica 3 do segmento S1. Em primeiro lugar devemos verificar se o segmento está aberto (ESTADO[1] = 1). Se não estiver, devemos executar uma operação de "abre segmento S1". Nesse instante a tabela de páginas corrente (Vetor V1Ø) do segmento S1 é copiada para a tabela de páginas V11, e é atribuído o valor 1 ao bit ESTADO[1]. Para acessarmos a página física que contem o conteúdo da página lógica 3 devemos acessar a página cujo endereço está na terceira entrada da tabela de páginas corrente do segmento S1 (no caso, acessaríamos a página física 1, pois V1Ø[3] = 1). Como a nova versão da página 3 será construída em uma página física livre, devemos utilizar o vetor MAP_CORRENTE para encontrar essa página livre (digamos que a página j foi escolhida). Para encerrar a atualização, depois que a nova versão da página lógica 3 foi construída na página física j, devemos fazer com que o terceiro elemento da tabela de páginas corrente do segmento S1 aponte para o novo valor da página lógica 3 (V1Ø[3] = j).

Note que a página física 1 não é liberada e continua sendo apontada por V11[3].

Para efetivar a atualização da página lógica 3 do segmento S1, devemos modificar o primeiro bit do vetor MAPØ (vetor que representa o último estado de integridade do BD) para o valor Ø, indicando com isso que a página física 1 pode ser liberada; e o jésimo bit do mesmo vetor para o valor 1, indicando com isso que a página física j está ocupada. Para evitarmos a possibilidade de uma falha, constrói-se o novo esta-

do de integridade do BD sobre o vetor MAP1. Quando o vetor MAP1 estiver representando o novo estado de integridade devemos fazer com que o sistema reconheça essa nova situação fazendo MAP_CHAVE = 1.

De uma maneira geral, para efetivar todas as atualizações feitas sobre um segmento Sk devemos construir sobre o vetor MAP1 o novo estado de integridade tal que, todos os bits do vetor que correspondem as páginas físicas que contêm as novas versões das páginas lógicas modificadas possuam o valor 1, bem como, todos os bits que correspondem as páginas físicas que contêm as versões originais das páginas lógicas modificadas possuam o valor Ø. Só então, devemos trocar o valor de MAP_CHAVE para 1 (Se o valor original de MAP_CHAVE é igual a 1 a construção do novo estado de integridade é feita sobre o vetor MAPØ).

Note que no MAP_CORRENTE, também devemos fazer com que os bits que correspondem as páginas físicas que contêm as versões originais das páginas lógicas modificadas possuam o valor Ø.

Se uma falha ocorrer antes de efetivarmos as atualizações executadas sobre um segmento Sk e se quisermos voltar ao estado de integridade anterior desse segmento, basta fazermos com que os bits do MAP_CORRENTE que correspondem as páginas físicas que contêm as novas versões das páginas lógicas modificadas possuam o valor Ø; e copiarmos a tabela de páginas Vk1 para a tabela de páginas VkØ. Note que nada precisa ser

feito com os vetores MAPØ e MAP1, pois estes não sofreram nenhuma alteração desde o tempo t que representava o último estado de integridade até o momento da ocorrência da falha.

3.3 - Conclusão

Para finalisar este trabalho apresentaremos a seguir algumas relações que podem ser obtidas entre as diferentes técnicas a partir da descrição realizada.

A figura 9 mostra uma referência cruzada entre as 7 diferentes categorias que agrupam as técnicas de recuperação e os diferentes tipos de recuperação [1].

Eis aqui algumas relações aparentes:

- A técnica de arquivos diferenciais torna muito fácil a implementação de dump incremental.
- A trilha auditora é uma alternativa para as técnicas de arquivos diferenciais, substituição cuidadosa ou duplicação.
- As técnicas de duplicação e substituição cuidadosa podem ser usadas alternativamente ou como complementares. Ambas provêm resistência contra falhas do mesmo tipo.
- As técnicas de dump, da trilha auditora, da cópia e versão corrente e de arquivos diferenciais podem ser utilizadas para prover recuperação de falhas particulares ou para complementarem umas as outras a fim de recuperar o BD após a ocorrência de uma falha.
- O programa restaurador como técnica de recuperação é um último recurso, usado quando todas as outras técnicas falham. O programa restaurador não pode trazer o BD a um estado prévio. Ele meramente "recolhe" o que restou após a ocorrência de uma falha.

	1) UM ESTADO CORRETO	2) UM ESTADO CORRETO ANTERIOR	3) UM POSSÍVEL ESTADO ANTERIOR	4) UM ESTADO VÁLIDO	5) UM ESTADO CONSISTENTE	6) RESISTÊNCIA CONTRA FALHAS
1) PROGRAMA RESTAURADOR				*	*	
2) DUMP			*	*		
3) TRILHA AUDITORA	*	*	*			
4) CÓPIA E VERSÃO CORRENTE		*	*			
5) DUPLICAÇÃO						*
6) ARQUIVOS DIFERENCIAIS		*	*			*
7) SUBSTITUIÇÃO CUIDADOSA		*				*

Figura 9 - Referência cruzada indicando quais os propósitos das diferentes técnicas de recuperação.

: O mecanismo de Lorie [6] e o mecanismo de Lampson e Sturgis [7] implementam a propriedade atômica: ou todas ou nenhuma atualização feita em um segmento [6] ou por uma transação ao BD [7] são efetivadas. Em Lampson isso é conseguido através da execução da lista de intenções e em Lorie é conseguido através da troca do bit MAP_CHAVE de 1 para ∅ ou vice-versa.

REFERÊNCIAS

[1] VERHOFSTAD, J.S.M., "Recovery Techniques For Database Systems", Computer Surveys, Vol 10, No 2, Junho 1978.

[2] ROSENKRANTZ, D.J., "Dynamic Database Dumping",ACM/SIGMOD International Conference on the Management of Data 31 de maio a 2 de junho de 1978, Austin, Texas.

[3] MENASCÉ, D.A., MUNTZ, R.R., "Locking and Deadlock Detection in Distributed Databases", IEEE Transactions on Software Engineering, Maio 1979, pags. 195-201.

[4] GRAY, J.N., "Notes on Data Base Operating Systems", Capítulo 3.F. do livro "Operating Systems An Advance Course", Springer-Verlag, 1978.

[5] SEVERANCE, D.N., LOHMAN, G.M., "Differential Files: Their Application to the Maintenance of Large Databases", ACM Transactions on Database Systems, Vol 1, No 3, Setembro 1976.

[6] LORIE, R.A., "Physical Integrity in a Large Segmented Database", ACM Transaction on Database Systems, Vol 2, No 1, Março 1977.

[7] LAMPSON, B., STURGIS, H.E., "Crash Recovery in a Distributed Data Storage System", Xerox Palo Alto Research Center, Palo Alto, Califórnia, USA, 1978 (a ser publicado na CACM).

[8] DAVENPORT, R.A., "On-line Data Base Integrity", London School of Economics e CACI Inc-International,1977.