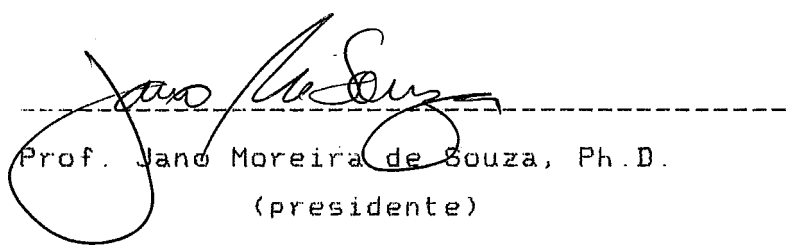
O SUB-SISTEMA DE RECONSTRUÇÃO DO SGBD COPPEREL

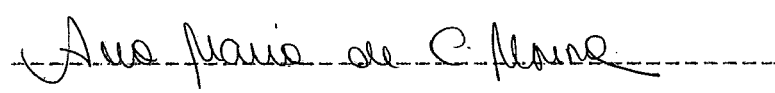
Laércio Antônio Castelo Branco Gonçalves

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

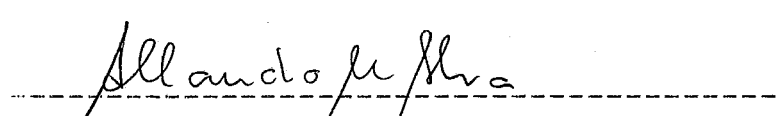
Aprovada por:



Prof. Jano Moreira de Souza, Ph.D.
(presidente)



Prof. Ana Maria de Carvalho Moura, D.I.



Prof. Antônio Cláudio de Carvalho Monteiro da Silva, Ph.D.

RIO DE JANEIRO, RJ - BRASIL
JUNHO DE 1990

GONÇALVES, **LAÉRCIO ANTÔNIO CASTELO BRANCO**

O Sub-Sistema de Reconstrução do SGBD COPPEREL. [Rio de Janeiro] 1990.

XII, ***317*** p. 29,7 cm (COPPE/UFRJ, M.Sc., **Engenharia de Sistemas e** Computação, 1990)

TESE - Universidade Federal do Rio de Janeiro, COPPE.

1. Reconstrução **em Bancos de Dados I.** COPPE/UFRJ II. **Titulo** (série).

AGRADECIMENTOS

Antes de tudo quero agradecer à Deus as oportunidades e o amparo ao longo de minha existência.

Aos meus pais pelo esforço e exemplo na minha formação.

À Dória pelo seu amor e paciencia

Ao Jano Moreira de Souza pela oportunidade deste trabalho e pela orientação.

Ao Claudio Newton Ferreira Trotta pela co-orientação e idéias preciosas na implementação do trabalho.

À Marta Lima de Queirós Mattoso pela sua ajuda perene ao longo das cadeiras e na confecção da tese.

Ao Guilherme Horta Travassos e ao Luis Carlos Montez Monte pela ajuda sempre que necessária.

À FUCAPI (Fundação Centro de Análise, Pesquisa e Inovação Tecnológica) pelo apoio financeiro recebido durante o curso.

Aos membros da banca, por terem aceito o convite e me honrado com sua participação.

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

O SUB-SISTEMA DE RECONSTRUÇÃO DO SGBD COPPEREL

Laércio Antônio Castelo Branco Goncalves

Junho, 1990

Orientador Jano Moreira de Souza

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta os principais conceitos envolvidos na definição de um sub-sistema de reconstrução de banco de dados, estuda as tecnicas de reconstrução mais utilizadas, descreve alguns sistemas e finalmente, apresenta a proposta e os resultados da implementação do sub-sistema de reconstrução desenvolvido para o Sistema de Gerência de Banco de Dados (SGBD) COPPEREL-PC envolvendo as tecnicas de diário e descarga. O sub-sistema de reconstrução proposto e implementado recupera falhas de transação, de sistema e de meio de armarenamento.

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

THE RECOVERY ÇUBSYSTEM OF THE DBMS COPPEREL

Laércio Antônio Castelo Branco Gonçalves

June, 1990

Thesis Supervisor: Jano Moreira de Souza

Department: Engenharia de Sistemas e Computação

This work presents the main concepts involved in the definition of a data base recovery subsystem, reviews the most used recovery thechniques, describes some other systems and presents a proposal and the solution adopted in implementation of the recovery subsystem developed to the Data Base Management System (DBMS) COPPEREL-PC. The recovery subsystem proposed implements transaction failure, system failure and media failure recovers.

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO I

INTRODUÇÃO

Atualmente, devido a importância da informacão na maioria das organizações, o banco de dados é um recurso extremamente valioso. Hoje, mais e mais organizações confiam em sistemas de processamento de dados para gerenciar dados críticos do seu dia a dia. Em alguns casos, quando os dados não estão disponíveis, são inconsistentes ou contem erros (mesmo por pequenos períodos de tempo), interferem seriamente nas operações normais do negócio gerando, quase sempre, significantes prejuízos.

Um banco de dados representa o estado de uma parte do mundo real. Este não é estático, está sofrendo alterações a cada dia, as quais devem refletir-se no banco de dados para que este esteja sempre num estado consistente, reproduzindo a realidade atual.

Sistemas de banco de dados são concebidas para gerenciar grandes quantidades de informação. O gerenciamento de dados envolve tanta a definição de estruturaç para armaxenamento da informação coma a provisão de mecanismos para manipula-la. Além disso, o sistema de banco de dadas deve proporcionar a segurança das informações armazenadas no banco de dados, mesmo em caso de quedas no sistema ou de tentativas de acessos desautorizados. Se os dados forem compartilhados por diversos usuários, o sistema também deve impedir possíveis resultados anômalos (KORTH 1989).

Porem, não e uma tarefa simples alcançar estes objetivos, mesmo que tivessemos hardware e software perfeitos, ainda assim o sistema poderia falhar ocasionamente, porque as pessoas que o adaptam ao ambiente podem cometer erros (erros de programação de aplicação) e

as pessoas que operam o sistema também podem cometer erros (entradas de dados e erros de procedimentos) (GRAY 1981a).

Pode-se tirar duas conclusões disso:

a) Não é necessário fazer um sistema perfeito (totalmente imune a falhas)

b) Mesmo que o sistema seja considerado perfeito, alguns erros poderão acontecer.

Erros em dados podem ser divididos em dois tipos, erros lógicos e erros físicos (CRUS 1984). Erroç lógicos incluem atualizações sue deveriam ser feitas no banco de dados e não foram, atualizações que foram feitas mas não deveriam ter sido e atualizações que foram feitas incorretamente. Erros deste tipo podem ser causados por falha do sistema, erros no gerenciador do banco de dados, erros em vários componentes críticos da sistema e ate erros em programas de aplicação sue usam o gerenciador de banco de dados. Erros físicos são causados par mal funcionamento do hardware.

Antes de falarmos sobre falhas e importante relembrarmos dois conceitos básicos (KORTH 1989):

a) integridade de um sistema é a medida de sucesso pela qual o sistema submete-se à alguma especificação de comportamento. Sem essa especificação, nada pode ser dito sobre a integridade do sistema. Quando o seu comportamento se desvia de sua especificação, acontece o que chamamos de falha. Uma falha é um evento. A integridade do sistema é inversamente relacionada com a frequência destes eventos.

b) disponibilidade é a fração de tempo que um sistema atende a sua especificação.

Infelizmente camputadores e "softwares" não são imunes a falhas. Por isso, sistemas precisam incorporar não apenas

várias verificações e controles para reduzir a probabilidade de falhas, mas também, e mais significamente, um conjunto de procedimentos para recuperação de falhas que inevitavelmente irão ocorrer, apesar das verificações e controles (JONES 1987).

Uma parte integrante de um sistema de banco de dados é um esquema de recuperação, que é responsável pela detecção de falhas e pela restauração do banco de dados para um estado consistente que existia antes da ocorrência da falha.

Segunda Haerder e Reuter em (HAERDER 1983), para se entender os conceitos de recuperação de banco de dados é necessário primeiro compreender dois Fatores:

a) os tipos de falhas que o banco de dados deve suportar; e

b) a noção de consistência que é assumida como critério para descrever o estado a ser restabelecido.

Há vários tipos de falhas que podem ocorrer em um sistema, cada qual necessitando ser solucionado de um modo diferente. O tipo mais simples de falha a ser solucionado é aquele que não resulta em perda de informação no sistema. Os que são mais difíceis de serem solucionados são aqueles que resultam em perda de informação.

A recuperação em um sistema de banco de dados significa, essencialmente, recuperar o próprio banco de dados, isto é, restaurar o banco de dados a um estado que se saiba estar correto após alguma falha ter deixado o estado atual incorreto, ou pelo menos suspeito. Como já foi dito, existem muitas causas de falhas (erros de programação em uma aplicação, ou no sistema operacional, ou no próprio sistema de banco de dados, erros no dispositivo de hardware ou no canal ou na CPU, erros de operação, flutuações no fornecimento de energia eletrica, incêndio, sabotagem, etc). Em todos os casos, os princípios subjacentes em que

se baseia a recuperação são bastante simples, e podem ser resumidos em uma única palavra: **redundância. Isto é,** o meio de proteger o banco de dados e assegurar que qualquer informação nele existente possa ser reconstruida a partir de alguma outra informacão armazenada redundantemente, em algum outro lugar do sistema.

Os vários tipos de falhas que podem ocorrer no banco de dados podem causar desde situações amenas e imediatamente contornáveis até catástrofes que deixam marcas irrecuperáveis, destruindo dados e procedimentos do sistema. O problema e agravado pela imprevisibilidade das Palhas r, ainda, pelo fato de que novos desarranjos podem surgir enquanto o sistema de banco de dados está se recuperando de outros anteriores, caracteritanda múltiplas falhas simultâneas (CASANOVA 1985). É importante notar, desde já, que não há proteção total contra todas as falhas que eventualmente podem vir a acontecer. O objetivo dos mecanismos de proteção é tornar o sistema de banco de dados mais e mais confiável e seguro a medida que se tornam mais sofisticados. No entanto, alguns casos sempre podem ser imaginados de tal forma que não possam ser tratados a contento pelos mecanismos de proteção. Não deve ser esquecido, tambem, o aspecto da eficiência do sistema de banco de dados como um todo. Dispor de um sistema extremamente seguro e confiável, porém a um custo muito alto, inviabiliza o próprio sistema de banco de dados, na medida que não será posto em operação devido a sua impraticabilidade. O desafio esta em se imaginar técnicas, estruturas e procedimentos que dêem proteção razoavelmente eficaz contra as falhas mais obseruadas na pratica e que não causem muito impacto nas atividades do usuário.

Uma possível abordagem para recuperação é distinguir os diferentes tipos de Calhas baseando-se em dois critérios (VERHOFSTAD 1978):

a) a extensão da falha;

b) a causa da Calha.

Os três tipos de falhas mais comuns em sistemas de banco de dados são: a falha de transação, a falha de sistema e a falha de meio de armazenamento. Para cada um desses tipos são usados diferentes procedimentos de recuperação.

A principal abordagem usada por muitos bancos de dados comerciais disponíveis se baseia no requisito do sistema ser capaz de desfazer, refazer ou completar transações (seção III.1). Uma transação e considerada como uma unidade lógica de trabalho, uma unidade de bloqueio e também uma unidade de recuperação, aparecendo para o usuário como uma ação atômica.

Segundo Landes em (LANDES 1980), a existência de diferentes tipos de falhas requer a existência de diferentes mecanismos e técnicas, bem como o uso combinado destes para levar o banco de dados novamente a um estado de integridade. Algumas técnicas são usadas para restaurar o banco de dados para um estado previamente consistente (em algum tempo passado), outras para tolerar a ocorrência de falhas e, outras para retornar o banco de dados ao último estado consistente e imediatamente anterior a falha.

Para que a recuperação de falhas seja possível, é necessário que dados que possibilitem restaurar o banco de dados ao estado em que se encontrava no momento da falha sejam armazenados. Estes dados de recuperação também estão sujeitos a Calhas e também devem ser protegidos.

I.1 - Antecedentes

O Programa de Engenharia de Sistemas da COPPE/UFRJ desenvolveu, através de um grupo de pesquisadores desta instituição, um SGBD baseado no modelo relacional denominado COPPEREL (Sistema de Gerência de Base de Dados da COPPE baseado em álgebra relacional) que começou a

operar em 1983 e desde então tem servido de base para novos estudos e pesquisas no âmbito acadêmico.

O COPPEREL é um sistema mono-usuário dedicado, onde o usuário pode criar, alterar ou excluir bases de dados locais residentes num dispositivo de memória secundária, não existindo possibilidade de vários usuários se conectarem à uma mesma cópia do COPPEREL e acessarem uma mesma base de dados (PALERMO 1985).

A linguagem de operação do SGBD COPPEREL - LOPEREL, é uma linguagem autocontida baseada na álgebra relacional com comandos em português. Reúne comandos da linguagem de definição de dados (LDD), comandos da linguagem de manipulação de dados (LMD) e, possui também, comandos herdados das linguagens de programação tradicionais.

O SGBD COPPEREL possui uma arquitetura modular que permite a realização de extensões. O sistema foi implementado através de dois módulos principais: uma máquina virtual e um compilador da linguagem LOQEREL.

A partir de 1987 o COPPEREL começou a ser adaptado para uma versão PC como núcleo do projeto Engenharia de Dados (SOUZA 1987). Esta versão chamada de COPPEREL-PC vem sendo utilizada para receber extensões e atender a classes de aplicações específicas.

Atualmente, vários projetos e experiências utilizam a versão PC do COPPEREL. Na área de banco de dados, existem trabalhos como o que propõe adaptações ao COPPEREL para lidar com aplicações não convencionais (TROTTA 1989) e os que propõem e implementam os conceitos de banco de dados orientados a objetos usando o COPPEREL-PC como base (BLUM 1989a) e (BLUM 1989b).

Além desses, existem trabalhos anteriores como o de (PALERMO 1985) que propõe um método de controle de concorrência para o COPPEREL e o de (MATTOSO 1987) que

propõe e incorpora ferramentas de apoio aos usuários desse SGBD. Todos esses trabalhos são oriundos do COPPEREL, e são exemplos da importância do desenvolvimento de um projeto como este, que visa antes de mais nada a apredizagem, o domínio do conhecimento e a auto-suficiência tecnológica.

Maiores informações a respeito da origem do COPPEREL, de sua arquitetura e principais características podem ser encontrados em (MATTOSO 1985a), (MATTOSO 1985b), (MATTOSO 1987), (PALERMO 1985), (ZAKIMI 1982) e (ZAKIMI 1985).

I.2 - Motivação

Na primeira versão do COPPEREL, no inicio de 1983, nem todas as características previstas na sua especificação inicial foram implementadas. Além disso, outras características que não existiam na especificação original mas que eram exigidas pelos usuários precisavam existir para tornar o software mais abrangente e flexível (MATTOSO 1987).

Através do trabalho de Mattoso (MATTOSO 1987), foi efetuado o levantamento, a avaliação, a critica e várias correções das principais características do COPPEREL.

O principal fator de motivação para a realização deste trabalho é o fato de que ainda hoje, em sua versão mais atual "COPPEREL-PC", o sistema não apresenta as características fundamentais para assegurar a consistência dos dados: a existência de um conjunto de procedimentos de recuperação de falhas (reconstrução) e o conceito de transação.

1.3 - Objetivos da Trabalho

Este trabalho tem como objetivos fazer um estuda conceitual sobre transacão e reconstrução, mostrar algumas técnicas usadas em reconstrução, apresentar uma proposta de um sub-sistema de reconstrucão para a COPPEREL-PC englobando o conceito de transação, e, finalmente, implementar no COPPEREL-PC o conceito de transacão e o sub-sistema de reconstrução proposto.

I.4 - Organização da Tese

O capítulo 11 contém a descrição de um modelo de dados abstrato de um sistema de banco de dados dando ênfase a descrição do seu módulo mais importante para nosso estudo, o Gerenciador de Dados. A apresentação deste modelo tem somo objetivo propiciar um embasamento sobre os principais aspectos de banco de dados relacionados com transação e reconstrução.

O capitulo 111 fala dos elementos necessários para definição dos sub-sistemas de reconstrução encontrados com mais frequência nos sistemas de banco de dados atuais, abordando os conceitos de transação, falhas e controle de acesso concorrente. Neste capitulo também descrevemos alguns conceitos e procedimentos que se relacionam com o sub-sistema de reconstrucão e apresentamos várias técnicas de reconstrucão bem como sugestões de como combiná-las para criar um sub-sistema de reconstrucão.

O capitulo IV trás exemplos de sub-sistemas de reconstrução existentes. Para tal, são descritos dois sistemas relacionais (Sistema-R e INGRES) e três propostas de sistemas orientados a objetos (CACTIS, GemStone e POSTGRES) enfatizando-se sempre que possível o aspecto de reconstrução desses sistemas. Neste capítulo também são descritos os principais aspectos relacionados com bancos de dados distribuídos e como o sub-sistema de reconstrução funciona nesta abordagem.

No capitulo V é apresentada uma proposta de um sub-sistema de reconstrução para o COPPEREL-PC e os motivos que nos levaram a escolher determinados conceitos e técnicas.

O capitulo VI fala da implementação da proposta apresentada no capitulo V, da estratégia de implementação utilizada, do que foi implementado, como foi implementado, das principais características da implementação e das modificações realizadas no CQPPEREL-PC.

Finalmente, no capitulo VII são apresentadas as conclusões, onde faremos uma avaliação da proposta implementada, falamos sobre as possíveis contribuições do trabalho e damos algumas sugestões para futuros trabalhos no COPPEREL-PC relacionados com o tema aqui abordado.

O apêndice contém os resultados de simulações e testes do sub-sistema de reconstrução proposto e implementado no CQPPEREL-PC.

CAPITULO II

MODELO ABSTRATO DE UM SISTEMA DE BANCO DE DADOS

O objetivo principal deste capítulo e apresentar um modelo abstrato da estrutura interna de um sistema de banco de dados (BERNSTEIN 1987) que dará subsídios para um melhor entendimento do restante do trabalho. Aqui, problemas ligados a reconstrucão como tipos, tamanhos e custos de memória, gerenciamento de "buffers" e principalmente gerenciamento de dados (devido a manipulacão das memórias que podem ser corrompidas por falhas) são discorridos.

II.1 - O Modelo Abstratõ

Para um estudo mais particularizado de reconstrução, usaremos um modelo abstrato da estrutura interna de um sistema de banco de dados apresentado em (BERNSTEIN 1987). Segundo Bernstein este modelo não corresponde à arquitetura de nenhum software conhecido, sendo usada somente com objetivo pedagógico. Neste modelo, um sistema de banco de dados consiste de quatro módulos (figura I.I : um gerenciador de transação, que executa qualquer preprocessamento necessário do banco de dados e operações sue ele recebe das transações; um scheduler, que controla a ordem relativa na qual o banco de dados e as operações de transação são executadas; um gerenciador de recuperação, que é responsável pelo comprometimento e aborto das transações; e um gerenciador de "buffer", que opera diretamente sobre o banco de dados.

Cada modulo envia pedidos para e recebe respostas do próximo módulo de nível mais baixo. Assim, operações de banco de dados e operações emitidas por transações para o sistema de banco de dados são recebidas primeiro pelo

gerenciador de transação indo depois para o scheduler, para o gerenciador de recuperação e para a gerenciador de "buffers".

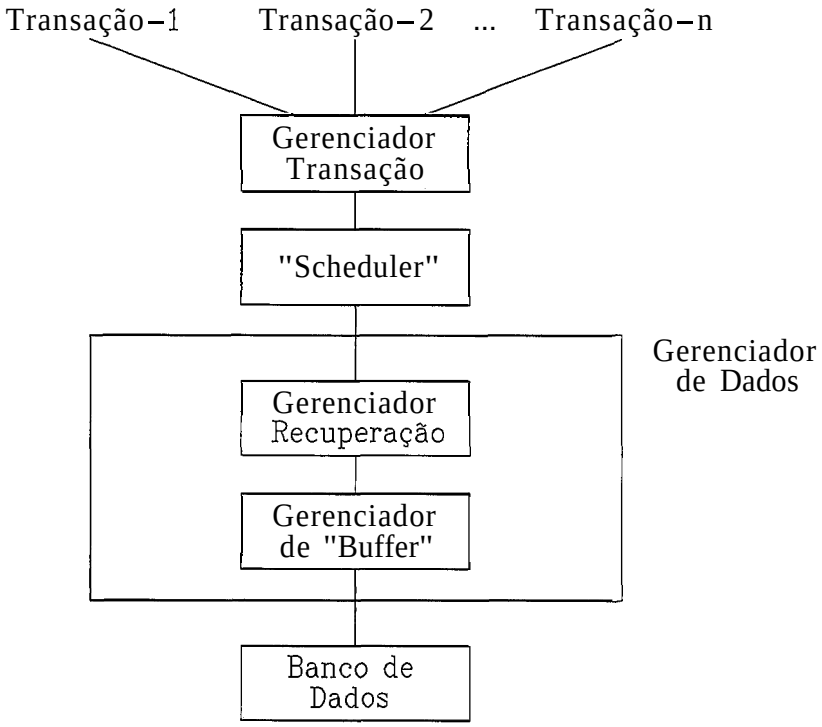


Figura II.1 - Modelo Abstrato de um Sistema de Banco de Dados Centralizado.

II.1.1 - O Gerenciador de "Buffer"

Um sistema de computação usualmente possui tanto memória volátil como memória estável. Memória volátil (memória principal e cachê) pode ser acessada de forma muito eficiente, mas é suscetível a falhas de hardware e de sistema (a seção III.2 aborda os tipos de falhas existentes). Devido a seu custo relativamente alto tem seu tamanho limitado. Memória estável (disco e fita magnética) é resistente a falhas (com alguma precaução), sendo sue o acesso a esse tipo de memória é muito mais lento em relação ao primeiro. Em decorrência de seu custo relativamente baixo a memória estável é usualmente abundante.

O sistema de banco de dados mantem somente uma parte do banco de dados na memória principal de cada vez devido ao seu tamanho limitado. A porção da memória volátil separada para manusear partes do banco de dados é chamada de cachê. Este tipo de memória é a mais rápida e mais cara forma de armarenamento, seu tamanho é pequeno e gerenciar seu uso e função do Gerenciador de "Buffer". O Gerenciador de "Buffer" move dados entre memória estável e memória volátil respondendo a pedidos de camadas mais altas do sistema de banco de dados.

II.1.1.1 - O Gerenciamento de "Buffer"

No intuito de facilitar a troca de dados entre disco e memória principal, devem existir "buffers" próprios do banco de dados para interfacear a memória principal com a memória estável (EFFELSBERG 1984). Neste sentido, o banco de dados é dividido em paginas de igual tamanho (geralmente de 512 a 4086 bytes). O "buffer" consiste de páginas de mesmo tamanho, sendo que o número de páginas do "buffer" pode ser determinado por um parâmetro do SGBD que permanece constante durante uma sessão do SGBD.

Uma ver que um acesso a uma página do banco de dados em memória estável é muito mais caro que um acesso a uma página no "buffer", o principal objetivo de um gerenciador de "buffer" é minimizar o I/O físico, maximizando a probabilidade de que quando um bloco seja acessado ele já esteja na memória principal, não sendo necessário acessar a memória estável.

O gerenciador de "buffer" intercepta todos os pedidos do sistema de armarenamento do banco de dados. Se o bloco já está no "buffer", é passado ao requisitante o endereço do bloco na memória principal. Se o bloco não está no "buffer", o gerenciador de "buffer" lê a bloco do disco

para o "buffer", e passa o endereço do bloco na memória principal para o requisitante.

Especificamente, o gerenciador de "buffer" suporta as operações "fetch (x)" que recupera x da memória estável para a memória volátil, e "flush (x)" que transfere a cópia de x de memória volátil para a memória estável (figura II.2).

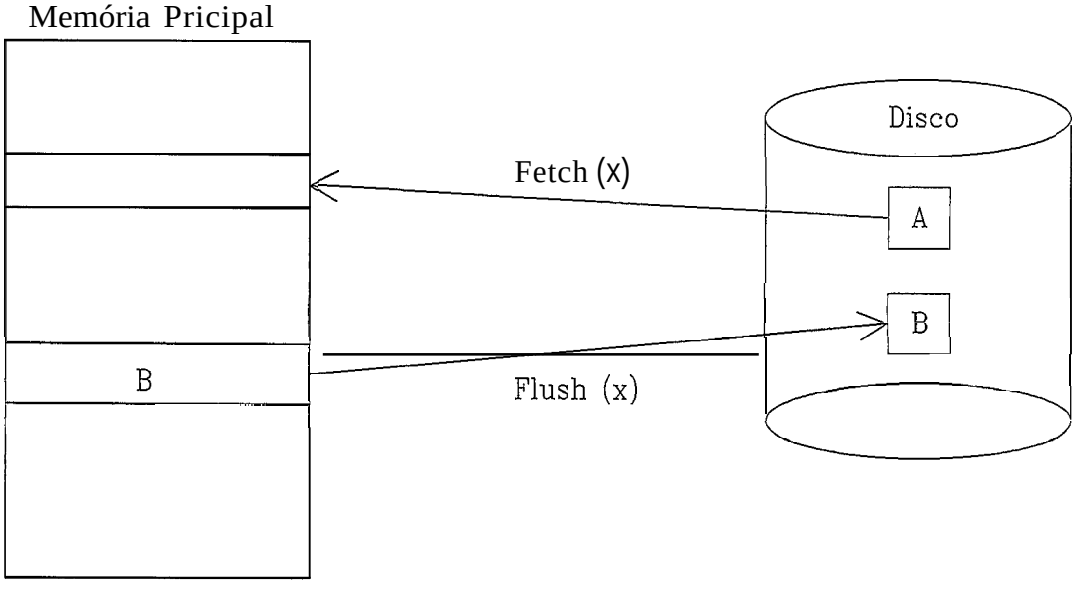


Figura II.2 - Operações de Armazenamento.

De uma forma geral, a gerência de "buffer" se parece com a gerência de memória virtual encontrado na maioria dos sistemas operacionais. Porém, de forma a servir bem ao sistema de banco de dados, ela deve usar técnicas mais sofisticadas, entre elas podemos citar (KORTH 1989):

a) Estratégia de reposição

Existem momentos que o gerenciador de "buffer" não pode processar a operação "fetch" devido a falta de espaço na memória volátil. Neste caso, quando não há espaço suficiente no "buffer", um bloco deve ser removido antes que um novo bloco seja lida. Entre as estratégias

de reposição mais conhecidas temos a "menos recentemente usada" ("least recently used" ou LRU) na qual o bloco que foi referenciado pela ultima vez há mais tempo e gravado de volta no disco e removido do "buffer", e a "primeiro a entrar - primeiro a sair" ("first-in-first-out" ou FIFO), onde o bloco mais antigo é o escolhido para ser gravado de volta no disco.

b) Blocos apontados

Para que um sistema de banco de dados possa ser capaz de se recuperar de falhas, é necessário que se restrinja, às vezes, que um bloco deva ser gravado no disco, por exemplo, enquanto estiver atualizando o conteúdo de um bloco. Por esta razão, o gerenciador de "buffer" oferece duas operações adicionais, "Pin" (diz ao gerenciador de "buffer" para não fazer o "flush" no bloco apontado) e "Unpin" (desfaz a operação "Pin" tornando o bloco novamente disponível para "flush"). Assim, o gerenciador de ""buffer" nunca faz um "flush" num bloco apontado (que sofreu uma operação "Pin").

c) Saída forçada de blocos

Há situações nas quais e necessária a gravação do bloco de volta ao disco, muito embora o espaço ocupado no "buffer" não seja necessário. Isto é chamado de saída forçada de um bloco. O pedido é devido ao fato de que o conteúdo da memória principal, e em consequência o conteúdo do "buffer", se perde com a ocorrência de uma falha, enquanto o conteúdo do disco normalmente sobrevive após uma falha.

Maiores informações sobre gerenciamento de "buffer" podem ser encontradas em (EFFELSBERG 1984).

Além do conhecimento que o sistema deve ter acerca do pedido sendo processado, o gerenciador de "buffer" pode utilizar informações estatísticas considerando a probabilidade que um pedido referenciará uma relação particular. O esquema é a parte mais frequentemente acessada do banco de dados. Assim, o gerenciador de "buffer" deve tentar não remover os blocos do esquema da memória principal. No caso de índices de arquivos, que são acessados mais frequentemente que os próprios arquivos, o gerenciador de "buffer", em geral, também deve evitar remover blocos de índices da memória principal.

A estratégia utilizada pelo gerenciador de "buffer" para a reposição de blocos é influenciada por outros fatores além do tempo em que o bloco será referenciado novamente. Se o sistema está processando pedidos de diversos usuários concorrentemente, o sub-sistema de controle de concorrência pode precisar retardar certos pedidos de forma a assegurar a preservação da consistência do banco de dados. Se o gerenciador de "buffer" é alimentado com informações do sub-sistema de controle de concorrência, ele pode usar esta informação para mudar sua estratégia de alteração de blocos (blocos que são necessários a pedidos ativos podem ser retidos no acesso ao "buffer" as custas de blocos necessários por pedidos retardados).

Um sub-sistema de reconstrucão impõe controles mais rígidos na reposição de blocos. Se um bloco foi modificado, não é permitido ao gerenciador de "buffer" gravar a nova versão do bloco de volta ao disco, visto que destruirá a versão antiga. Em vez disto, o gerenciador de "buffer" deve pedir permissão ao sub-sistema de reconstrução antes de gravar o bloco. O sub-sistema de reconstrução pode pedir que certos outros blocos sejam forçados a sair da memória, antes de conceder ao gerenciador de "buffer" a permissão para gravar o bloco.

II.1.1.2 - Operações de Armazenamento

Na seção anterior vimos as duas operações responsáveis pela movimentação de blocos entre memória estável e memória principal "fetch" e "flush", bem como as duas operações adicionais "Pin" e "Unpin" oferecidas pelo gerenciador de "buffer" para controlar convenientemente a movimentação desses blocos. Nesta seção são apresentadas as operações que permitem a interação entre os programas de aplicação com o sistema de banco de dados. As operações são as seguintes (KORTH 1989):

1) read (X,xi), que determina o valor do item de dados X para a variável local xi.

2) write (X,xi), que atribui o valor da variável local xi para o item de dados X no bloco "buffer".

Como visto anteriormente, um bloco "buffer" é eventualmente regravado no disco ou porque o gerenciador do "buffer" precisa do espaço de memória para outros propósitos ou porque o sistema de banco de dadas quer refletir a alteração de X no disco. Diremos que o sistema de banco de dados forca saídas para o bloco "buffer" de X, se ele executa um flush (X).

Quando um programa necessita acessar um item de dados X pela primeira vez, deve executar read (X,xi). Todas as atualizações de X são então executadas em xi. Depois que o programa acessa X pela Ultima vez, deve executar write (X,xi) a fim de refletir a alteração de X no próprio banco de dados.

A operação flush (X) não precisa ocorrer imediatamente após a write (X,xi), uma ver que o bloco no qual X reside pode conter outros itens de dados que ainda estão senda acessados. Desse modo, a saída real ocorre mais tarde. Observe que se o sistema pára após a execução da operação

write (X,xi) mas antes da operação flush (X), o novo valor de X não será regravado no disco e fica, consequentemente perdido. Situações como essa são abordadas na seção 111.5.

II.1.2 - O Gerenciador de Recuperação

0 Gerenciador de Recuperação tem como principal responsabilidade açsesurar que o banco de dados contenha todos os efeitos das transações compromissadas e nenhum dos efeitos das abortadas (BERNSTEIN 1987).

0 gerenciador de recuperação deve ser projetado para ser resistente a falhas que causam a perda do conteúdo da memória volátil. Estas falhas são chamadas de falhas de sistema (seção III.2.2). Assim, o gerenciador de recuperação deve assegurar que, após uma falha de sistema, a banca de dados contém as efeitos de todas as transações compromissadas e nenhum efeito das transações que foram abortadas ou estavam ativas na hora da falha. Devido a perda do conteúdo da memória principal, as transações que estavam ativas na hora da falha devem ter seus efeitos eliminados, uma ver que elas perdem seu estado interno ficando impossibilitadas de terminarem suas execuções.

Após a ocorrência de uma falha onde o conteúdo da memória volátil é perdido, só resta ao gerenciador de recuperação o conteúdo da memória estável. Por esta razão, o gerenciador de recuperação usualmente armazena infomações em memória estável além das informações do banco de dados estável. Um diário é uma das maneiras de manter as informações necessárias (a técnica de diário é discutida na seção III.6.3).

Cwmo o gerenciador de recuperação não sabe quando vai ocorrer uma falha, ele deve se prevenir para evitar uma das duas situações não recuperáveis:

1) quando a memória estável não contem uma atualização de alguma transação compromissada; ou

2) quando a memória estável contem o valor de um objeto escrito por alguma transação não compromissada, mas não contém o Último valor do objeto que foi escrito por uma transação compromiçsada.

Para evitar esses problemas, em qualquer estatégia de substituição usada pelo gerenciador de "buffer", existem situações que o gerenciador de recuperação pode insistir para que o gerenciador de "buffer" faça o "flush" de certos tipos de itens de dados para memória estavel. O gerenciador de recuperação também pode restringir as situações nas quais o yerrntiador de "buffer" pode unilateralmente decidir executar uma operação "flush" como já visto na seção II.1.1.1. Estas operações de "flush" coordenam a gravação em memória estável para que o reinicio sempre ache as informações que ele necessita.

O gerenciador de recuperação também deve ser projetado para ser resistente a falhas de porções de memória estável, chamadas de falhas de meio de armazenamento (seção III.2.3). Um dos modos mais utilizados para se alcançar este objetivo é guardar cópias redundantes dos dados em pelo menos dois dispoçitivos de memória estavel diferentes, ou seja, com probabilidade mínima de falharem ao mesmo tempo.

O fato de um reinício poder interromper a sua própria execução, faz com que seja necessário um importante requisito para esta operação: ela deve ser idempotente, o que significa que qualquer sequência de execuções de reinício incompletas seguidas de uma execução completa tem o mesmo efeito que a execução de somente um reinício completo.

II.1.3 - O "Scheduler"

Aproveitando a definisão de Bernstein em (BERNSTEIN 1987), temos um "scheduler" como sendo um programa ou uma coleção de programas que controla(m) a execução concorrente de transações. Este controle é exercido pelo "scheduler" pela restrição da ordem na qual o gerenciador de dados executa ar operações de leitura, gravação, comprometimento r aborto de diferentes transações. Seu objetivo é ordenar estas operações de tal forma que a execução resultante seja serializável (tenha o mesmo efeito de execuções seriais, ou seja, uma não interfere na outra) e recuperável.

Para executar uma operação de banco de dados, a transação passa a operação para o "scheduler". Depois de receber a operação, o "scheduler" pode tomar uma das três ações:

a) Executar: a operação é passada para o gerenciador de dados que informa ao "scheduler" quando ele termina de executá-la. Entretanto, se a operação for uma leitura, o gerrnciador de dados retorna o(s) valor(es) lido(s), os quais o "scheduler" devolve para a transação.

b) Rejeitar: o processamento da operação é recusado pelo "scheduler". Neste caso, ele diz à transação que a operação foi rejeitada, o que causa o aborto da transação.

c) Adiar: a operação e adiada e por isso colocada numa fila interna do "scheduler". Mais tarde, ele pode remover a operação da fila para executá-la ou rejeitá-la. Neste intervalo de tempo (enquanto a operação está sendo adiada), o "scheduler" esta livre para tratar outras operações.

Usando estas três ações básicas, o "scheduler" pode controlar a ordem da execução das operações para produzir execuções corretas. Quando ele recebe uma operação da transação, usualmente tenta passá-la direto para o

gerenciador de dados. Se ele decidir que a execução da operação pode produzir um resultado incorreto, então ele adia a operação (se ele for capaz de processar corretamente a operação no futuro) ou rejeita-a (se não Cor capaz de processar corretamente a operação no futuro).

Com isso, podemos definir que a parte de sistemas de banco de dados que controla a ordem relativa na qual operações no banco de dados pedidas por execuções de transações e chamada de "scheduler", e este determina a intercalação das operações do banco de dados, preservando assim a consistência do banco de dados.

II.i.4 - 0 Gerenciadar de Transação

A interação entre transações e o sistema de banco de dados e feita através do Gerenciador de Transação.

Das atividades do gerenciador de transação, a mais importante e a implementação do conceito de ãtomicidade. Esta propriedade é caracterizada por uma sequência de comandos bem delimitada de tal sorte que o sistema deve garantir que, ou todos os comandos na sequência sejam completamente executados, ou o banco de dados não reflete o resultado da execução de nenhum deles, mesmo que o sistema falhe antes da completa execução de todos os comandos da sequência. 0 gerenciador de transações, portanto, transforma o produto final do processador de consultas em uma unidade atômica de trabalho. Mais detalhes sobre atomicidade da transação podem ser vistos na seção III.1.6.

Com relação aos outros componentes do SGBD, o gerenciador de transação recebe operações de transações e direciona-as para o "scheduler". Dependendo dos algorítmos de recuperação e de controle de concorrência que são usados, o gerenciador de transação pode executar outras funções. Por exemplo, num sistema de banco de dados

distribuído ele é responsável por determinar qual no pode processar cada operação submetida por uma transação.

II.1.5 - O Gerenciador de Dados

Continuando nosso estudo, concentraremos nossas atenções no Gerenciador de Dados pelo fato dele manipular memórias que podem ser corrompidas por falhas. O gerenciador de dados é dividido em dois componentes: o gerenciador de recuperação e gerenciador de "buffer" (figura II.1).

Recordando um pouco o que já vimos, sabemos que o gerenciador de "buffer" manipula memória fornecendo operações para trazer dados da memória estável para a memória volátil ("fetch"), e para enviar dados da memória volátil para a memória estável ("flush"). Vimos também que o gerenciador de recuperação processa as operações de leitura, gravação, comprometimento aborto e reinício, e controla parcialmente a operação "flush", para assegurar que a memória estável sempre tem o dado sue ele precisa para processar o reinício corretamente.

O "scheduler" recebe operações de leitura, gravação, comprometimento e aborto do gerenciador de transação. As operações de aborto, podem ser passadas imediatamente para o gerenciador de dados. Para as operações de leitura, gravação e comprometimento, o "scheduler" deve decidir quando aceitar ou rejeitar a operação. Se decidir rejeitar, ele envia um sinal de reconhecimento negativo para o gerenciador de transação, que envia de volta para ele uma operação de aborto, sue por sua vez é passada de pronto para o gerenciador de dados. Se decidir aceitar, ele a envia para o gerenciador de dados, que a processa manipulando a memória. Quando o gerenciador de dados termina o processamento da operação, ele avisa o "scheduler", que passa o sinal de reconhecimento para o

grrenciador de transação. Para a operação de leitura, o sinal de reconhecimento inclui o valor da leitura.

Além das operações de leitura, gravação, comprometimento e aborto, o grrenciador de dados também rode receber uma operação de reinício. Ela é enviada por um módulo externa, como o sistema operacional, para a recuperação de uma falha de sistema. A tarefa do reinício é trazer o sistema para um estado consistente, removendo os efeitos das transações não compromissadas e refazendo os efeitos das transações comrromissadas.

II.1.5.1 - Memória Estável

Quando o gerenciador de "buffer" envia uma operação de escrita para gravar um item de dados na memória estável, assume-se que a operação é completamente executada ou inteiramente desprezada, respondendo com um código de retorno sue indica qual das duas situações ocorreu. Estas operações de escrita são chamadas de atômicas (seção III.1.6).

Atualmente, discas são a forma mais popular de memória estável, neles, a granularidade do item de dado normalmente usada é a página. Quando uma página é gravada no disco, existem dois resultados possiveis: a execução pode ser correta ou resultar numa falha.

A granularidade dos itens de dados podem diferir daquela gravada para memória estável. O que requer uma atenção especial com a granularidade do item de dados quando do projeto de algorítmos de recuperação, uma ver que itens de dados individuais não podem ser gravados um por um (para registros pequenos) ou atomicamente (para registros grandes).

Para facilitar o raciocínio, assume-se que a granularidade dos itens de dados suportados pelo

serenciador de dados é idêntica a granularidade suportada pela memória estável. O que normalmente significa que um item de dado tem seu tamanho fixado em página.

Existem gerenciadores de dados que guardam exatamente uma cópia de cada item de dados em memória estável e gerenciadores de dados que guardam mais de uma. No primeiro grupo, cada vez que um item de dado é regravado seu valor antigo é destruido (atualização no lugar ou "in-place updating"). No segundo grupo, onde existe mais de uma cópia de cada item de dado, o gerenciador de "buffer" pode gravar um item de dado em memória estável sem destruir suas versões antigas. As versões antigas são chamadas de cópias sombras ou "shadow copies". A seção III.5.4 contem mais detalhes sobre esses dois tipos de atualização.

Assim, podemos definir o banco de dados estável como rendo o estado de um banco de dados em memória estável. Com "atualização no lugar", existe exatamente uma cópia de cada item de dado na memória estável. Com "cópias sombras", cada um dos diretórios define um estado do banco de dados. O esquema de paginação sombra é explicado na seção III.6.7.

CAPITULO III

RECONSTRUCÃO

Este capitulo descreve alguns dos elementos encontrados na maioria dos sub-sistemas de reconstrução, os tipos de falhas que afetam os sistemas de bancos de dados, as ações que um sub-sistema de reconstrução deve ser capaz de executar, e a parte do processamento normal que se relaciona com a atividade de reconstrução. O capitulo apresenta também algumas das técnicas usadas para reconstrução mais conhecidas e utilizadas por SGBDs, alguns comentários sobre como combinar estas técnicas para obter um sub-sistema de reconstrução satisfatório e uma comparação envolvendo custo e "overhead" entre as técnicas.

III.1 - Transação

No capitulo de introdução deste trabalha consideramos uma transação como sendo uma unidade lógica de trabalho, uma unidade de bloqueio r também uma unidade de recuperação, aparecendo sempre para o usuário como uma ação atômica. Entretanto, para nosso estudo é importantissimo que se faca um aprofundamento desse elemento de reconstrução. Assim, o conceito de transação, sua importância, suas propriedades, sua estrutura, os tipos de transações existentes, bem como seu conceito de atomicidade são apresentados a seguir.

III.1.1 - Definição de Transação

Na literatura podemos encontrar inúmeras definições para transação. Objetivando uma melhor compreensão do conceito

de transação, mostramos a seguir algumas definições em ordem cronológica de apresentação na literatura:

Uma transação é a unidade de bloqueio e recuperação na maioria dos bancos de dados comerciais disponíveis. Para o usuário aparece como uma ação atômica (ou todas ou nenhuma das modificações de uma transação são consideradas para o sistema)(VERHOFSTAD 1978).

Uma transação é uma sequência de comandos de consultas ou manipulação de dados que são especificados como uma unidade de interação contra o banco de dados (KIM 1979).

Ações consecutivas são grupadas em sequências chamadas transações que transformam o banco de dados de um estado consistente para um novo estado consistente. Para garantir a consistência o sistema somente aceita transações completas. (REUTER 19801.

Uma transação é uma transformacão de estado que tem as propriedades de atomicidade, durabilidade e consistência (a seção III.I.3 fala das propriedades da transação), e é um conceito chave para a estruturação de aplicações de gerenciamento de dados (GRAY 1981a).

Tipicamente uma transação é uma sequência curta de interações com o banco de dados, usando operações que representam uma atividade significativa no ambiente do usuário. Transação basicamente reflete a ideia que atividades de um particular usuário são isoladas de todas as atividades concorrentes, porém, restritas ao grau de isolamento e ao tamanho da transação. (HAERDER 1983).

Uma transação é uma unidade lógica de trabalho. Pode envolver varias operações que transformam um estada consistente do banco de dados em outro estada também consistente, sem necessariamente preservar esta consistência nos pontos intermediários. Também é uma unidade de recuperação e concorrência (DATE 1987).

Transação é uma unidade lógica de trabalho. Consiste na execução de uma sequência de operações especificadas pela aplicação. Começa com uma operação especial de inicio, e termina ou com uma operação de comprometimento (indicando que a unidade de trabalho foi completada com sucesso e tornando ar atualizações permanentes), ou com uma operação de cancelamento (indicando sue a unidade de trabalho não pôde ser copletada com sucesso por haver ocorrido alguma situação excepcional) (DATE 1988).

Transação é uma unidade de programa cuja execução preserva a consistência do banco de dados. Se antes sue se execute uma transação, o banco de dados esta num estado consistente, então quando se completa a execução da transação, ele estará num estado consistente (KORTH 1989).

III.1.2 - Objetivo e Necessidade da Transação

Segundo Ceri em (CERI 1984), problemas de recuperação e concorrência em bancos de dados estão muito ligados com a noção de transação. O objetivo da transação, e a eficiência, confiabilidade e a execução concorrente.

A maioria das transações podem ser refletidas num computador como transformações de estado do sistema. O estado do sistema inclue asserções sobre os valores de registros e sobre as transformações permitidas dos valores. Essas asserções sãs chamadas de restrições de consistência do sistema.

O sistema fornece ações que leêm e transformam os valores dos registros. Uma coleção de ações que compreendem uma transformação consistente dos estados pode ser grupada para formar uma transação. Transações preservam as restrições de consistência do sistema.

O banco de dados se encontra em um estado consistente se todos os seus itens satisfazem um conjunto de assertivas ou restrições de integridade. Uma transação e uma sequência de acessos sue levam o banco de dados de um estado consistente a outro estado consistente. Nesse sentido, uma transação é uma unidade de consistência.

Uma transação também deve preservar tanto a consistência interna do banco de dados quanto a sua consistência externa. Por esse motivo uma transação é considerada também como uma unidade de recuperação, já que pode ser vista como uma unidade atÔmica.

Finalmente, para que se possa manipular dados num ambiente com múltiplos usuários, deve existir um isolamento das execuções de cada um que previna interações indesejáveis ou descontroladas entre elas (HAERDER 1983). Essa idéia de isolar as atividades de um usuário particular de todas as outras que estão ocorrendo concorrentemente no sistema necessita também do conceita de transação.

III.P.3 - Propriedades

O conceito de transação requer que todas as ações sejam executadas de maneira indivísível (todas as ações são propriamente refletidas no banco de dados ou nenhuma). Para que esse tipo de indivisibilidade seja possível, uma transação deve ter as seguintes propriedades (GRAY 1981a), (HAERDER 1983), (CERI 1984):

> ATOMICIDADE : Se uma transação for interrompida por uma falha, seu resultado parcial é desfeito. As operações da transação serão todas executadas ou nenhuma (tudo ou nada).

DURABILIDADE: Uma vez que a transação foi completada e fez valer seus resultados no banco de dados, o sistema deve garantir que os resultados das operações nunca serão perdidos, independente de falhas posteriores.

CONSISTÊNCIA: Uma transação que encontra seu fim normal fazendo valer seus resultados, preserva a consistência do banco de dados. Em outras palavras, cada transação realizada com sucesso obedece a protocolos legais e faz valer somente resultados legais. Esta condição é necessária para a propriedade durabilidade.

ISOLAMENTO : Uma transação incompleta não pode revelar seus resultados para outras transações concorrentes antes de ser compromissada. As técnicas que permitem o isolamento são conhecidas como sincronização.

SERIALLZABILIDADE : Se várias transações são executadas concorrentemente, o resultado deve ser o mesmo que se elas fossem executadas serialmente (seção III.3.2).

III.1.4 - Estrutura de uma Transação

A estrutura típica de uma transação, ou seja, a que podemos supor que todas as transações se enquadram no mesmo padrão é a seguinte (DATE 1988):

. receber mensagem de entrada;

. efetuar processamento no banco de dados;

. enviar uma ou mais mensagens de saída.

Uma Única mensagem de entrada pode dar origem a várias mensagens de saída. A geração dessas mensagens pode acontecer paralelamente com o passo de "efetuar processamento", embora, as mensagens não sejam transmitidas até o final da transação.

Caso haja necessidade de uma maior interação do usuário com o sistema durante uma transação, pode-se agir de duas formas:

a) subdividir a interação em uma sequência de transações simples, cada uma delas tendo a estrutura mostrada acima;

b) tratá-la como uma grande transação que repete o ciclo entrada-processamento-saída muitas vezes.

Nenhuma das soluções é plenamente satisfatória. O problema da primeira abordagem é que outro usuário pode acessar e alterar dados entre as transações, o que poderá causar inconsistência no banco de dados. A segunda abordagem requer a capacidade de transmitir muitas mensagens sem aguardar o fim da transação. Se houver um problema na transação, mesmo depois de transmitir várias mensagens, o usuário deverá ser informado de que deve ignorar todas as mensagens desde o inicio da interação, devido a ocorrência da falha.

Segundo Haerder e Reuter em (HAERDER 19831, uma transação pode terminar de três maneiras:

a) pode encontrar seu ponto de compromisso fazendo valer seus resultados (fim normal);

b) pode detectar alguma entrada errada ou algum outro tipo de violação de consistência, evitando assim uma terminação normal e desfazendo tudo o que tinha feito (desfeita por ela própria).

c) pode executar dentro de um problema que somente pode ser detectado pelo sistema, como estouro de tempo ("time-out") ou impasse ("deadlock"), cujos efeitos são abortados pelo SGBD (desfeita pelo sistema).

III.1.5 - Tipos de Transação

Podemos classificar transações em quatro tipos (GRAY 1978): simples, conversacional, em grupo e distribuída.

Uma transação simples aceita uma única mensagem, faz alguma coisa, e produz uma Única mensagem. Transações simples tipicamente fazem poucas chamadas ao banco de dados. A maioria das transações são simples, e metade de todas as transações simples são somente de leitura, ou seja, não fazem mudanças no banco de dados.

Se uma transação envia e recebe varias mensagens sincronas é chamada de conversacional. Uma transação conversacional tem várias mensagens por processo e instâncias de transação. Nesse tipo, transações são suspensas por algum tempa (enquanto o operador pensa e responde) e portanto levantam problemas especiais de gerenciamento de recursos.

O termo transação em grupo ("batch") e usado para descrever uma transação grande e não usual. Em geral, tais transações não são "on-line", são iniciadas por um evento do sistema e executam por um longo tempo como um "job" em ultimo plano.

Se uma transação faz trabalho em vários nós de uma rede então ela precisa de uma estrutura de processo para representar seu trabalho em cada nó participante. Tal transação é chamada de distribuída.

III.1.6 - Atomicidade da Transação

A noção de transação é introduzida para forçar o sistema a executar uma sequência de ações elementares como se fosse uma unidade atômica, sem interferência externa (seção III.1.3).

É importante que o banco de dados esteja num estado consistente sempre que se iniciar uma transação. A fim de assegurarmos que isso ocorra, exigimos que as transações sejam atômicas, isto é, que todas as instruções associadas a ela sejam executadas completamente ou não.

Esta atomicidadr é fornecida pelo gerenciador de transação através de comandos que permitem esta visão. Como exemplo temos os seguintes comandos (DATE 1987):

> BEGIN_TRANSACTION - assinala o inicio de uma nova transação.
>
> END_TRANSACTION - assinala um final de transação com sucesso. Diz ao gerenciador de transações que a unidade lógica de trabalho foi completada com sucesso, estando novamente o banco de dados num estado consistente.
>
> ABORT - assinala um final de transação com fracasso. Diz ao gerenciador de transações que alguma coisa saiu errada e que todas as atualizações feitas pela unidade lógica de trabalho devem ser interrompidas e desfeitas.

Já sabemos que um sistema de banco de dados executa cada operação atomicamente. Isto quer dizer que o sistema de banco de dados se comporta como se executasse cada operação sequencialmente, ou seja, uma por vez. Porém, tipicamente, operações são rxecutadas concorrentemente. Entretanto, mesmo sendo executadas concorrentemente, o resultado final deve ser a mesmo de qualquer execução sequencial, isto é,

deve deixar o banco de dados consistente. Para que o banco de dados veja as transações como se fossem seriais (seção III.3.2), os recursos do banco de dados usados pela transação devem ser bloqueados.

Existem várias técnicas que podem ser utilizadas para controlar o acesso concorrente aos recursos e permitir o processamento paralelo de transações. Algumas destas técnicas são apresentadas na seção 111.3 que aborda controle de acesso concorrente).

III.i.7 - Executando Transações

Quando do inicio de uma transação, o gerenciador de transações deve identificar a transação de maneira univoca. Esta identificação será estendida a cada ação elementar executada a favor **desta** transação. Desta forma, uma eventual necessidade de desfazer parte das ações elementares associadas a uma particular transação não acarretaria problemas. Também é importante iniciar todo um contexto apropriado do qual dependerão os mecanismos de controle de concorrência e controle de integridade, que poderão ser acionados durante a execução da transação (CASANOVA 1985).

Durante a execução de transações sobre seu controle, o gerenciador de transações interage tanto com os mecanismos de controle de concorrência quanto com os mecanismos de controle de integridade. É preciso que o gerenciador de transações esteja atento ao fato de que recursos do sistema estarão sendo requisitados pelas várias transações em andamento. Ao conceder o uso de determinados recursos o gerenciador de transações aciana os procedimentos de controle de concorrência para evitar que mais de uma transação tenha acesso, simultaneamente, a recursos que não podem ser compartilhadas. Por outro lado, o próprio fato de cancelar transações pressupõe, tambem, que o serenciador de

transações mantém um histórico da sequência de ações que vão sendo executadas em favor da transação. De outra forma, pode ser impossível realizar a função de controle de integridade uma vez que não haveria meios de inverter efeitos de ações passadas. Manter este histórico é tarefa dos processas de controle de integridade com os quais o gerente de transações deve, portanto, manter estreito relacionamento durante a execução da transação.

Quando do termino da transação, uma decisão é tomada na sentido de tornar público todos os efeitos das ações elementares executadas em beneficio da transação (compromissar) ou nenhum deles (abortar), em cujo caso as ações elementares já invocadas devem ter seus efeitos removidos do banco de dados. No instante em que ocorre um aborto os mecanismos de controle de integridade que inverterão a transação entram em ação, restaurando o banco de dados a um estado consistente de onde possa recomeçar suas operações normais. Transações também podem ser canceladas mesmo estando em processamento normal. Isto pode ser visualizado num cenário onde transações são executadas iterativamente e o usuário tem a opção de simplesmente abandonar a transação antes de completá-la. Nesse caço, a transação está sendo cancelada a pedido do usuário, não devido a fatores anormais à operação do sistema. é claro que os meios de controle de integridade devem, também neste caso, ser acionados para garantir a consistência do banco de dados. 0 processo de confirmar uma transação também envolve os mecanismos de controle de integridade, pelo menos para registrar que este fato ocorreu, evitando que uma transação já confirmada tenha seus efeitos removidos quando de uma eventual falha do sistema. Finalmente, ao término da transação, o gerenciador de transações deve garantir que todos os recursos apropriados temporariamente por esta transação voltem ao controle do SGBD, sendo postos à disposição de outras transações.

111.2 - Falhas

No projeto de um sub-sistema de reconstrução, é importante ter uma clara noção de suais tipos de falhas devem ser consideradas, com que frequência ocorrem, qual a expectativa de tempo rara recuperação, etc. Parem, e extremamente difícil para alguém fazer todoç os tipos de prognósticos quanto ao comportamento do "hardware" r "software" envolvidos no processamento, além de sabermos que algumas Calhas são extremamente raras de acontecer. Assim, iremos considerar os seguintes tipos de falhas (HAERDER 1983): falhas de transação, falhas de sistema e falhas de meio de armazenamento.

111.2.P - Falha de Transação.

Segundo Casanova em (CASANOVA 1985), existem situações que exigem ações por parte do sistema de controle de integridade do banco de dados embora não se configurem como falhas em componentes do sistema. O caso típico é quando, sob operação normal, surge a necessidade de cancelar transações. Isto pode ocorrer tanto por erro ou a pedido do usuário, como podem ser ações forçadas pelo SGBD como Última instância para evitar bloqueios na execução de transações que competem por certos recursos do sistema. Estes casos, rotulados como pseudo-falhas do sistema, são recuperáveis localmente e afetam çomente uma ou poucas transações. O banco da dados continua processando normalmente.

Sobre falhas de transações Haerder e Reuter em (HAERDER 1983) concluem que:

a) Dentro de uma aplicação, a média de transações abortadas por elas próprias e muito constante, dependendo somente da quantidade de entradas inválidas de dados, da

qualidade dos testes de consistência executados pela transação, etc.

b) A média de transações abortadas pelo SGBD, especialmente aquelas causadas por "deadlocks", depende em grande parte do grau de paralelismo, da granularidade do bloqueio usada pelo SGBD, do esquema lógico (podem existir pontos de concentração de dados - "hot spots") e do grau de interferência entre atividades concorrentes.

Isto permite dizer que falhas de transação ocorrem varias vezes por minuto e a recuperação dessas falhas deve ocorrer dentro da tempo requerido pela transação para sua execução normal.

III.2.2 - Falha de Sistema

O termo "falha de sistema" é utilizado para designar qualquer evento que obrigue o sistema a parar, exigindo um subsequente reinício. As falhas de sistema aqui consideradas podem ser causadas por um erro no código do SGBD, por uma falha no sistema operacional, ou por uma falha de hardware. Estes defeitos são cognominados de falhas primárias.

Em todos esses casos o processamento termina de maneira descontrolada, e se assume a perda do conteúdo da memória volátil. Uma vez que a memória não volátil permanece intacta (o banco de dados não é danificado), é requisitada uma recuperação na mesma quantidade de tempo sue seria necessário para executar todas as transações ininterruptamente. Uma falha de sistema pode ocorrer varias vezes por mês, dependendo da estabilidade do SGBD e do seu ambiente operacional.

III.2.3 - Falha de Meio de Armazenamento

Esse tipo de falha pode causar a perda de parte ou de todo o armazenamento secundário que contém o banco de dados. Existem várias causas para esse problema, as mais comuns são:

- erros no sistema operacional nas rotinas de gravação em disco;
- erros de hardware na controladora de disco ou canal;
- falha na cabeça de leitura/gravação;
- partículas de poeira sobre a superfície dos discos;
- perda de informação por deterioração magnética;
- etc.

Isto se verificando, diz-se sue o sistema sofreu uma falha secundária.

Dispositivos de armazenamento magnético geralmente são muita seguros, tornando a necessidade de recuperação desse tipo de falha difícil de acontecer, talvez poucas vezes por ano. Dependendo do tamanho do banco de dados, do meio usado para armazenamento da cópia e da idade da cópia, a recuperação pode levar algumas horas. Nesse caso, o controle de integridade do SGBD apela para a memória secundária dormente ("off-line").

Entende-se como memória secundária dormente toda memória fisicamente desconectada do sistema. Geralmente, devido a sua grande capacidade de armazenamento de dados, fitas magnéticas são empregadas para este fim. Eventos que destruam ou inutilizem a conteúdo deste tipo de memória são chamados de falhas terciárias.

III.2.4 - Resumo de Falhas

Neste estudo sobre falhas (seção III.2), é importante observarmos a noção de frequência com que os vários tipos de falhas costumam ocorrer na pratica, e também o tempo necessário para que o controle de integridade restaure a operação normal do banco de dados em cada caso. A figura III.1 resume a seção (CASANOVA 1985).

TIPO DE FALHA	FREQUENCIA	TEMPO DE RECUPERAÇÃO
Pseudo–falha	várias por minuto	milisegundos/segundos
Primária	várias por mês	segundos/minutos
Secundária	várias por ano	minutos/horas
Terciária	várias por século	dias

III.1 - Características das falhas.

111.3 - Controle de Concorrência

Segundo Brrnstein em (BERNSTEIN 1987), controle de concorrência é a atividade de coordenar ações de processos que operam em paralelo, aressam dados partilhados, e assim, potencialmente interferem entre si.

Controle de concorrência visa a garantir que, em toda execução simultânea de um grupo de transações, cada uma seja executada como se fosse a única do sistema (CASANOVA 1985). Mais precisamente, quando duas ou mais transações executam concorrentemente, suas operações no banco de dados são executadas de maneira intercalada, isto é, operações de uma transação podem ser executadas entre duas operações de outra transação. Esta intercalação pode fazer com que transações se comportem incorretamente, levando o banco de dados a um estado inconsistente (BERNSTEIN 1987).

A parte do sistema de banco de dados sue determina a intercalação das operações do banco de dados, preservando a

consistência do banco de dados é chamada de mecanismos de controle de concorrência.

Técnicas de controle de concorrência devem garantir que toda execução concorrente de um conjunto de transações seja serializável, ou seja, equivalente a alguma execução das transações em que cada transação é completamente processada antes da próxima começar.

Existem três classes básicas de tecnicas de controle de concorrência: técnicas de bloqueio, pré-ordenação ou mistas (CASANOVA 1985).

As técnicas de bloqueia (seção III.3.3) exigem que antes de cada operação a transação bloqueie o objeto em questão. Este bloqueio deve ser mantido até a transação ser completada ou cancelada. Um efeito indesejável do uso de bloqueios é a possibilidade da ocorrência de bloqueias mútuos ou "deadlocks" (seção III.3.3.4).

A técnicas de pré-ordenação consistem em estabelecer, a priori, uma sequência serial para a execução das transações. Esta sequência deve ser respeitada por todas as transações. Desta forma, as transações são executadas concorrentemente como se fossem processadas serialmente na ardem escolhida. Em geral estas técnicas evitam problemas de bloqueio mútuo, mas criam problemas de "reinício cíclico de transações" e "postergação indefinida de transações" (BERNSTEIN 1987).

As técnicas mistas, como o próprio nome indica, tentam combinar as vantagens do bloqueio e pré-ordenação.

Dessas técnicas, o bloqueio é com certeza a mais utilizada na prática (pelo menos nos sistemas centralizados). Um quadro comparativo entre essas técnicas pode ser encontrado em (CASANOVA 1985).

III.3.1 - Anomalias

Mesmo no caso de todas as transações estarem individualmente carretas, é possível que transações executadas concorrentemente num sistema compartilhado (multiusuário) interfiram entre si, produzindo anomalias e como consequência um resultado final incorreto.

As anomalias mais comumente encontradas são perda de atualizações, dependência não compromissada e análise inconsistente (DATE 1986).

a) Perda de Atualizações

No problema de perda de atualização ("lost update"), uma transação interfere sobre a outra, alterando um mesmo item de dado numa operação que ocorre no intervalo de execução das operações da outra transacão sobre o mesmo item de dado, e que portanto acaba sendo ignorada.

Na figura III.2 temos um exemplo de perda de atualização. A transação A lê um registro, em seguida, a transação B lê o mesmo registro. A transação A atualiza o registro com base nu que ela leu, em seguida, a transação B atualiza o mesmo registro. Assim, a atualização feita por A (anterior a feita par B) não foi considerada, ficando perdida na transação B.

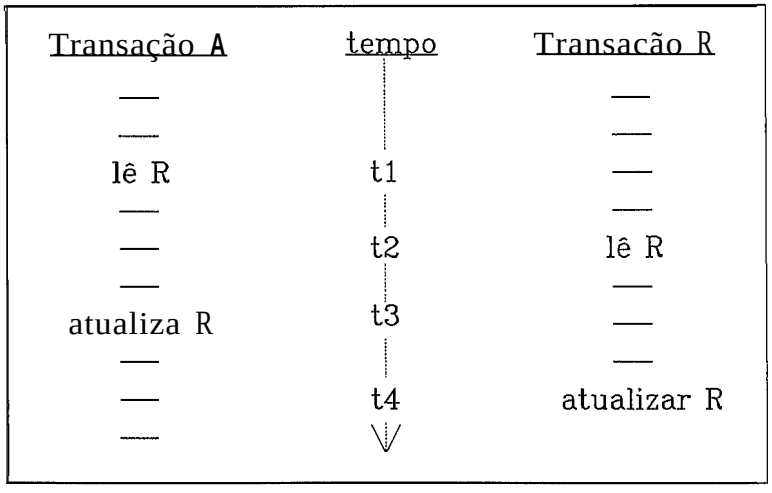


Figura III.2 - Perda de atualização (DATE 1986).

b) Dependência não Compromissada

Este problema aparece quando se permite uma transação recuperar (ou, pior, atualizar) um registro que foi atualizado por outra transação ainda não compromissada. Assim, existe a possibilidade de ela não ser compromissada e sim desfeita. Neste caso, a primeira transação terá visto (ou atualizado) algum dado que não existe mais (ou nunca existiu).

Neste exemplo (figura III.3), a transação A vê uma atualização não compromissada. Esta atualização e desfeita depois que a transação A já fez a leitura, assim, a transação A está operando sobre um falso valor, pois com o cancelamento de B seu valor volta a ser o que era antes da atualização feita por B.

Se e transação A fizesse uma atualização em ver de uma leitura, o problema seria bem pior.

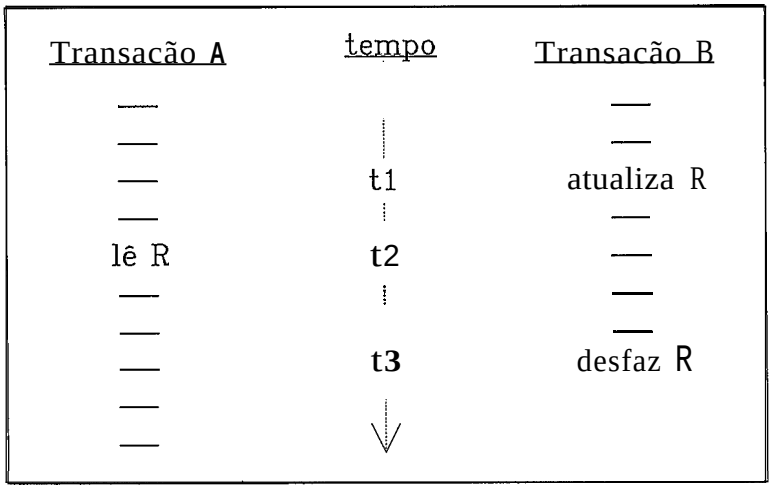


Figura III.3 - Dependência não compromissada (DATE 1986).

c) Analise Inconsistente

Neste tiro de problema, informações inconsistentes são recuperadas. Isto acontece quando uma transação acessa um estado inconsistente do banco de dados e assim procede uma análise inconsistente. A figura III.4 ilustra um exemplo onde uma transação A soma contas, enquanto uma transacão B transfere uma quantidade 10 da conta 3 para a conta 1. O resultado produzido por A (110) *E* incorreto.

Dizemos que A viu um estado inconsistente do banco de dados e por isso fez uma análise inconsistente. A diferenca entre esse caso e o anterior é que aqui não existe a questão de A ser dependente de uma alteracão não compromissada, uma vez que B comprometeu todas as suas atualizações antes de A ver a conta 3.

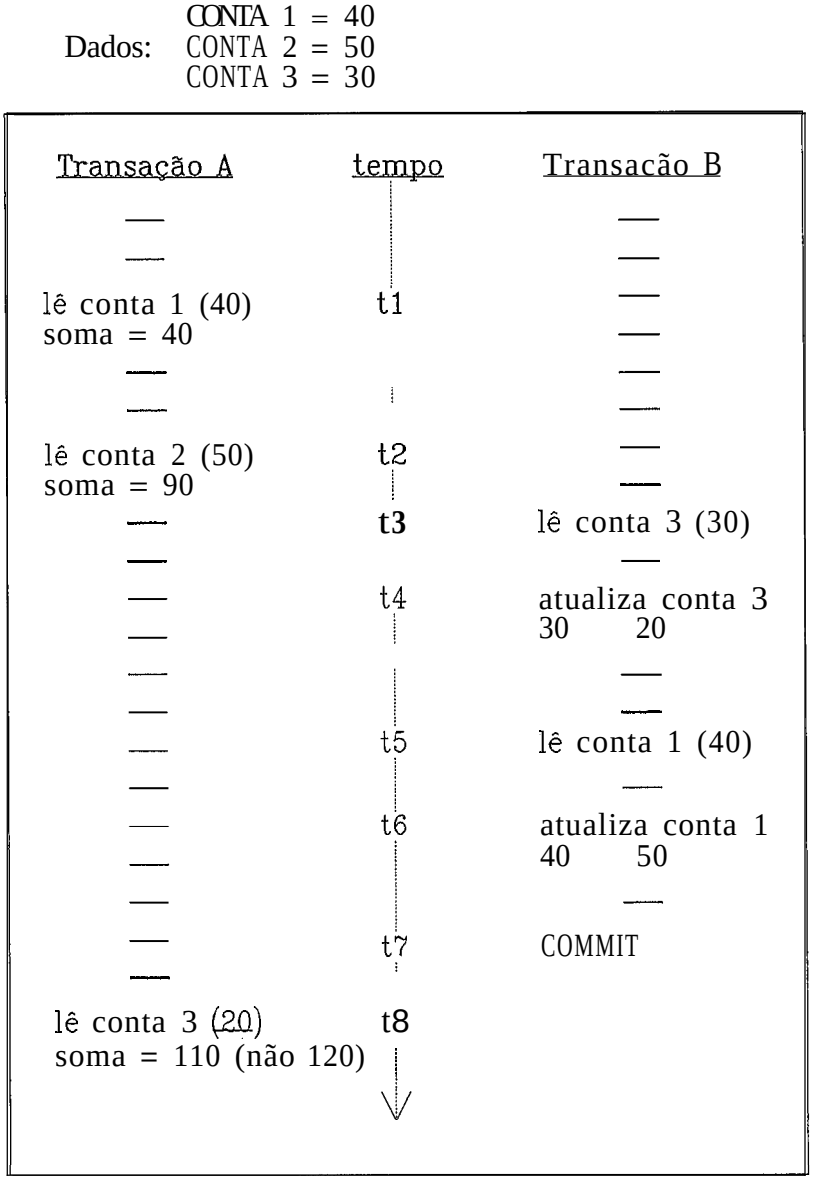


Figura III.4 - Análise Inconsistente (DATE 1986).

III.3.2 - Serialização

Outra definição importante para concorrência (já levantada previamente mas que deve ser enfatizada) é a execução serial das transações, ou seja, uma determinada execução entremeada de certo conjunto de transações é tida como serializável se, e apenas se, produzir o mesmo resultado que alguma execução serial dessas mesmas

transações, isto é, dado um estado inicial arbitrário do banco de dados como entrada, a execução entremeada produz a mesma saída que alguma execução serial operando no mesmo estado do banco de dados inicial.

Segundo Bernstein em (BERNSTEIN 1987), serializabilidade é a definição de exatidão para controle da concorrência em sistemas de banco de dados. Ou seja, uma determinada sequência de execução entremeada será considerada correta se, e apenas se, ela for serializável (DATE 1988).

III.3.3 - Bloqueio

Bloqueio é um mecanismo comumente usado para resolver problemas de sincronização de acesso para dados partilhados tais como registros, blocos e páginas em um banco de dados compartilhado. O bloqueio pode ser imaginado como um bloco de controle que inclui, entre outras coisas, a identificação do registro com o qual esta associado (isto é, o registro que esta bloqueado), e a identificação da transação que fez o bloqueio. Se a transação T bloquear o registro R, então são feitas certas garantias a T, com relação a R (por exemplo, T certamente terá garantias de que nenhuma outra transação concorrente será capaz de atualizar R, até que T libere seu bloqueio).

A natureza exata das garantias depende do tipo de bloqueio (exclusivo/compartilhado), da granularidade do bloqueio (unidade de dados bloqueada: registro, página, bloco, arquivo ou até todo o banco de dados) e do seu nível da isolamento (grau de interferência que uma transação pode tolerar com relação a uma base).

III.3.3.1 - Tipos de Bloqueio

a) Bloqueio exclusivo

Podemos definir bloqueio exclusivo ("exclusive lock") da seguinte maneira:

Se uma transação **T** contiver um bloqueio exclusivo em algum objeto (como um registro do banco de dados), então nenhuma transação distinta de **T** pode fazer um bloqueio daquele objeto, até que **T** libere o seu bloqueio. Este tipo de bloqueio fornece base para a resolução do problema da atualização perdida (seção III.3.1).

b) Bloqueio Compartilhado

Uma transação pode precisar manter dados bloqueados, mesmo que não os esteja atualizando. Assim, e introduzido um segundo tipo de bloqueio, o bloqueio compartilhado ("shared lock") com a seguinte definição:

Se uma transação **T** mantiver um bloqueio compartilhado em algum objeto (como um registro do banco de dados), então uma transação distinta de **T também** pode fazer um bloqueio compartilhado sobre aquele objeto; mas nenhuma outra transação distinta de **T** pode fazer um bloqueio exclusivo sobre aquele objeto, até que todos os bloqueios compartilhados dele tenham sido liberados.

III.3.3.2 - Granularidade do Bloqueio

Por granularidade de um bloqueio deve-se entender o tamanho da porção da base de dados escolhida para ser operada pelas primitivas do protocolo de bloqueio.

Bloqueios podem ser aplicados a unidades de dados maiores ou menores do que um registro. Pode ser aplicado para todo um banco de dados, para um espaço lógico, para um

arquivo, ou, num extremo oposto, para um campo específico de um registro.

Em termos da granularidade empregada num banco de dados, existem prós e contras. Quanto mais fina a granularidade, tanto maior a concorrência; quanto mais grossa, menos bloqueios serão feitos e testados, e menor será a sobrecarga no sistema.

III.3.3.3 - Níveis de Isolamento.

O isolamento define o grau de interferência que uma transação pode tolerar com relação as entidades que ela acessa (DATE 1988). Quanto maior o nivel de isolamento, menor a interferência e portanto mais baixa a concorrência.

A conclusão é que no maior nível de isolamento (nível 5) o problema de "fantasmas" (onde uma transação concorrente cria um novo registro ou atualira um registro existente ainda não lido de modo que eles passem a atender a uma seleção feita anteriormente) é contornado. O que significa, informalmente, que a transação não está ciente da existência de qualquer transação concorrente sobre a mesma entidade. A intenção do nível 5 e que a transação não deva ver nenhuma mudanca, exceto, naturalmente, as mudancas feitas pela própria transação (DATE **1988).**

Uma forma de implementação da nível 5 de isolamento é bloquear o método de acesso usado pelo SGBD para selecionar os objetos do bloqueio.

III.3.3.4 - Impasses ("deadlocks")

Outro problema existente em sistemas compartilhados é o bloqueio mútuo, ou impasse, ou "deadlock", decorrente do

protocolo de bloqueio exclusivo. O "deadlock" é uma situação em que duas ou mais transações estão em um estado simultâneo de espera, cada qual aguardando que uma das demais libere um bloqueio antes que ela possa prosseguir.

Existem três abordagens para resolver "deadlocks". Prevenção de "deadlocks", detecção de "deadlocks" e impedimento de "deadlocks".

Prevenção de "deadlocks" é um esquema onde a forma mais simples de evitar impasses consiste em liberar um objeto sempre que a transação pedir novo bloqueio. Este método é problemático pois permite a criação de execuções não serializáveis.

Detecção de "deadlocks", onde estes são detectados pela construção explícita de um grafo de espera ("wait-for graph"). Os nodos do grafo representam transações em execução e os arcos as esperas. Toda vez que uma solicitação de bloqueio causar um "wait" (espera) pode-se verificar a ocorrência de um "deadlock". Essa verificação pode ser menos frequente, o que pode significar que alguns "deadlocks" sejam detectados muito tarde.

Impedimento do "deadlock" é uma técnica conservativa onde uma transação adquire todos os bloqueios ou nenhum ao mesmo tempo.

Em geral, a maioria dos sistemas permitem que os "deadlocks" ocorram, ao invés de tentar evita-los. Isto porquê para a maioria dos sistemas centralizados evitar "deadlocks" é quase sempre mais oneroso do que resolvê-los.

Tomando-se este enfoque, o sistema deve estar preparado para a possibilidade de um "deadlock". Isto é, precisa estar apto para detectar a ocorrência de "deadlocks" e resolve-los, ou quebrá-los, quando de fato ocorrerem.

Segundo Bernstein em (BERNSTEIN 1987), o controlador de concorrência necessita de uma estratégia para detectar "deadlocks", assim nenhuma transação será bloqueada para sempre. Uma estratégia é através do controle de tempo ("timeout"). Se o controlador de concorrência acha que uma transação está esperando tempo demais por um bloqueio, então ele simplesmente supõe que ocorreu um impasse envolvendo a transação e aborta-a. Uma vem que o controlador de concorrência somente supõe que a tansação está envolvida num impasse, ele pode estar cometendo um erro.

Outra abordagem que pude ser uçada pelo controlador de concorrência é detectar precisamente "deadlocks". Para fazer isto, o contrulador de concorrência precisa manter um grafo de espera. Quando todas as transações estão bloqueadas esperando ror bloqueios que nunca serão liberados, então estão num "deadlock". Explorando esta observação, o controlador de concorrência pode detectar "deadlocks" testando os ciclos do grafo.

A quebra de um "deadlock" consiste na escolha de uma vitima (uma das transações sob impasse) que é forçada a desfazer sua ação. A vitima não é necessariamente a transação que causou o "deadlock", pode ser a que foi iniciada mais recentemente, a que tiver feito o menor número de bloqueios, a que tiver feito o menor número de atualizações, ou ainda a que consumiu menos recurso ate então. O processo de "desfazer" envolve não só terminar a transação e desfazer todas as suas ações, mas também liberar todos os seus bloqueios, de modo que os recursos em questão possam agora ser alocados para outras transações.

III.3.4 - Ordenação de Selos Temporais ("timestamps")

Nesta abordagem, uma ordem de serialiração é selecionada a priori e a execução da transação é forçada a obedecer

esta ordem (AGRAWAL 1985). Na versão básica desse esquema, cada transação é assinalada com um único selo temporal (número sequencial não repetido). Um pedido de leitura de um objeto por uma transação é aceito somente se nenhuma outra transação com um selo temporal maior (transação mais nova) escreveu aquele objeto. Similarmente, um pedido de gravação é honrado somente se nenhuma outra transação com um selo temporal maior leu ou gravou o objeto em questão. Existem duas variações deste algorítmo básico, ordenação de selo temporal multiversão e conservativo, descritos em (BERNSTEIN 1987). Ambos se preocupam em reduzir o número de reinícios introduzidos pelo algorítmo básico.

Como principal vantagem desta técnica temos a não ocorrência de "deadlocks" como consequência da inexistência de bloqueios. A desvantagem surge quando o grau de concorrência torna-se alto, pois muitas vezes transações terão que ser refeitas, comprometendo a performance do sistema.

Na literatura especializada esta abordagem é tida como a mais apropriada para sistemas distribuídos (SGBDDs). Assim, é possível que no futuro esta técnica seja a mais utilizada, devido a tendência de descentralização dos sistemas computadorizados.

111.4 - Ações de Reconstrução

Um banco de dados é consistente a nível de transações se e somente se ele contém os resultados de transações com sucesso. Objetivando alcançar esta consistência é necessária a existência de quatro ações de reconstrução para conviver com diferentes situações (HAERDER 1983):

a) Desfazer uma transação ("transaction undo"). Ocorre quando uma transação é abortada por ela mesma ou pelo sistema durante a execução normal. Todas as alterações

devem ser removidas do banco de dados e nenhuma outra transação é afetada.

b) Desfazer todas as transações inacabadas ("global undo"). Quando há uma falha de sistema, os efeitos de todas as transações incompletas devem ser desfeitos.

c) Refazer parcialmente ("partial redo"). Quando há uma falha de sistema, os resultados de transações completas podem não ter sido refletidos ainda no banco de dados, pois suas alterações poderiam estar nos "buffers" na ocasião da falha. Esta alterações devem ser refeitas no banco de dados pelo gerenciador de recuperação.

d) Refazer globalmente ("global redo"). é assumido que o banca de dados foi destruido fisicamente, devendo ser restaurado através de uma cópia que reflita o estado do banco de dados a algum tempo atrás Juntamente com os efektos de todas as transações compromissadas desde a criação da cópia.

Com estas definições introduzimos o conceito de transação como unidade de reconstrução num sistema de banco de dados.

III.5 - A Hierarquia de Mapeamento de um SGBD

Para uma melhor compreensão da atuação de um sistema de reconstrução é necessário conhecer alguns procedimentos do SGBD r do sistema operacional envolvendo a alteração dos dados na memória e sua consolidação no banco de dados. Para isso, a hierarquia de armazenamento e as operações de atualização juntamente com seus conceitos são apresentadas (HAERDER 1983).

III.5.1 - O Processo de Mapeamento: Objetos e Operações

NUM nível mais baixo, o banco de dados é constituído de alguns bilhões de bits armazenados em disco, os quais são interpretados e transformados pelo SGBD em informações significativas sobre as quais os usuários podem operar. Os objetos tornam-se mais complexos a cada nível, permitindo operações mais poderosas e sendo restritos por mais regras de integridade.

A hierarquia de mapeamento é composta pelos seguintes níveis:

a) Gerência de Arquivos (nível 1)

é a camada mais baixa e opera diretamente sobre os bits armazenados em memória não volátil. Esta camada deve reconhecer as características físicas de cada meio de armazenamento existente no sistema e abstrair o canceito de blocos de tamanho fixo que podem ser lidos, escritos e identificados.

b) Controle de Propagação de Páginas (nível 2).

Esta camada trabalha com o conceito de pagina, que é uma partição de tamanha fixo de um espaço de endereçamento mapeado em blocos físicos por essa camada. Uma página pode ser armazenada em diferentes blocos físicos durante sua existência, dependendo da sua estratégia de implementação.

c) Gerência de Caminhos de Acesso (nível 3).

Esta camada implementa funções de mapeamento mais complicadas que as implementadas pelas camadas mais baixas. Ela mantém todos os objetos armazenados na banco de dados (registros, campos, etc.) e seus caminhos de acesso relacionados (ponteiros, tabelas "hash", arvores de pesquisa, etc.) em um espaço de endereçamento virtual potencialmente ilimitado. Este espaço é dividido em páginas de tamanho fixo.

d) Camada de Acesso Navegacional (nível 4).

Nesta ramada encontram-se os objetos e as operações típicas de uma linguagem de manipulação de dados (LMD). Comandos como "store", "modify" e "find next" manipulam os objetos vistos neste nível, que são ocorrências de registro, elementos de conjuntos, etc. Nesta camada as ocorrências de registros estão na estrutura imposta pelo SGBD, par exemplo hierarquia ou rede.

e) Camada de Acesso não Procedural (nível 5)

Este nível fornece uma interface não procedural para o banco de dados. O modela de dados e despido de todos os aspectos relacionados com caminhos de acesso. As operações só dizem o que deve ser feito e com que dados, não precisando dizer coma encontrá-los. O usuário pode trabalhar com conjuntos de registros em vez de um por vez. O modelo relacional com sua linguagem de consulta de alto nível tipo SQL e SEQUEL e um exemplo da abstração encontrada neste nível.

O modelo apresentado na figura III.5 mostra a descrição dos diversos níveis da hierarquia de mapeamento de um banco da dados desde a nível de armazenamento físico (o mais baixo) até o nivel de interface com o usuário (o mais alto).

Nível de Abstração	Objetos	Dados Auxiliares de Mapeamento
acesso não procedural ou algébrico	relações, tuplas	descrição lógica do esquema
acesso navegacional orientado a registro	registros, conj., hierarq., redes	descrição lógica e física
ger. de caminhos de acesso e registro	regs. físicos, caminhos acesso	tab. espaços livres tab. ponteiros
controle de propagação	segmentos, páginas	tabelas de páginas filtros
gerência de arquivos	arquivos, blocos	diretórios, VTOCs

Figura III.3 - Descrição da hierarquia de mapeamento.

III.5.2 - Visões de um Banco de Dados

O processo de atualização de um banco de dados leva-nos a existência de diferentes visões deste mesmo banco de dados (HAERDER 1983). O fato das alterações dos dados se processarem em primeiro lugar em memória volátil ("buffers") r só depois de um determinado período de tempo ou sequência de ações serem gravadas em memória não volátil, justifica a existência de diferentes visões do banco de dados, que devem ter suas características consideradas ao se projetar um sistema de reconstrução.

Um banco de dados corrente compreende todos os objetos disponíveis para o SGBD durante o processamento normal. Isto inclui todos os dadas em qualquer nível da hierarquia de armazenamento em discos ou "buffers" da memória.

Banco de dados materializado e o estado do banco de dados que está em memória secundária depois de uma falha. Não existe "buffer" e as páginas alteradas em memória volátil são perdidas, permanecendo seus antigos valores.

Este banco de dados é o ponto de partida para uma carga ou um reinício após uma falha.

Banca de dados físico é composto por todos os blocos da cópia "on-line" contendo imagens de páginas tanto atuais como obsoletas. Dependendo da estrategia de propagação usada no nível 2, podem existir diferentes valores para uma página no banco de dados físico, nenhuma delas é necessariamente o conteúdo corrente. Esta visão não é normalmente usada por procedimentos de recuperação, podendo ser explorada por um programa de salvamento para recuperar informações.

III.5.3 - Operações de Atualização

Com estas três visões de um banco de dados, pode-se distinguir três tipos de operações de atualização, afetando de maneira diferente cada uma das visões (HAERDER 1983):

a) Modificação do conteúdo da pagina

Esta operação se realiza no "buffer" do banco de dados (memória volátil) afetando somente o banco de dados corrente.

b) Operação de gravação.

Transfere uma página modificada para um bloco em disco, em geral afeta somente o banco de dados físico. Se a informação sobre a bloco contendo o valor da nova página é armazenada em memória volátil, os novos valores não poderão ser acessados após uma falha, pois ainda não fazem parte do banco de dados materializado.

c) Propagação.

é a operação que torna a imagem de uma página gravada previamente, parte do banco de dados materializado. Esta operação grava a estrutura de controle atualizada do mapeamento de páginas para blocos num lugar seguro e não volátil, tornando-o disponível após uma falha.

Se páginas são sempre escritas no mesmo bloco (alocação direta de páginas "update-in-place"), implicitamente gravação é equivalente a propagação. Entretanto, existe uma diferença importante entre estas operações se uma pagina pode ser armazenada em diferentes blocos.

III.5.4 - Mapeamento de Conceitos para Atualizações

Nesta seção são definidos conceitos relacionados com a operação de mapeamento de alterações num banco de dados da memória volátil para a não volatil. Eles estão diretamente relacionados com o conceito de visões de um banco de dados visto anteriormente (seção III.5.2).

Aqui, a questão chave e que cada modificação de uma página se passa no "buffer" do banco de dados sendo alocada em memória volátil, devendo ser levada para memória não volátil (banco de dados físico) para salvar seu estado. Dois diferentes esquemas podem ser usados para isso (HAERDER 1983): alocação direta de página e alocação indireta de página.

Usando alocação direta de página "update-in-place" (figura III.6), cada propagação (gravação física) está sujeita a interrupções por falhas do sistema, deixando o banco de dados materializado e possivelmente o físico num estado inconsistente.

Usando o esquema de alocação indireta de pagina, cada saída é direcionada para um novo bloco, mantendo o conteúdo antigo da pagina sem alteração (figura III.7). Isto da a

possibilidade de se guardar várias versões de uma página e determinar qual o momento em que a página modificada começa a fazer parte do banco de dados materializada. Neste tipo de alocação, existe sempre uma maneira de voltar ao estado antigo.

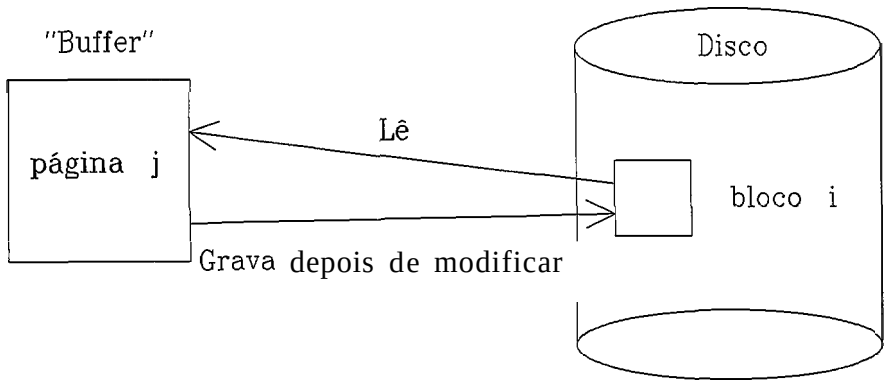


Figura III.6 - Alocação direta de páginas.

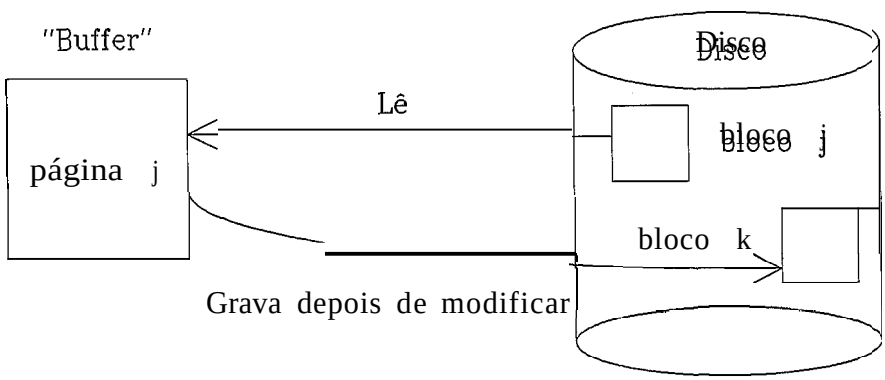


Figura III.7- Alocação indireta de páginas.

Tomando como base esses dois conceitos (alocação direta e indireta de paginas) podemos analisar os métodos de gravação dos "buffers" de memória em um meio de armazenamento, envolvendo as seguintes estratégias (HAERDER 1983):

- estratégias de propagação de alterações;

- estratégias de substituição de páginas;

- processamento de fim de transação (EOT).

Pode-se distinguir dois tipos de estratégias para propagação:

a) ATÔMICA - Um grupo de paginas modificadas pode ser propagada como uma unidade, tornando-se parte do banco de dados materializado. Nesta estrategia todas as paginas são substituidas ou nenhuma.

b) NÃO ATÔMICA - As paginas são escritas em blocos de acordo com a política de alocação direta de página ("update-in-place"). Como a propagação indivísivel de um conjunto de páginas não é permitida (escrita e propagação coincidem), uma página escrita automaticamente torna-se parte do banco de dados materialirado.

Estas duas estratégias estão relacionadas com o mecanismo de mapeamento da estrutura de armazenamento e descrevem as transições do banco de dados materializado de um estado para outro causadas pelas alterações de paginas. Como apenas o conteúdo do banco de dados materializado está disponível após a ocorrência de uma falha, a escolha da estratégia de propagação pode determinar o custo do reinício do sistema.

Quanto a substituição de páginas, num sistema com tamanho de "buffer" limitado, paginas modificadas podem ser gravadas em disco de acôrdo com algum algoritmo de substituição (LRU, FIFO, CLOCK, etc.) antes que as transações que elas refletem terminem. Com isso, surgem dois diferentes métodos de manusear paginas modificadas:

a) ROUBAR - Páginas modificadas pertencentes a uma transação não encerrada podem ser escritas e/ou propagadas a qualquer tempo. O banco de dados físico (materializado, se for propagação "não atômica") é modificado em tempos arbitrários, mesmo por transações incompletas.

b) NÃO ROUBAR - Páginas modificadas numa transação são mantidas no "buffer" pelo menos até o final da transação (EOT).

Essas duas propriedades (ROUBAR e NÃO ROUBAR) estão relacionada5 com a transação e indicam como a reconstrução pode ser feita r influenciam o custo de desfazer transações.

Nos esquemas de propagação NÃO ATÔMICA a definição de ROUBAR pode ser baseada tanto para gravação como para propagação por não serem descriminadas neste tipo de esquema. No caso de propagação ATÔMICA ambas as variantes de ROUBAR são concebidas, e cada uma tem um impacto diferente sobre ações de recuperação "undo".

O Ultimo critério referente a manuseio de "buffer", o processamento de fim de transação (EOT), está relacionado com a questão de como as páginas modificadas são tratadas na final da transação. Para que transações sejam duráveis, deve ser garantido que estas paginas modificadas passarão para o banco de dados materializado. Existem duas maneiras de fazer isso:

a) FORÇAR - Todas as páginas modificadas são escritas e propagadas durante o processamento de fim da transação. (Se ainda não substituídas como no caso de ROUBAR.)

b) NÃO FORÇAR - Nenhuma propagação é disparada durante o processamento de fim de transação.

(Arriscando ter que fazer "redo" parcial num reinício.)

A primeira alternativa dispensa informações do conteúdo do "buffer" para reconstrução, porém, necessita informações redundantes para uma reconstrução global. A segunda opção precisa de informações sobre as alterações que ainda estão nos "buffers" para uma reconstrução parcial.

111.6 - Técnicas de Reconstrução

Diferentes mecanismos e técnicas de recuperação de erros utilizados por sistemas gerenciadores de arquivos são apresentadas em (VERHOFSTAD 1978), (LANDES 1980), (CASANOVA 1985) e (KORTH 1989). O propósito destes mecanismos e técnicas de recuperação é restaurar os dados do sistema para um estado usavel após a ocorrência de uma falha. Assim, componentes e algoritmos podem ser adicionados a um sistema com o prbposito de remover erros e restaurar o banco de dados para um estada aceitável pelos usuários desse sistema. A interpretação de aceitável muda de acôrdo com o ambiente. Em geral, aceitável significa correto, válido ou consistente.

Um banco de dados está num estado correto se suas informações consistem na mais recente copia dos dados introduzidos pelos usuários no banco de dados, e não contém os dados removidos pelos usuários. Está num estado válido se suas informações fazem parte das informações de algum estado correto (isto significa que não existem dados espúrios, embora alguma informação possa ter sido perdida). Finalmente, esta num estado consistente se estiver num estada válido e suas informações satisfazem o conjunto de restrições de integridade dos usuários (VERHOFSTAD 1978). Para ilustrar essas definições Verhofstad apresenta o seguinte exemplo: um usuário mantem um arquivo fonte e um arquivo objeto. O banco de dados estará num estado correto se os arquivos fonte e objeto mais recentes estiverem

disponíveis. O banco de dados estará num estado válido, se os arquivos fonte e objeto (não necessariamente os mais recentes) estiverem disponíveis. O banco de dados estará num estado consistente somente se os arquivos fonte e objeto "correspondentes" estiverem disponíveis. Considera-se assim, que um estado correto é também um estado consistente.

Como visto anteriormente na introdução, alguns componentes e algoritmos são usados para restaurar o banco de dados para um estado previamente consistente, outros para tolerar a ocorrência de falhas e, outros para retornar o banco da dados ao Último estado consistente e anterior a falha.

Diferentes tipos de recuperação são possíveis em sistemas de banco de dados. Verhofstad em (VERHOFSTAD 1978) considera os seguintes tipos de recuperação:

- recuperação para um estado correto

- recuperação para um estado correto sue existia em algum momento no passado.

- recuperação para um possivel estado prévio (restauração de um conjunto de estados de arquivos previamente existentes que podem não ter existido simultaneamente).

- recuperação para um estado valido

- recuperação para um estado consistente

- resistência a falhas (restaura o estado implicitamente)

A resistência contra Calhas difere dos outros tipos de recuperação, pois enquanto os outros tipos de recuperação restauram estados explicitamente, a resistência contra falhas mantém estados corretos através da forma com que os dados são manipulados durante o processamento normal.

As diferentes técnicas de recuperação usadas em sistemas de banco de dados quase nunca são empregadas separadamente. é bastante comum que SGBDs disponham de varias estratégias procurando combina-las para cobrir os diversos tipos de falhas. Também controlar várias técnicas distintas permite ao SGBD explorar particularidades dessas técnicas e, assim, torna-las mais eficientes e eficazes do que se operadas individualmente. Para uma melhor compreensão, algumas dessas técnicas de recuperação são descritas separadamente a seguir. Na seção III.7 deste capitulo são feitos comentários sobre o uso das varias técnicas descritas.

III.6.1 - Programa Restaurador

Estes programas simplesmente revêem todos os dados da banco de dados após a ocorrência de uma falha e tentam salvar o que for possível. Dados ou ate mesmo arquivos podem ser perdidos no processa se o algoritmo decidir que não pode recuperá-los (CASANOVA 1985).

O programa restaurador não utiliza dados de recuperação. Ele é geralmente usado quando outras técnicas de recuperação (que usam dados de recuperação) foram aplicadas sem sucesso, ou suando o sistema não é tolerante a falhas. O programa restaurador analisa o banco de dados após a ocorrência de uma falha, avalia os danos ocorridos e restaura o banca de dados rara algum estado válido, salvando as informações que ainda se mantem reconhecíveis após a falha.

O programa restaurador examina as estruturas de dados e tenta reconstruir o banco de dados ou restaurar sua consistência. Nesse processo pode acontecer a exclusão de alguns dados ou arquivos.

A situação típica quando programas restauradores são ativados se dá em SGBDs onde os algoritmos empregados em operação normal não garantem a consistência dos dados na memória secundária ativa a cada instante. Se o SGBD é acometido por uma falha primaria quando está justamente no processo de transferir dados da memória principal para a memória secundária ativa o conteúdo da primeira é destruido, impedindo que a ação seja completada. Neste caso, a memória secundária ativa, ainda não tendo sido completamente atualizada, representa um estado inconsistente do banco de dados. O programa restaurador vai, então, examinar os arquivos da memória secundaria ativa e tentar, de alguma forma, torná-los consistentes. É fácil imaginar situações onde os dados da memória principal, destruidos quando da ocorrência da falha, sejam de tal forma críticos que a única alternativa seria o programa restaurador eliminar parte dos dados residentes na memória secundária ativa a fim de trazer todo o banco de dados a um estado consistente.

Programas restauradores, geralmente entram em ação como um último recurso ou na ausência de mecanismos apropriados para contornar certos tipos de falhas. Embora seja um mecanismo primitivo de controle de integridade, o programa restaurador pode se tornar atraente em bancos de dados simples que implementem estruturas pouco sofisticadas.

III.6.2 - Descarga ("dump")

É sem dúvida a técnica mais empregada atualmente contra falhas secundárias. A idéia básica nesta técnica também é bastante simples. Periodicamente, o conteúdo de toda a memória secundaria ativa é descarregado para outros dispositivos ativos, geralmente fitas magnéticas, que são em seguida arquivadas como memória secundária dormente. Ocorrendo uma falha secundária, e outros mecanismos de restauração mais eficientes não operando a contento, O SGBD

pode apelar para esta cópia arquivada e retornar a um estado consistente, anterior, do banco de dados. Apenas falhas terciárias podem afetar as cópias arquivadas. Devemos observar que se estas se constituem nos Únicos objetos de que se pode valer o ÇGBD para voltar à normalidade, então todas as transações invocadas desde o inicia da descarga até o momento da ocorrência das falhas serão perdidas. Nenhum traço de sua existência, mesmo daquelas compromissadas, será notado após a recuperação do sistema. Em muitos casos isto pode ser intolerável. Métodos apoiados em descargas do banco de dados são, portanto, candidatos naturais a se aliarem a outras estratégias que tentam remediar este defeito, mantendo algum tipo de registro das atividades que ocorrem no espaço de tempo entre a obtenção de duas descargas consecutivas. Uma variação consiste em se descarregar não todo o banca de dados, mas apenas as partes do banco de dadas que tenham sido afetadas desde a última descarga. Este processo é conhecido como "dump incremental"

A periodicidade com sue as cópias são obtidas deve ser analisada com cuidado. Quanto mais frequentes, tanto mais eficazes serão no combate a falhas secundarias, visto que representam um estado mais recente do banco de dados em relação ao momento de ocorrência da falha. Por outro lado, maior frequência implicara em maior degradação do tempo de resposta do sistema, especialmente se o banco de dados for muito grande. Em (LOHMAN 1977) são feitas considerações a respeito da frequência para este tipo de operação.

Uma outra utilidade da descarga é reorganizar o banco de dados recarregando-o numa nova organização. Um exemplo disso é a consolidação da espaço livre deixada pela deleção de registros.

São dois os processos de criação de descarga: estático ou dinâmico.

III.6.2.1 - Descarga Estática

Usualmente a descarga do banco de dados é criada de maneira estática, ou seja, enquanto a descarga esta sendo feita, não é permitido que transações que atualizem o banco de dados sejam executadas. A frequência da criação de descargas é proporcional ao tempo necessário para reconstruir o banco de dados depois da ocorrência de uma falha. Quanto mais frequente a criação de descargas, menor é o tempo necessário para recontruir o banco de dados. Entretanto, a geração de descargas estáticas frequentes aumenta a quantidade de tempo que o sistema de banco de dados não pode ser utilizado por transações de atualização.

III.6.2.2 - Descarga Dinâmica

Nessa estratégia, enquanto a descarga está sendo gerada, transações que atualizam o banco de dados podem ser executadas. A descarga gerada dinamicamente deve representar um estado consistente do banco de dados em algum instante de tempo. Para isso, precisa incluir todas as atualizações feitas pelas transações que tenham terminado em uin determinado tempo "t", e nenhuma atualização feita por trançações que terminaram depois desse tempo "t".

Existem três métodos de geração de descargas dinãmicas (LANDES 1980):

a) Descarga da versão inicial: a descarga gerada representa o banco de dados que existia no começo do processo de geração da descarga.

b) Descarga da versão final: a descarga gerada representa o banco de dados que existe no momento do término do processo de geração da descarga.

c) Descarga da versão intermediária: a descarga gerada representa a versão do banco de dados que existia em algum tempo "t" durante a execução do processo de geração da descarga.

A descarga é uma das técnicas mais empregadas para prover a recuperação de um banco de dados face a ocorrência de uma falha.

III.6.3 - Diário ("log")

O diário, também conhecido como trilha auditora, ata ou "log", é um arquivo sequencial que contem um histórico das ações executadas no sistema (BJORK 1975).

O diário pode ser usado para diferentes propósitos, tais como:

a) Reconstruir o banco de dados: se após a ocorrência de uma falha uma descarga do banco de dados é reinstalada, um diário é utilizado para restaurar o estado em que o banco de dados se encontrava no momento da falha executando as operações necessárias sobre ele.

b) Desfazer atualizações: se o sistema falha, sem danificar a memória secundaria, as atualizações feitas por transações que estavam ativas no momento da ocorrência da falha devem ser desfeitas, restaurando o banco de dados para o estado em que estava antes do início destas transações. Para isso, o diário é processado desfazendo todas as atualizações realizadas no banco de dados. O efeito de uma transação sobre os dados do banco de dados tambdm pode ter que ser desfeito no caso de ocorreu um "deadlock" ou uma falha de transação. Em ambos os casos os dados afetados pela transação podem ser restaurados para seus estados anteriores ao inicio da transação.

c) Assegurar a integridade do sistema: aqui, o diário serve para verificar se todas as restrições de integridade definidas pelos usuários estão sendo respeitadas. Neste sentido o diário e usado com o propósito de segurança e auditoria.

Mesmo a idéia sendo simples, diários enfrentam problemas de sincronismo e de eficiência que geram mecanismos de controle de integridade bastante sofisticados (CASANOVA 1985). Diários garantem em primeira instância, proteção contra falhas primarias, mesmo porquê, a exceção de pseudo-falhas, estas são os tipos mais frequentes de falhas com que deve se preocupar um SGBD.

Num SGBD onde transações são as unidades básicas de trabalho do sistema, a ocorrência de uma falha encontrará algumas transações em andamento, isto é, transações não compromissadas ou canceladas. O SGBD deve, portanto, desfazer todos os efeitos parciais registrados por cada transação que ainda não completou até o instante da ocorrência da falha. Para que isto possa ocorrer, cada ação elementar deve registrar em um diário, os valores iniciais e finais de cada objeto modificado, bem como deixar claro em beneficio de que transação esta ação elementar foi invocada. Normalmente, o diário reside na memória secundária ativa. Ocorrendo uma falha primária, tudo que o SGBD precisa é consultar o diário e:

a) refazer transações confirmadas cujo efeito ainda não tenha sido registrado na memória secundária ativa;

b) desfazer transações canceladas cujo efeito ja foi registrado em memória secundária ativa;

Também deve ir para o diário um registro de inicio e término para cada transação executada. Assim, os mecanismos de controle de integridade do SGBD teriam condições de

identificar todas as transações em andamento a cada instante, bem como não teriam dificuldades em agrupar e associar ações elementares que correspondam a uma mesma transação. A própria ordem sequencial em que aparecem os registros de ações elementares no diário se presta muito bem ao processo de refazer/desfazer transações.

Um problema muito sutil nesta técnica, diz respeito ao sincronismo entre a entrada de registros no diário e a alteração de valores de objetos na memória secundaria ativa. Um bom procedimento é adotar um protocolo para uso do diário que garanta a gravação de cada ação elementar antes de registrar seus efeitos em memória secundária ativa.

Gray em (GRAY 1978), apresenta a protocolo de "gravar o diário na frente ("write ahead protocol" - WAL) onde:

a) Antes de gravar atualizações não compromissadas de uma transação em memória estável, força seus registros de valores anteriores do diário para membria estável.

b) Antes de compromissar atualizações de uma transação, força todos os seus resistros de valores atuais do diária para memória estável.

A razão disto é que a memória principal é volátil enquanto a memória secundária é não volátil. Com isso, se acontecer uma gravação em memória não volátil antes do registro diário ser gravado, a ocorrência de uma falha pode tornar impossível desfazer uma ação.

Diários devem residir em memória secundária ativa, por serem estruturas muito solicitadas uma vez que a invocação de cada ação elementar grava nele um registro. Uma falha que destrua parte da membria secundaria ativa, afetando segmentos do diário, poderia provocar danos no sistema, pois transações confirmadas cujos efeitos não foram ainda

registrados sobre os respectivos objetos simplesmente desapareceriam, uma vez ser o diária o Único registro destas transações capaz de fornecer informações suficientes para sue sejam refeitas.

Uma alternativa é duplicar o diário em memória secundária ativa de tal modo que as duas "cópias" residam em dispositivos físicos distintos. Como supomos que a ocorrência de falhas são eventos independentes, isto minimizaria as chances de que partes do diário sejam perdidas. Em principio, pode-se tornar o mecanismo tão confiável quanto se queira através da duplicação, triplicação, etc., do diário em unidades distintas.

Entretanto, o diário tem um custo. Para seu armazenamento utiliza-se uma area considerável de memória não volátil (principalmente se ele Cor replicado), utiliza-se "buffers" da memória principal e aumenta-se o numero de 1/05 na execução de cada transação.

Casanova em (CASANOVA 1985), observa que, se tomado isoladamente, o diário teria que registrar todas as ações elementares efetuadas contra o banco de dados desde que este foi criado. Esta situação é remediada com o emprego de descargas. A ultima descarga obtida seria revivida e, ao ser o diário, todas as transações que porventura completaram antes da descarga ter sido iniciada poderiam ser ignoradas. As demais seriam desfeitas ou refeitas, de acordo com a respectiva situação no instante da ocorrência da falha. Como podemos ver, descargas complementam naturalmente técnicas de controle de integridade que operam através de diários.

III.6.3.1 - Classificacão de Dados do Diário

Uma preocupação necessária quando se usa diário num sub-sistema de reconstrução é definir que informações gravar nu

diário para representar a operação efetuada, isto e, se deverão ser gravados os registros alterados, ou as páginas onde se encontram estes registros, ou instruções que efetuam alterações no banco de dados, etc. Conforme a escolha feita, as rotinas de gravação no diário se localizarão em um ou em outro ponto do SGBD, como por exemplo no analisador semântico (para gravar instruções) ou numa rotina de baixo nível que faz a gravação de paginas em disco.

Haerder e Reuter em (HAERDER 1983) classificam os dados do diário de duas maneiras:

a) referente aos tipos de objetos a serem gravados no diário. Se alguma parte da representação física for gravada nos referenciamos a ele como diário físico. Se os operadores e seus argumentos são gravados num nível mais alto, é chamado de diário lógico.

b) referente ao estado do banco de dados, ou seja, guardar as imagens antes ou depois de uma alteração, ou a transição que causa a alteração.

A união destas duas maneiras gera os diferentes tipos de diário descritos a seguir (HAERDER 1983):

a) Diário de Estado Físico a Nível de Página

é o método mais básico, usa a página como unidade de informação do diário. Cada ver que uma modificação é feita no espaço de endereçamento, toda a página que contém esse espaço é gravada no diário duas vezes: uma chamada "imagem anterior" ("before image") gravada antes da alteração ser consolidada com o propósito de "desfazer", e outra chamada "imagem posterior" ("after image") gravada após a alteração com o propósito de "refazer".

b) Diário de Transição Física a Nível de Página

Esta técnica também é baseada em paginas. Entretanto, em ver de gravar os estados antigos e novos de uma página explicitamente, ela grava no diário a "diferenca entre elas". A função que fornece a diferenca entre as duas cadeias de bits é a ou-exclusiva ("exclusive-or"). Se a diferença for aplicada para um estado antigo de uma página, usando novamente a função o resultado será o estado novo da página e vice-versa.

c) Diário de Estado Físico a Nível de Caminhos de Acesso

Este tipo de diário usa objetos do nível de caminho de acesso (registros, estruturas de caminho de acesso, tabelas, etc.). O componente do diário grava somente as entradas (declarações) alteradas destas estruturas em vez de páginas em volta. A vantagem deste método é que apenas aqueles objetos do nível de representação realmente afetados por modificações são gravados no diário, diminuindo assim a quantidade de informações gravadas e consequentemente o espaço necessário para o diário.

d) Diário de Transição a Nível de Caminhos de Acesso

No nível de caminho de acesso trabalha-se com as entradas das estruturas de armazenamento, mas não se sabe como elas estão relacionadas entre si em relação aos objetos do esquema do banco de dados. Este tipo de informação é mantido nos níveis mais altos da hierarquia de mapeamento. Se for enfocada apenas a representação física da entrada ("diário de transição física"), transição do estado neste nível significa que um registro físico, uma entrada de tabela, etc., é adicionado, excluído, ou modificado numa página. Os argumentos pertencentes a estas operações são as próprias entradas, e assim, existe uma pequena diferença entre esta e a abordagem anterior. No caso do "diário de estado físico a nível de caminho de acesso", o endereço físico é colocado junto com a

representação da entrada. Aqui, coloca-se o código da operação e o identificador do objeto com o mesmo tipo de argumento. Assim, "diário de transição física" neste nível não fornece nada essencialmente diferente.

e) Diário Lógico no Nível Orientado a Registros

Num nível mais acima, pode-se expreçsar as alterações executadas pelas transações de uma forma mas compacta, através da gravação de comandos de alteração da LMD e de seus parâmetros. A gravação no diário neste nível significa que somente as operações de insercão, alteração e deleção, junto com seus identificadores de registros e valores de atributos, são gravados no diário. O processo de mapeamento diferencia quais entradas são afetadas, quais paginas devem ser modificadas, etc. Assim, reconstrução é realizada pela reexecução de alguns comandos de LMD processados previamente. Por exemplo, comandos de LMD inversos devem ser executados, ou seja, um EXCLUIR compensa um INCLUIR, e vice-versa, e um ATUALIZAR retorna seu valor inicial.

Os comandos de LMD inversos devem ser gerados automaticamente como parte da atividade regular do diário, r por esta razão, esta abordagem não é viável para SGBDs orientados a rede que tem uma noção explícita de relacionamento entre registros.

Neste esquema o número de dados gravados no diário e pequeno, porém a recuperação é mais cara que a de outros esquemas.

A figura III.8 (HAERDER 1983), apresenta um sumário das propriedades das técnicas apresentadas, considerando-se dois aspectos: o custo da captação de dados do diário durante processamento normal e o custo da recuperação baseada no tipo de informação do diário. Os custos são basicamente medidos em unidades de operações físicas de I/O. O número do nível de mapeamento rorresponde à seção III.5.1.

Técnica	No. Nível Mapeamento	Custo durante proc. normal	Custo operações de recuperação
Estado físico	2	Alto	Baixo
Transição física	2	Médio	Baixo
Estado físico	3	Baixo	Baixo
Transição lógica	4	Muito Baixo	Médio

Figura III.8 - Comparação entre diferentes tipos de diário.

III.6.3.2 - Pontos de Verificação ("checkpoints")

Os métodos de psnto de verificação gravam informações em memória estável (no próprio diário, no banco de dados ou num arquivo de pontos de verificação) durante o processamento normal. O objetivo da existência de pontos de verificação é facilitar e agilizar ações de reconstrução quando da utilização de diários, uma ver que eles delimitam a quantidade de trabalho a ser feita na reconstrução. A questão da eficiência do reinicio é muita importante, uma vez que o banco de dados não fica disponível para as usuários até que a reinício se complete.

A questão básica é a seguinte: quando ocorre uma falha de sistema, as transações incompletas devem ser desfeitas e as completas cujos efeitos ainda não se refletiram no banco de dados devem ser refeitas. Porém, não se sabe ate onde deve-se reprocessar o diário e nem em que ponto do diário tem-se a certeza de que todas as transações anteriores estão completas no banco de dados. Para solucionar estas questões surgiu o conceito de ponto de verificação.

A seguir são mostrados quatro critérios para determinação de quando iniciar as atividades doç pontos de verificação (HAERDER 1983):

a) Pontos de verificação orientados a transações (TOC - "Transaction-Oriented Checkpoints").

Uma disciplina de FORÇAR (seção III.5.4) permite um rápido reinício, por não haver necessidade de reconstrução de transações. Todas as páginas modificadas são propagadas antes que um registro de fim de transacão (EOT) seja escrito no diário, o qual torna a transação durável. Se este rrgistro não for encontrado no diário após uma falha, a transação será considerada incompleta e será desfeita. Assim, o registro EOT de cada transação pode ser interpretado como um ponto de verificação, o que limitará o escopo de um REDO.

Como pode ser visto na figura 111.9, pontos de verificação orientados a transações são inferidos por uma disciplina de FORÇAR (seção III.5.4). Na figura, as transações Ti e T2 já estavam encerradas quando ocorreu a falha. A transação T3 ainda estava em andamento.

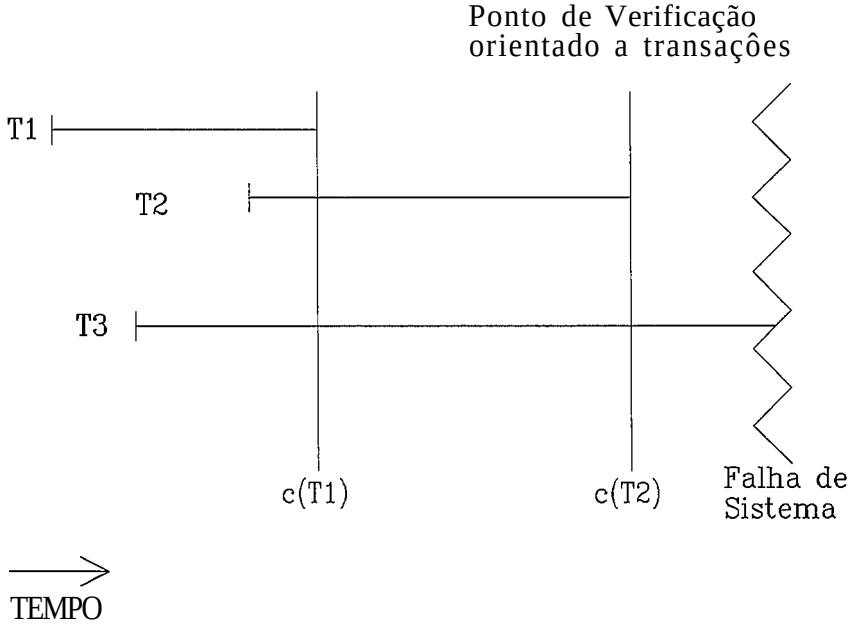


Figura III.9 - Exemplo de ponto de verificação orientado a transação.

b) Pontos de verificação consistentes a nível de transação (TCC - "Transaction-Consistent Checkpoints").

Na geração deste tipo de ponto de verificação todas as transações de atualização são completadas e nenhuma outra transação nova e admitida até que o ponto de verificação seja completado. Assim, o ponto de verificação é gerado quando a última atualização for completada, como mostra a figura III.10.

Fazer o ponto de verificação significa propagar todas as paginas de "buffer" modificadas para disco e gravar um registro de ponto de verificação no diário, indicando que neste ponto foi estabelecido um estado consistente em relação a transações nu banco de dados materializado.

Na figura 111.10, a transação **T3** deve ser refeita completamente, enquanto T4 deve ser desfeita. Não há nada a fazer com as tansações Ti e T2, uma vez que suas atualizações foram propagadas na geração do ponto de verificação Ci. Ao ocorrer uma falha de sistema, só é necessário voltar até o Último ponto de verificação, desfazer T4 e refazer **T3.**

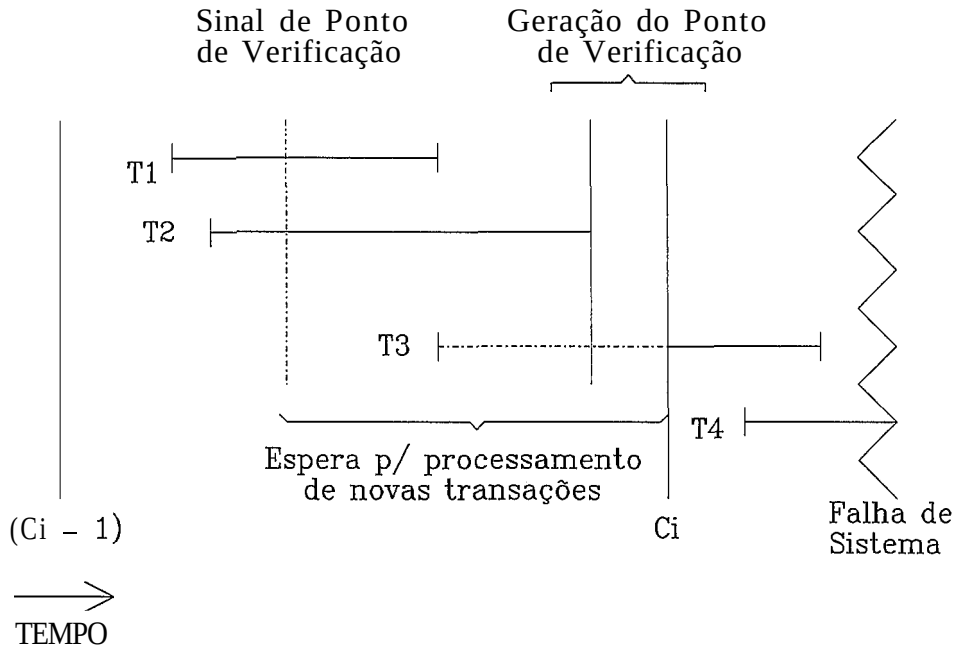


Figura III.10 - Exemplo de ponto de verificação consistente a nível de transação.

Esta abordagem não e aconselhável para grandes SGBDs acessados por vários usuários devido as seguintes razões:

b.1) muitas transações podam ficar esperando para iniciarem suas execuções;

b.2) o tempo de gravação do5 "buffers" pode ficar muito longo quando muitas paginas alteradas se acumulam.

c) Pontos de verificação consistentes ao nível de ações (ACC - "Action-Consistent Checkpoints").

Uma transação é considerada uma sequência de ações elementares que afetam o banco de dados (podem ser vistas como comandos de LMD). Um ponto de verificação deste tipo só pode ser gerado quando não existir nenhuma ação de atualização sendo processada. Quando um ponto de verificação é sinalisado, ações em execução são concluídas e as novas ações são retardadas ate que o ponto de

verificação seja concluído. O ponto de verificação é gerado da mesma maneira que na técnica anterior, porém, indicando neste caso não a consistência da transação mas da ação.

No exemplo da figura III.11, após ocorrer uma falha, deve-se refazer T5 e T6 e desfazer T1, T2 e T3. As transações T4 e T7 já fazem parte do banco de dados materializado.

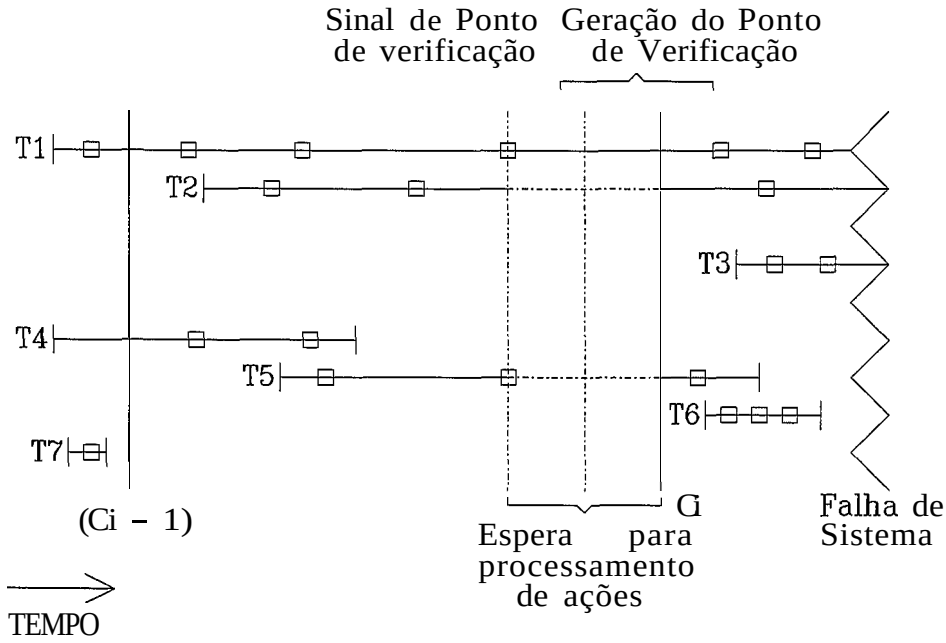


Figura III.11 - Exemplo de ponta de verificação consistente a nível de ação.

Apesar deste esquema seu mais realista, uma ver que não causa retardos demorados, o custo do ponto de verificação permanece alto quando grandes "buffers" são usados.

d) Pontos de Verificação Parciais ("Fuzzy Checkpoints")

Visando diminuir os custos do ponto de verificação e a atividade de propagação no momento do ponto de verificação, pode-se usar as técnicas de ponto de verificação chamadas indiretas. Nestas técnicas, as informações sobre a ocupação dos "buffers" são gravadas no diário em vez das próprias

paginas. Isto pode ser feito com duas ou três operações de gravação, mesma com grandes "buffers", e ajuda a determinar quais paginas contendo dados campromissados estavam nos "buffers" no momento da falha. A passagem dos "buffers" para disco e feita normalmente pelo gerenciador de "buffers" e não no momento do ponto de verificação.

Esta técnica é usada somente para propagações NÃO ATÔMICAS, uma vez sue em propagações ATÔMICAS as paginas são escritas em disco nos pontos de verificação e as não propagadas são perdidas após a ocorrência de falhas de sistrma.

III.6.4 - Arquivos Diferenciais

No esquema de arquivos diferenciais, todos os arquivos lógicos residem em dois arquivos físicos (AGRAWAL 1985): um arquiva base somente de leitura e um arquivo diferencial de leitura e gravação. O arquivo base permanece inalterado ate a reorganização. Todas as atualizações são feitas na arquivo diferencial.

Esta técnica adota a filosofia de coletar todas as modificações efetuadas contra certas entidades do banco de dados, em particular arquivoç de dados, em estruturas próprias, especificamente planejadas para **este** fim, mantendo os valores dos objetos originalmente associados as estas entidades inalterados. Estas estruturas, onde as modificações são agrupadas, são denominadas arquivos diferenciais. Tudo se **passa** coma se o SGBD, ao concentrar as modificações sobre os arquivos diferenciais, atrasasse o momento **de** tornar irreversível o efeito de ações elementares **sobre** o banco de dados, mesmo daquelas correspondentes a transações que já foram confirmadas. A vantagem principal é óbvia. Os originais formam uma cópia que reflete um estado completo e consistente anterior da banco de dados. Ocorrendo uma falha que venha a prejudicar

os arquivos diferenciais tudo que o SGBD tem a fazer é reler da memória secundária ativa os originais intactos, eventualmente desperdiçando os trabalhos das Ultimas transações. Porém, se o usuário já Coi informado de que determinada transação se confirmou, ignorá-la nesse ponto pode ser inadmissível. Para maior segurança, os originais e os respectivos arquivos diferenciais podem ser mantidos em dispositivos físicos distintos. A descarga dos originais em memória secundaria dormente protege o banco de dados contra qualquer falha secundaria.

No entanto, em algum momento os arquivos diferenciais terão de ser usados para reconstruir os originais, sendo este processo oneroso.

Nesta técnica, o arquivo diferencial é sempre pesquisado primeiro. Se o dado não for encontrada no arquivo diferencial deve-se pesquisar o arquivo principal. O algoritmo descrito na figura III.12 representa a leitura de um registro (SEVERANCE 1976).

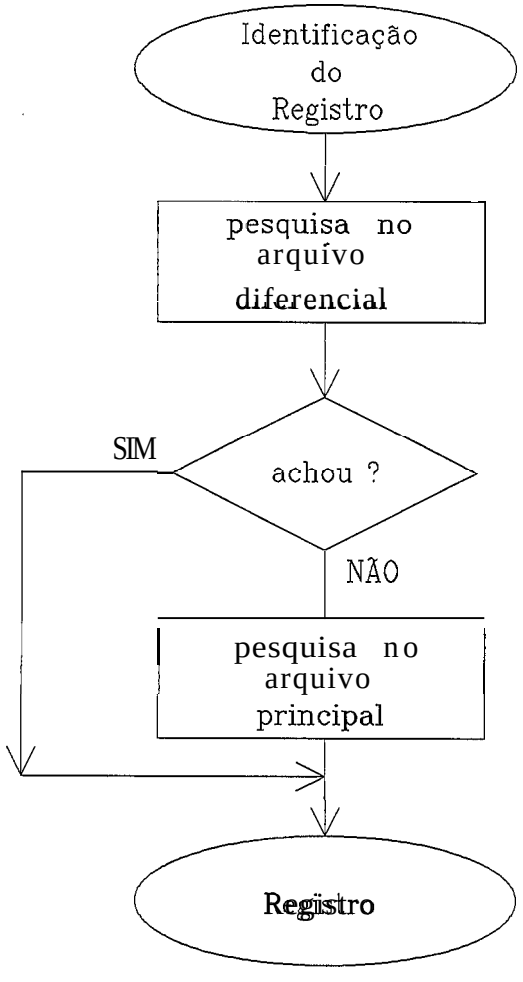


figura III.12 - Algoritmo de leitura de um registro

Severance e Lohman em (SEVERANCE 1976) sugerem uma estratégia de pesquisa que usa filtros a fim de reduzir o número de acessos desnecessarios ao arquivo diferencial.

A técnica associa o arquivo diferencial com um vetor de bits B de tamanho M na memória principal, e com algum número X de funções "hashing" que geram a partir de uma identificação de um registro, um endereço no vetor B. Quando o arquivo diferencial está vazio (inicialmente), todos os bits de B são iguais a 0 (zero). Toda vez que um registro é armazenado no arquivo diferencial, cada transformação é aplicada para o identificador do registro e cada bit X enderecado e setado para 1 (um). Ao recuperar-se

um registro R, pode-se saber se o registro não se encontra nu arquivo diferencial.

A recuperação de um registru R usando filtros é ilustrada pelo diagrama da figura III.13.

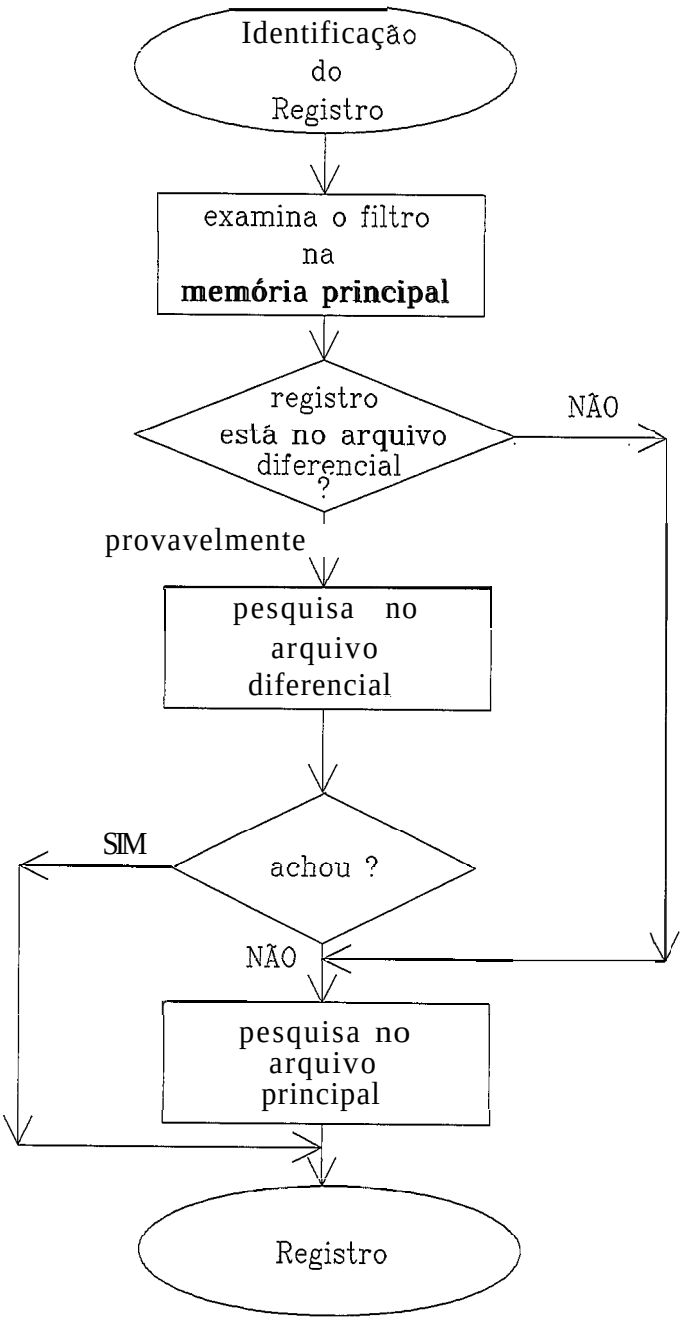


Figura III.13 - Exemplo de recuperação de um registro usando filtro.

As vantagens da técnica de arquivos diferenciais são descritas em (SEVERANCE 1976). Entre elas podemos citar:

a3 redução do custo da descarga: uma vez que o tempo necessário para a descarga é proporcional ao volume de dados copiados, o arquivo diferencial pode reduzir drasticamente o custo da descarga em grandes bancos de dados, particularmente quando o número de registros alteradas durante o período da descarga for pequeno;

b) facilita a descarga incremental: a implementação de arquivos diferenciais no geral, onde cada registro e sequencialmente alocado na memória secundaria, contribui naturalmente para essa estratégia;

c) permite fazer descarga dinâmica;

d) acelera recuperação de perda de dados;

e) simplifica o desenvolvimento de software: o fato de que o arquivo de dados principal e seu índice associado não são afetados par atualizações, proporciona um ambiente natural para desenvolvimento de software. Se forem usados dois arquivos diferenciais, pode-se ter um sistema de desenvolvimento e outra de produção executando em paralelo, ambos acessando o mesmo arquivo principal mas modificando seu próprio arquivo diferencial.

Essa técnica apresenta as seguintes desvantagens (SEVERANCE **19763:**

a) O acesso a um dado pode ser lento, pois pode requerer dois acessos ao banco de dados por consulta (um no arquivo diferencial e outro no principal);

b) O arquivo diferencial eventualmente deve ser fundido com o arquivo principal. O mais grave disso, e que esta operação exige uso exclusivo dos dois arquivos;

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

Em (AGHILI 1982) encontra-se um estudo interessante sobre o uso de arquivos diferenciais em termos de um modelo analítico baseado em alguns critérios escolhidos.

III.6.5 - Cópia e Versão Corrente

Essa técnica consiste em manter duas copias do banco de dados a fim de poder realizar uma possível restauração do banco de dados para um estado prévio (VERHOFSTAD 1978). Uma das cópias e a versão corrente do banco de dados, composta pelos arquivos que contém as últimas atualizações feitas nos dados, a outra copia é a versão de respaldo consistente do banco de dados, composta pelos arquivos que armazenam os valores prévios dos dados.

Nessa técnica as modificações feitas pelas transações são realizadas na versão corrente do banco de dados, produzindo assim uma nova versão. No caso da ocorrência de uma falha, o estado atual do banco de dados pode ser recuperado através da cópia (versão anterior) e das transações que acessaram essa copia.

Como exemplo desta técnica temos alguns editores de arquivos, que criam uma nova versão do arquivo editado pelo usuário mantendo a versão original intacta durante a edição. No caço do usuário cometer um erro durante a adição, somente a versão atual do arquivo e afetada, possibilitando uma reconstrução do estado anterior. Quando o usuario encerra a edição, a versão atual permanece e a versão anterior é descartada.

III.6.6 - Múltiplas Cópias

Nesta tecnica duas ou mais versões do banco de dados são mantidas, e alteradas pelas mesmas transações paralelamente, porem, respeitando uma sincronização que permita em todo momento ter uma versão consistente do banco de dados reconhecível pelo sistema.

A técnica de múltiplas cópias engloba duas outras técnicas (VERHOFSTAD 1978):

a) voto da maioria: mantém-se um número ímpar de cópias dos dados. Se a maioria das cópias tiver o mesmo valor, então este valor é tomado como correto. Esta técnica é usada em sistemas sue exigem muita seguranca. Esta tecnica pode ser usada para detectar problemas em processadores. Vários processadores recebem a mesma entrada, trabalhando sobre cópias de dados identicas. Os que tiverem saída diferente da maioria, estão efetuando mal alguma operação.

b) duplicação: mantém-se duas cópias, cada uma com um "flag" para indicar atualizações em progresso. Uma cópia inconsistente (ou suspeita) e sempre reconhecida como inconsistente pelo valor do seu "flag". Se o sistema falhar durante a atualização de uma das cópias, o "flag" continuará ligado. Esta cópia estará inconsistente e sera descartada. A outra cópia permanece consistente, uma ver que atualizações sobre cópias nunca são feitas no mesmo instante. Só depois que uma acaba é que a outra começa. A cópia consistente só terá dois estados: já alterada ou ainda não alterada. Esta técnica pode ser utilizada para controlar a alteração de diretórios.

Com a técnica de múltiplas cdpias, as varias cdpias devem ter sempre o mesmo valor, exceto durante as atualizações. Nesse sentido, a tecnica de múltiplas cópias tem a característica de prover resistência a falhas, mas no mínimo duplica os custos de armazenamento e atualização. Com ela, 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 cópias outras poderão manter o sistema em funcionamento. A cópia inconsistente pode sempre ser reconhecida e ignorada.

A diferença principal desta técnica, se comparada com a de cópia e versão corrente, é que na técnica de múltiplas cópias todas as copias se encontram ativas, enquanto que na outra técnica só uma das copias fica ativa.

III.6.7 - Paginação Sombra

A idéia básica do mecanismo de "paginação sombra" é evitar atualizações "in-place" nas estruturas de dados. A alteração é feita numa cópia do objeto (registro, pagina, bloco) que está sendo modificado, sendo o original substituido somente se a atualização for bem sucedida. Com a técnica de paginação sombra as duas cópias do objeto **só** são mantidas durante a atualização. Desta forma, consegue-se ter um banco de dados resistente a falhas.

No mecanismo de paginação sombra, o mapeamento dos itens de dados para a memória estável é dinâmica. Assim sendo, e conveniente implementar este mapeamento usando diretórios, com uma entrada por item de dado, contendo o nome do item de dado e sua localização na memória estável. Cada diretório identifica um estado diferente do banco de dados (BERNSTEIN 1987).

Neste esquema, todas as atualizações de uma transação Ti devem estar no banco de dados materializado na hora que Ti é compromissada (evita refazer transações), e nenhuma das atualizações de Ti pode estar no banco de dados

materializado antes que Ti seja compromissada (evita desfazer transações). Portanto, para eliminar tanto "undo" como "redo", todas as atualizações de Ti devem ser gravadas no banco de dados materializado numa Única operação atômica, na hora que Ti sofrer o comprometimento.

A dificuldade está em organizar as estruturas de dados como uma ação atômica. Isto e, deve-se instalar indivisivelmente todas as atualizações das transações no banco de dados materializado e inserir Ti numa lista de comprometimento.

O mecanismo de paginação sombra alcança estes objetivos da seguinte maneira: o local do Último valor compromissado de cada item de dados é gravado num diretório, armazenado na memória estável, e possivelmente mantido na memória cachê para um rápido acesso. Existem também diretórios de trabalho que apontam para versões não compromisçadas dos mesmos itens de dados. Juntos, estes diretórios apontam para todas as imagens anteriores e posteriores que poderiam estar armazenadas no diário normalmente, mas que neste esquema não é mantido como um arquivo sequencial separado.

Quando uma transação Ti grava um item de dado x, uma nova versão de x é criada na memória estável. O diretório de trabalho que define o estado do banco de dados usado por Ti é atualizado para apontar para esta versão. Quando Ti é compromissada, o diretório que define o estado do banco de dados compromissado é atualizado para apontar para as versões sue Ti gravou. Isto faz com que os resultados das gravações de Ti se tornem parte do estado do banco de dados compromissado.

Com esta estrutura, um procedimento de camprometimento necessita mudar atomicamente as entradas do diretório para todos os itens de dados gravados pela transação sue está sendo compromissada. Se o diretório cabe num Único item de dado, então o problema esta resolvido. Caso contrario, parece que simplesmente o problema foi movido para uma

estrutura diferente. Em vez de instalar atualizações atomicamente no banco de dados materializado, agora é necessário instalar atualizações atomicamente no diretório.

A diferença critica é que uma vez que o diretório é muito menor que o banco de dados, é possível manter duas cópias dele na memória estável: um diretório corrente, apontando para o banco de dados compromissado, e uma cópia de trabalho. Para compromissar uma trançação Ti, o gerenciador de recuperação atualiza o diretório de trabalho para representar a estado do banco de dadas materializado que inclui as atualizações de Ti. Isto e, para cada item de dado x que Ti atualiza, o gerenciador de recuperação faz uma entrada no diretorio de trabalho para x apontando Para a nova versão de x em Ti. Para itens de dados que Ti não atualizou, ele faz as entradas do diretório de trabalho idênticas às entradas do diretório corrente. Então ele troca os diretórios corrente r de trabalho numa ação atômica. Esta ação atômica de troca é implementada através de um registro "mestre" na memória estável, o qual tem um bit indicando qual das duas cópias do diretório é a corrente. Para trocar as diretórios, o gerenciador de recuperação simplesmente complementa o bit no registro "mestre", o que é certamente uma ação atômica. A gravação deste bit é a operação que faz o comprometimento da transação. É importante observar que o gerenciador de recuperação só pode processar um comprometimenta por vez. Isto é, a atividade de atualização do diretório de trabalho seguida pela complementação do bit do registro "mestre" é a parte critica.

A figura III.14 ilustra as estruturas usadas no algorítmo de comprometimento da trançação Ti que atualizou itens de dados x e y. Na figura (a) a transação criou duas novas versões, deixando as versões antigas intactas como sombras. Na figura (b) Ti setou o diretório sombra para refletir o banco de dados materializado como se estivesse depois do seu comprometimento. Na figura (c) o bit do registro "mestre" é trocado, fazendo o comprometimento de

Ti e instalando **suas atualizações** no banco de dados materializado.

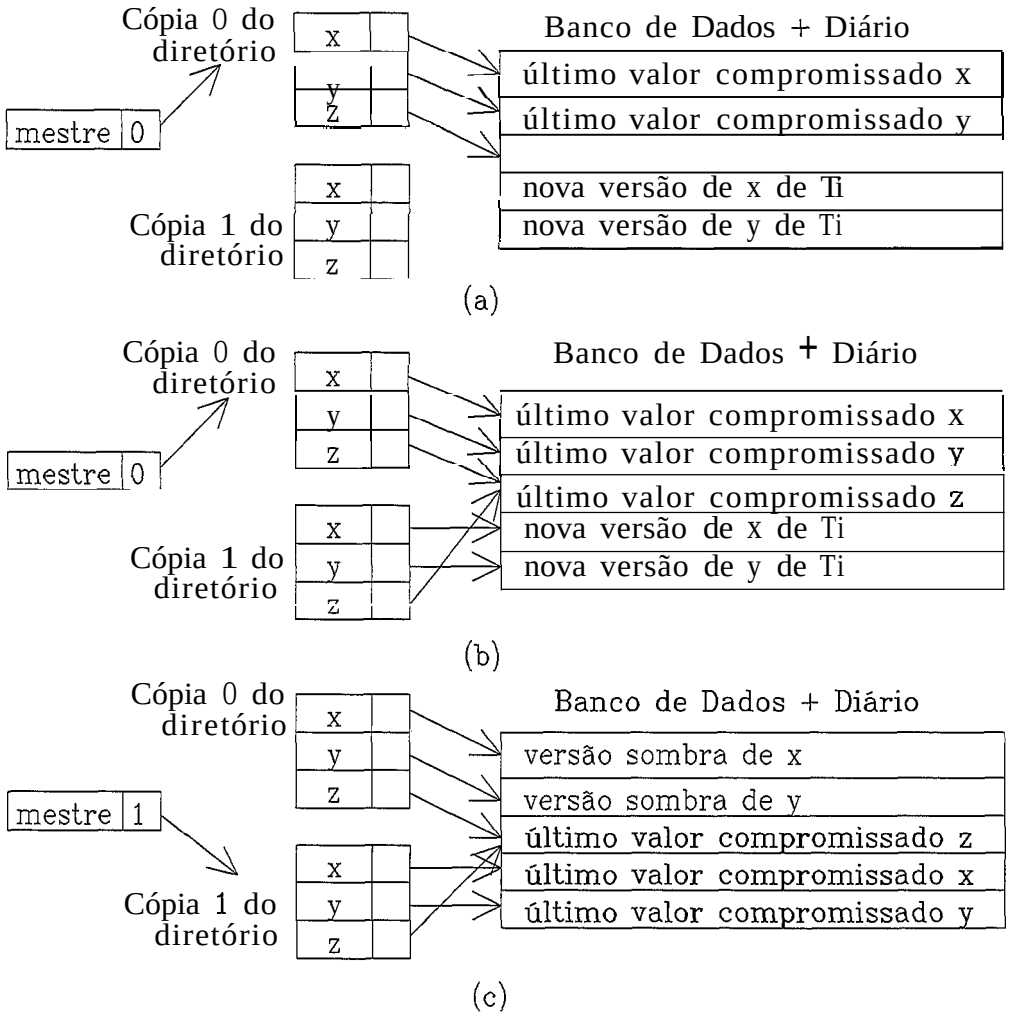


Figura III.14 - Um exemplo do esquema de paginação sombra.

(a) Estado do banco de dados depois da **criação** de novas versões para Ti.

(b) Estado do banco de dados depois de preparar o diretório para o comprometimento de Ti.

(c) Estado do banco de dados **depois** do comprometimento de Ti.

Entre as vantagens oferecidas pela **paginação** sombra **com relação** as técnicas baseadas **em** diário podemos citar:

a) a eliminação do "overhead" da gravação do registro diário;

b) a recuperação significativamente mais rápida de falhas (já que não são necessárias operações de "undo" e "redo").

Existem também a5 seguintes **desvantagens:**

a) Acessos a memória estável são indiretos e por isso mais caros. Entretanto, este custo pude ser pequeno se o diretório for suficientemente pequeno para ser armazenado na memória cachê.

b) Como custo mais importante, temos a fragmentacão dos dados, ou seja, o movimento de dados para novas versões destroi o "lay-out" original do banco de dados estável. Isto é, suando um item de dados e atualizado, pode não existir espaço para a nova cópia perto do local dos itens de dados originais. Quando a atualização é compromissada, o item de dados troca de lugar da sombra para a nova versão. Assim, se o banco de dados é projetado para que itens relacionados sejam armazenados em locais da memória estável próximos, este projeto ficará comprometido através do tempo.

c) Cada vez que uma transação é concretizada, as paginas do banco de dados que contem a versão antiga dos dados modificados pela transação, se tornam inacessíveis. Estas páginas **são** consideradas lixo, já que não são parte do espaço livre e não contem informações úteis. Assim, periodicamente é necessário encontrar todas as páginas de lixo e adicioná-las à lista de páginas livres. Este processo, chamado coleta de lixo, impõe "overhead" adiciona! e complexidade ao sistema.

Como um mecanismo de transação baseado somente em sombras suportando múltiplos usuários e considerado muito difícil e caro de implementar (GRAY 1981b), nesses sistemas ele é combinado com um diário incremental de todas as ações

que uma transação executa. O diário é usado para desfazer e refazer transações sobre arquivos compartilhado5 e e usado em combinação com sumbras no ponto de verifícação e reinício do sistema. A estratégia combinada reflete a evolução de um sistema monousuário (onde o esquema de paginação sombra funcionava bem) para um sistema multiusuário. Em (LORIE 1977) o esquema de paginação sombra é apresentado detalhadamente.

III.7 - Uso das Técnicas de Reconstrução

Segundo Verhofstad em (VERHOFSTAD 1978), as técnicas de reconstrução fornecem recuperação, resistência a falhas e manutencão de consistência de uma das três maneiras:

a) Pelo modo no qual o dado é estruturado. As múltiplas cópias, o arquivos diferenciais e cópia e versão corrente são parte da estrutura do banco de dados.

b) Pelo modo que os dados são alterados e manipulados. A técnica de paginacão sombra é um modo resistente a falhas de alualização de estruturas de dados complexas.

c) Através de utilitários. O programa restaurador, a descarga e a facilidade do diário, são utilitários que não tem relação com a estrutura de dados ou o modo de alteração dos mesmos. Podem ser adicionados a qualquer banco de dados.

Algumas das tdcnicas apresentadas anteriormente (seção III.6) se complementam bem, outras, se utilizadas no mesmo sistema, serão redundantes.

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 banco de dados a um estado correto anterior a falha. Ele meramente

"recolhe" o que restou após a ocorrência de uma falha, salvando o que for possível do banco de dados quando nenhuma outra técnica for capar de restaurá-lo. Pode também ser usado para deixar o banco de dados pronto para utilizar outra técnica qualquer.

A descarga incremental, o diário, os arquivos diferenciais e a cópia e versão corrente são consideradas técnicas altrrnativas, mas podem ser usadas como complementares, dependendo do grau de segurança necessário. Quando a perda das alterações feitas não for problema, pode-se usar uma combinação da cópia e versão corrente, descarga incremental e um programa restaurador. Quando a perda de qualquer alteração já feita for crítica, deve-se usar o diário, ou paginação sombra, ou múltiplas copias, ou arquivos diferenciais, ou ainda uma combinação delas.

O diário é uma alternativa para as tecnicas de arquivos diferenciais, paginação sombra ou múltiplas cópias; podendo ser usado para restaurar o estado correto após a ocorrência de uma falha e pode ser complementado com o uso de descargas.

A técnica de arquivos diferenciais torna muito fácil a implementação de descarga incremental. Essas duas técnicas podem complementar muito bem uma a outra.

As técnicas de múltiplas cópias e paginação sombra podem ser usadas alternativamente ou como complementares. Ambas provêm resistência contra falhas do mesmo tipo.

Quanto a comparação de custo e "overhead" das técnicas, pode-se dizer o seguinte (VERHOFSTAD 1978):

a) O custo do programa restaurador depende completamente da frequência de ocorrência das falhas. O "overhead" e acumulado somente quando o programa é utilizado.

b) A descarga e a cópia e versão corrente são consideradas as melhores técnicas para recuperaçao de falhas envolvendo meio de armazenamento. O "overhead" dessas duas técnicas é aceitável porquê seus pontos de verificação não são muitos fequentes.

c) O diário pode causar um "overhead" alto, dependendo da sua implementação, pois todas as alterações feitas contra o banco de dados devem ser registradas nele.

d) Se ar falhas acorrem com pouca frequência, arquivos diferenciais e paginação sombra geram um grande "overhead". No entanto, elas deixam o banco de dados resistente a falhas e mantém seu estado correto. Outras técnicas como descarga, cópia e versão corrente ou programa restaurador, deverão ser utilizadas para combater falhas que essas duas não conseguem.

e) A técnica de múltiplas cópias causa um "overhead" demasiadamente grande.

CAPÍTULO IV

EXEMPLOS DE SUB-SISTEMAS DE RECONSTRUÇÃO

Este capítulo tem como objetivo estudar alguns sub-sistemas de reconstrução existentes. Para isso, descrevemos dois sistemas relacionais, o Sistema-R e o INGRES e três propostas de sistemas orientados a objetos, o CACTIS, o GemStone e o POSTGRES. Neste capítulo também descrevemos os principais aspectos de reconstrução relacionados com bancos de dados distribuídos.

IV.1 - Descrição de Dois Sistemas Relacionais e de seus Sub-Sistemas de Reconstrução

Entre os vários SGBDs existentes dois foram escolhidos para serem estudados e descritos: o Sistema R e o INGRES. Dentre os motivos que nos levaram a essa escolha podemos citar a importância de suas implementações para a área de banco de dados relacionais, a relativa facilidade de se encontrar documentação a respeito de seus projetos, e a diferença entre as duas abordagens, o que torna o estudo mais interessante. A apresentação das características dos SGBDs é feita de forma resumida, objetivando situar o sub-sistema de reconstrução no sistema como um todo.

IV.1.1 - Sistema R

O Sistema R em (ASTRAHAN 1976) e (GRAY 1981b), é um protótipo que suporta o modelo de banco de dados relacional projetado e desenvolvido pela IBM, dando origem aos produtos SQL/DS e DB2. O objetivo foi fazer um sistema relaciona1 para resolver problemas reais em ambientes reais

tendo um desempenho comparável com o dos sistemas existentes até então.

IV.1.1.1 - Estrutura do Sistema R

O sistema *A* consiste de uma camada externa chamada de RDS ("Relational Data System") e de uma camada interna chamada RSS ("Relational Storage System").

O RDS fornece o modelo de dados relacional e seus operadores. Também fornece o gerenciamento de catálogos, dicionário de dados, autorização e visões alternadas dos dados. O RDS é manipulado através da linguagem SQL. O compilador SQL mapeia os comandos SQL em sequências de chamadas RSS.

O RSS é um método de acesso a dados do tipo um registro por vez. Suporta as noções de arquivo, tipo de registro, instância de registro, campo dentre de registro, índice, conjuntos do tipo "pai-filho" e cursor. O **RSS** suporta também a noção de transação (unidade de recuperação consistindo de uma sequência de ações RSS), assumindo toda a responsabilidade pela execução de transações concorrentes e assegurando que cada transação tenha uma visão consistente do banco de dados. O RSS também é responsável por recuperar os dados para seu estado mais recente de consistência em caso de falhas ou cancelamento.

IV.1.1.2 - Arquivas

Todos os dados persistentes do Sistema R são armazenados em arquivos. O usuário pode definir qualquer número de arquivos. Um arquivo, nesta proposta, é um espaço linear paginado de até 68 bilhões de bytes, que e dinamicamente alocado sobre disco em paginas de 4096 bytes. Um

gerenciador de "buffers" mareia todos os arquivos num conjunto de "buffers" em memória virtual compartilhado por todos os usuários. O gerenciador de "buffer" usa o algorítmo LRU ("Least Recently Used") para regular a ocupação das paginas. O conjunto de "buffers" é volátil e não sobrevive a um reinício do sistema.

Arquivos são separados em sombreados ("shadowed") e não sombreados. Arquivos não sombreados não tem recuperação automática. O usuário é responsável por fazer e guardar cópias redundantes destes arquivos. O Sistema R simplesmente atualiza paginas de arquivos não sombreados no conjunto de "buffers". Atualizações neste tipo de arquivo são gravadas em disco quando as paginas são removidas do conjunto de "buffers" e quando o arquivo é salvo ou fechado.

O RSS mantém duas versões "on-line" de arquivos sombreados, uma versão sombra e uma versão corrente. Ações RSS afetam somente a versão corrente do arquivo e nunca alteram a versão sombra. A versão corrente de um arquivo rode ser salva como versão sombra tornando as atualizações recentes no arquivo permanentes. A versão corrente também pode ser restaurada para uma versão sombra desfazendo todas as atualizações recentes no arquivo (figura IV.1).

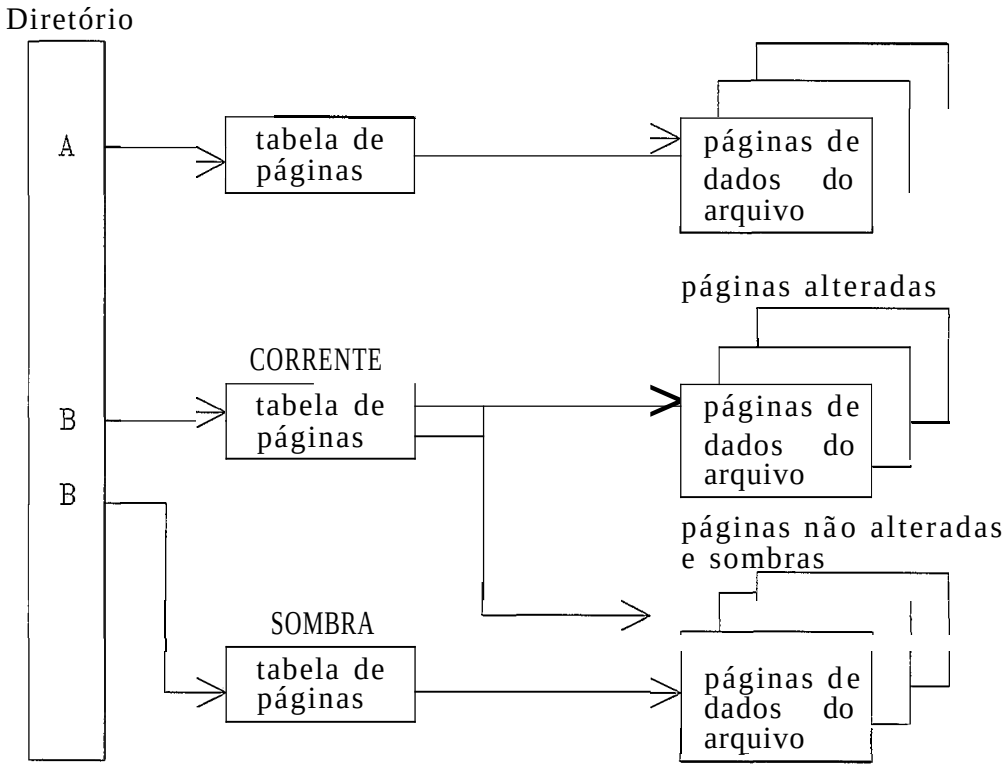


Figura IV.1 - A estrutura do diretório para arquivos sombreados e não sombreados.

A versão corrente de um arquivo não sobrevive a um reinício porquê suas atualizações recentes continuam residindo no conjunto de "buffers". No reinício do sistema todos os arquivos não sombreados tem seus valores como estavam no momento da falha do sistema e todos os arquivos sombreados são reajustados para suas versões sombras.

As versões corrente e sombra de um arquivo são implementadas de maneira eficiente. Quando uma página sombra é atualizada no "buffer" pela primeira vez, uma nova estrutura de página em disco é assinalada para ela. Quando essa página e lida do "buffer" ou gravada no "buffer", a nova estrutura é usada (a sumbra nunca é atualizada). Salvar um arquivo significa gravar em disco todas as páginas do arquivo alteradas correntemente! no "buffer" e gravar no disco a nova tabela de paginas liberando as

páginas sombras substituídas. Para restaurar um arquivo e necessário descartar páginas deste arquivo no "buffer", liberar todas as páginas novas do disco deste arquivo e, retornar a tabela antiga de paginas sombras.

IV.1.1.3 - Transação

O sistema de gerenciamento da transação e responsável pela programação das atividades do sistema, gerenciamento dos recursos físicos e gerenciamento da derrubada e reinício do sistema. Isto inclui componentes que executam planejamento, recuperação, "logging" e bloqueio.

A vida de uma instância de transação e simples. Uma mensagem ou pedido chega e faz com que um processo seja construido e emita uma ação BEGIN_TRANSACTION que estabelece uma unidade de recuperação. Em seguida uma série de ações são emitidas contra o estado do sistema e finalmente é emitida uma ação COMMIT_TRANSACTION que torna pública as saídas da transação. Alternativamente, se acontecer algum problema na execução da transação, pode ser emitido uma ação de ABORT_TRANSACTION cancelando todas as ações executadas por essa transação. Ações serão completadas com sucesso ou não modificarão o estado do sistema de maneira nenhuma.

Antes de uma transação completar, ela pode ser abortada tendo suas atualizações para dados recuperáveis desfeitas. O aborto pode vir pela própria transação (suicídio: entrada de dados ruim, operador cancela, etc.) ou por outro lado (assassinato: "deadlock", "timeout", falha do sistema, etc.). Entretanto, uma vez sue ela seja compromissada, seus efeitos não podem ser desfeitos. Para desfazê-los e necessário uma compensação, ou seja, executar uma nova transação que corrija os erros da anterior.

IV.1.1.4 - Pontos de Salvamento

O RSS define o conceito adicional de pontos de salvamento ("save points"). Um ponto de salvamento é um ponto limite que permite, quando uma transação tem problemas, voltar somente ate ele em vez de desfazer todo o trabalho da transação.

Pontos de salvamento não comprometem qualquer atualização da transação. Cada ponto de salvamento e numerado, o inicio da transação é o ponto de salvamento 1 e os sucessivos são 2, 3,.... O usuário pode salvar alguns dados em cada ponta de salvamento e restaurar estes dados se ele voltar para este ponto. Voltar ao ponto de salvamento i recompõe a transação ao seu estado inicial.

O RDS também utiliza pontos de salvamento para implementar certas operações complexas. Através deles ele garante a atomicidade de instruções SQL. Algumas das ações RDS são um conjunto de ações do RSS. Antes de operações complexas, o RDS coloca um ponto de salvamento. Se durante a operação acorrer uma falha no RDS ou RSS, o RDS volta ao ponto de salvamento do início da operação.

IV.1.1.5 - Diário e o Protocolo "DO-UNDO-REDO"

Não é possível criar um mecanismo de transação que suporte múltiplos usuários e pontos de salvamento baseado somente em arquivos sombras. Para isso, o mecanismo de sombra é combinado com um diário incremental de todas as ações que uma transação executa. O diário é usado pra desfazer e refazer transações sobre arquivos compartilhados e é usado em combinação com arquivos sombras para ponto de verificação e reinício ao sistema. Cada ação RSS de atualizar grava um registro diário com os valores antigos e novos do objeto atualizado. Estes registros são agregados

por transações e colecionados num arquivo diário comum do sistema.

Cada ação que modifica um objeto recuperável grava um registro no diário contendo informações suficientes (incluindo as valores antigo e novo do objeto atualizado) para que a operação possa ser mais tarde desfeita ou refeita completamente.

Quando um usuario define um arquivo sombreado ("shadowed"), ele informa se este arquivo usará diário ("logged") ou não. O RSS controla o salvamento e reconstrução de arquivos "shadowed" e "logged". O usuario controla o salvamento e reconstrução de arquivos "shadowed" e não "logged". Os arquivos não "shadowed" não podem usar diário, e seu salvamento e reconstrução são de inteira responsabilidade do usuário. Para isto ele deve fazer cópias do arquivo periodicamente.

Cada operação RSS num arquivo sombreado usando diário é implementada contendo as seguintes operações (figura IV.2):

a) DO, faz a ação solicitada e grava informações suficientes no diário para desfazer ou refazer a ação;

b) UNDO, desfaz a ação feita no DO usando o diário;

c) REDO, refaz a operação solicitada usando o diário.

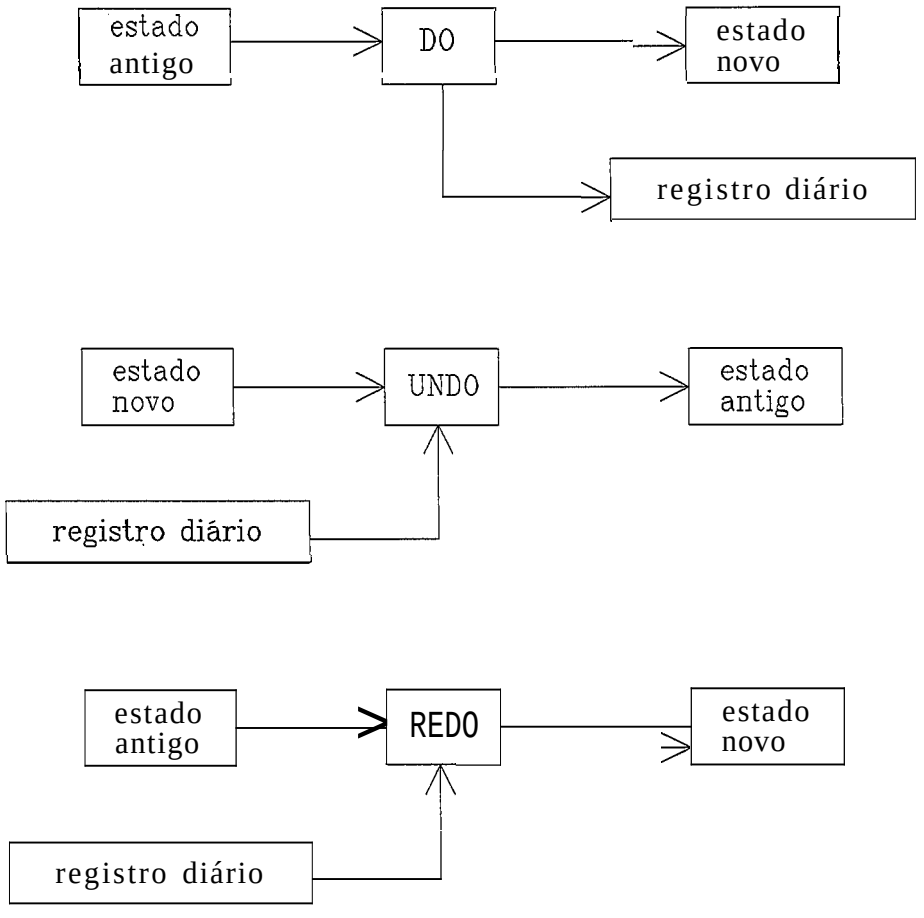


Figura IU.2 - Três aspectos de uma ação (DO, UNDO e REDO).

Como podemos observar todas as operações RSS sobre estes arquivos são recuperáveis.

O Sistema R garante que transações não compromissadas podem ser desfeitas e que transações compromissadas podem ser refeitas da seguinte maneira:

a) A transação é gravada no diário em disco antes quê sombras sejam substituídas por estados correntes do banco de dados.

b) A ação de compromissar a transação grava um registro de fim no "buffer" do diário e força todos os registros de transações do diário para disco (com o registro de comprometimento sendo o último destes registros).

Para que o efeito de qualquer transação não compromissada possa ser desfeito, é lido o registro no diário, é verificada a ação sue foi feita e é chamada a operação de UNDO desta ação, passando-lhe o registro do diário. Para desfazer uma transação, pode-se ir só até o primeiro ponto de salvamento. Para reconstruir um banco de dados até um ponto de salvamento, o gerenciador de reconstrução necessita de mais informações (o nome e o estado de todos os cursores ativos e todos as bloqueios existentes).

O diário é uma sequência de páginas, cada uma com um numero sequencial. Uma parte do diário é mantida "on-line" (necessária para UNDO e REDO de transações em execução) num anel de "buffers", a outra parte é mantida "off-line".

Se e anel de "buffers" está cheio, ou é porque está na hora de passar os "buffers" para disco ou fita, ou deve ser gravado um ponto de verificação, ou existe uma transação em execução há muito tempo (possivelmente esquecida pelo usuário), não liberando os "buffers" utilizados, e que por isto deve ser cancelada.

O tamanho de um diário pode ser modificado para uma sessão pelo administador, através de uma função que usa o banco de dados de forma exclusiva.

IV.1.1.6 - Cópias do Banco de Dados

Cópias do banco de dados podem ser feitas periodicamente para fita, através de um utilitário que não necessita de uso exclusivo do banca de dados. Pode-se copiar determinados arquivos ou todas eles.

A sessão de uso do banco de dados pode ser inicialirada de forma que seja feita uma copia do banco de dados

automaticamente quando o conteúdo diário atingir um determinado ponto.

IU.1.1.7 - Pontos de Verificação

Limita a quantidade de trabalho necessário (UNDO e REDO) no reinicio do sistema. Se são tomados com frequência tornam o reinicio mais rápido, porém causam uma sobrecarga maior no sistema. Isto deve ser balanceado contra o custo de fazer pontos de verificação frequentes.

A forma mais simples de ponto de verificação é gravar um estado consistente da transação pela inatividade do sistema, postergando todas as novas transações até que todas as transações em execução sejam completadas, gravando assim, um momento consistente logicamente. Isto aumenta a quantidade de trabalho perdido no reinicio que deve ser refeito. Inatividade do sistema, consequentemente, não é a maneira mais barata de obter um estado consistente da transação.

O RSS implementa pontos de verificação, sue são instântaneos do sistema, no momento quando não existem ações RSS em execução.

Pontos de verificação são tomados depois de determinada atividade do diário ou por pedido do operador do sistema. Quando ocorre um ponto de verificação e feito o seguinte:

a) é gravado um registro de ponto de verificacão no diário contendo uma lista de todas as transações em execução e ponteiros para o mais recente registro do diário.

b) depois que os registros do diário estão em disco, todos os arquivos do tipo "logged" são salvos (estado corrente substitui sombras), envolvendo o transporte dos

"buffers" do banco de dados e os diretórios dos arquivos sombras para memória secundaria.

c) como ultimo passo, o endereço do diário do registro do ponto de verificação é gravado como parte do registro do diretório na versão sombra do estado.

A raiz do diretório é duplicada em disco, permitindo o processo de ponto de verificação tolerar falhas enquanto grava a raiz do diretório. No reinicio o sistema sera capaz de localizar o registro de ponto de verificação correspondente examinando a raiz do diretório mais recente.

IV.1.1.8 - Falhas

Ao ser feito um reinício do sistema após uma falha o código do Sistema R é carregado e o gerenciador de arquivos restaura qualquer arquivo sombreado para suas versões sombras. Se um arquivo sombreado não foi salvo no encerramento, então a versão corrente sera substituída pela sua sombra. Todos os arquivos "logged" são levados ao estado em que estavam no último ponto de verificação. Para isto, é utilizada a parte "on-line" do diário.

Durante um reinício toda o banco de dados fica bloqueado. Para proteção contra falhas durante o reinício, o diário e as versões sombras não são alteradas enquanto o reinício não esta completo. A marca de fim de reinício com sucesso é um ponto de verificação. O ponto de verificação do sistema é atômico, tornando as alterações das versões sombras e do diário atômicas. Se ocorrer uma falha antes de completar o ponto de verificação, o processo de reinício retornará ao seu estado sombra original, se a falha ocorrer após o ponta de verificação estar completo, o banco de dados estará reconstruído.

é importante o uso de discos duplicados para proteção contra falha de meio de armazenamento. No caso de ocorrer uma falha de dispositivo ocorre o seguinte:

a) Se um dos diários duplicados falhar, um outro diário é alocado e a versão boa do diário duplicado é copiada rara a alocada.

b) Se qualquer outro arquivo falhar, a fita cópia mais recente que sobreviver e carregada para o banco de dados, r assim, todas as ações de atualizações compromissadas subsequentes ao registro do ponto de verificação são refeitas usando o diário.

Recuperação a partir de uma cópia para o gerenciador de recuperação é o mesmo que um reinício de um ponto de verificação muito antigo.

IV.1.1.9 - Recuperação e Bloqueio

Recuperação tem implicações e fornece requisitos sobre o subsistema de bloqueio.

Para que ações de uma transação não sejam desfeitas cegamente sem o desaparecimento de atualizações subsequentes feitas por outras transações, e essencial que as transações bloqueiem todas as atualizações de modo exclusivo e conservem esses bloqueios até que a transação seja compromissada ou desfeita.

Uma segunda questão é que no desfazer de uma transação não pode acontecer um "dêadlock". O UNDO deve tirar vantagem do fato que ele somente acessa registros que a transação bloqueou durante o DO, assim, não precisa pedir novamente o bloqueio (uma consequência da manutenção do bloqueio até o fim da transação). Porem, uma transação desfeita deve adquirir alguns bloqueios pelo fato de outras

transações RSS estarem executando e porquê ações RSS liberam alguns bloqueias no final de cada ação RSS (ex: bloqueios de páginas físicas quando bloqueio de registro lógico é a atual granularidade). Para resolver o problema, transações no processo de UNDO são marcadas como "douradas". Uma transação dourada nunca é escolhida como vítima de "deadlock", toda vez que estiverem envolvidas num "deadlock" com outra transação, a outra será escolhida como vitima. Para que esta regra não leve a um ciclo de "deadlock" inquebrável, uma regra adicional é adotada: somente uma transação dourada pode executar por vez. Um bloqueio especial (RSSBACKUP) é pedido no modo exclusivo rara cada transação dourada antes dela iniciar cada passo de UNDO. Este bloqueio é liberado no final de cada passo de UNDO.

Durante o reinício, bloqueios são desligados. O banco de Dados como um todo é bloqueado durante o processo de reinício que executa ações sequencialmente na ordem em que aparecem no diário.

fV.l.l.10 - Custo do Sistema de Reconstrução

Escrever ações recuperáveis (aquelas que podem se desfazer ou refazer por si mesmo) e um trabalho difícil. Segundo os autores em (GRAY 1981b), escrever estes tipo de ação é 30% mais difícil, e requer perto de 20% mais de código que escrever ações não recuperaveiç. O sistema de reconstrução sozinho (gerenciamento do diário, reinício do sistema, ponto de verificação, comprometimento, aborto, etc.) contribui com algo em torno de 15% do código do RSS, que é menos da metade do Sistema R, levando o custo de reconstrução para perto de 10% do custo total do sistema.

Cada transação compromissada adiciona dois I/Os ao custo da transação quando o diário duplicado é forçado para disco (como parte da ação de comprometimento), quando não é

duplicado acrescenta um I/O. O custo de I/O torna-se alto se o banco de dados for grande e usar o mecanismo de página sombra.

O diário em disco sai mais caro que em fita, porém, faz com que o reinício seja mais rápido e independente de operador.

Para Gray, o mecanismo de sombras não e aconselhável para grandes arquivos pela quantidade de espaço ocupado. O sistema durante operações sobre um arquivo, mantém duplicidade das informações para reconstrução, no diário e nas páginas sombras. No diretório do arquivo, deve-se ter uma entrada para cada página do arquivo, e quando o arquivo é alterado é necessário que se tenha dois diretórios simultaneamente (o atual e o sombra). Alem disso, é necessária uma área para manter as páginas duplicadas. No disco, as páginas não se encontram em ordem sequencial, o que torna uma consulta sequencial muito cara. Porem, o uso do mecanismo de sombras tem um bom desempenho em bancos de dados com menos de 10 megabytes, sendo uma boa técnica de reconstrução para arquivos privados (GRAY 1981b).

IU.1.2 - INGRES

INGRES (Interactive Graphics and Retrieval System) em (STONEBRAKER 1976) e (STONEBRAKER 1980), é um sistema de banco de dadoç relaciona1 implementado em cima do sistema operacional UNIX desenvolvido na Universidade da California em Berkley. Utiliza as liguagens QUEL ("Query Language"), baseada em calculo relacional, e EQUEL ("Embedded Quel"), composta pela linguagem C com comandos QUEL embutidos.

IV.I.2.1 - A Estrutura de Processos do INGRES

Um processo no UNIX é um espaço de endereçamento (o tamanho varia de 64 K até 128 K conforme a maquina) o qual é associado com um "user-id" e a uma unidade de trabalho que o gerenciador UNIX manipula. Um processo pode ter subprocessos, consequentemente um processo pai pode ser a raiz de uma subárvore de processos. Um processo pode pedir que o UNIX execute um arquivo em um processo descendente. Os processos se comunicam usando uma facilidade de comunicação entre processos chamada de "pipe", que são uma forma de comunicação numa só direção.

Quando um comando UNIX chama o PNGRES, é criada uma estrutura com quatro processos (figura IV.3). Esta arquitetura modular foi uma forma engenhosa de contornar a limitação imposta pelo computador **PDP-11/78** no qual o sistema foi desenvolvido. Nas versões comerciais mais recentes do INGRES esta arquitetura modular foi substituída por uma arquitetura monolítica mais eficiente.

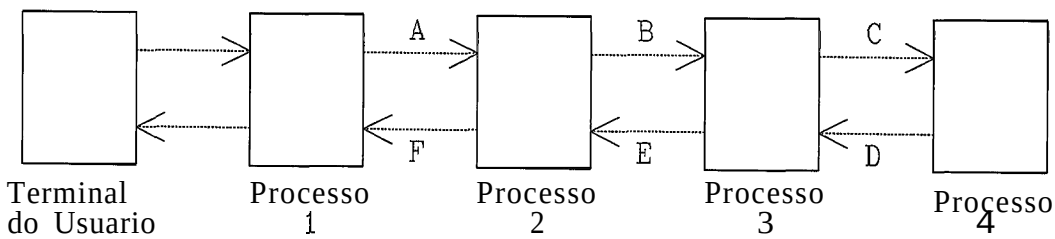


Figura IV.3 - A estrutura de processos do INGRES.

O processo 1 é um monitor interativo de terminal e permite ao usuário formular, imprimir, editar e executar conjuntos de comandos INGRES. Ele mantém uma área de trabalho com a qual o usuário interage. O conteúdo da área de trabalho é passado para o processo 2 ("pipe" A) como um conjunto de caracteres ASCII quando a execução é desejada.

O processo 2 contém um analisador léxico, um analisador sintático, rotinas de modificação de consultas para controle de integridade, e controle de concorrência. Quando

o processo 2 termina ele passa o "string" de "tokens" resultante para o processo 3 através do "pipe" B.

O processo 3 aceita a "string" de "tokens" e contem rotinas de execução para os comandos RETRIEVE, REPLACE, DELETE e APPEND. Todos os comandos de alteração do banco de dados são reformulados para um comando RETRIEVE com o objetivo de isolar as tuplas a serem alteradas. Todas as tuplas a serem alteradas são copiadas para um arquivo especial que é enviado para o processo 4.

O processo 4 possui um processador de atualização postergada, utilitarios, rotinas de alteração do banco de dados, rotinas para criar e destruir arquivos, etc.

Mensagens de erros são enviadas de volta ao processo 1 através dos "pipes" D, E e F, que por sua vez retorna ao usuário. Se o comando for um RETRIEVE com nenhum resultado de relação especificado, o processo 3 retorna as tuplas qualificadas num formato estilizado diretamente para o dispositivo de saída padrão do processo 1.

Se o INGRES for chamado pelo pré-compilador EQUEL, é criada a mesma estrutura de processos já descrita. O programa EQUEL é convertido num programa C com instruções QUEL convertidas num código C apropriado e chama o INGRES. O programa C resultante é então compilado pelo compilador C normal, produzindo um módulo executável. Além disso, quando um programa EQUEL está executando, o módulo executável produzido pelo compilador C é usado como o processo "front-end" no lugar do monitor interativo de terminal (figura IV.4).

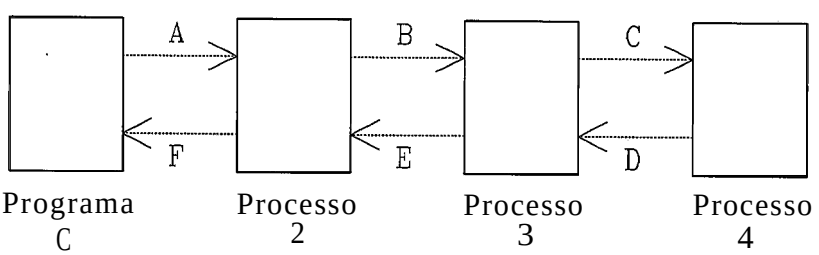


Figura IV.4 - A estrutura de processos com EQUEL.

IV.l.2.2- Arquivos

O sistema INGRES pode ser chamado de duas maneiras: Primeiro, ele pode ser diretamente chamado pelo UNIX através da execução do comando INGRES nome-do-banco-de-dados; segundo, ele pode ser chamado através da execucão de um programa escrito usando o pré-compilador EQUEL.

O UNIX suporta um sistema de arquivo estruturado em árvore. Cada arquivo é um diretório (contendo referências para arquivos descendentes no sistema de arquivo) ou um arquivo de dados (representações tabulares das relações do banco de dados). Cada arquivo é fisicamente dividido em blocos de 512 bytes (páginas). Em resposta a um pedido de leitura, o UNIX move uma ou mais páginas da memória secundaria para o conjunto de "buffers" do UNIX e então responde ao usuário a cadeia de bytes desejada. Se a mesma página for referenciada novamente (pelo mesmo ou por outro usuário) enquanto permanece no conjunto de "buffers", não haverá operações de I/O.

O conjunto de "buffers" do sistema UNIX obedece ao algorítmo de substituição LRU ("least recently used"). Assim, todo o arquivo do sistema e manuseado como um grande armarenamento virtual.

Os projetistas do INGRES acreditam que um sistema de banco de dados deve aparecer como um programa do UNIX. De outro modo, o sistema poderia operar fora do padrão UNIX e se tornar menos portável. Além disso, os projetistas acreditam que o UNIX deve gerenciar os "buffers" do sistema para o conjunto de programas em execucão. Em consequência disso, o INGRES não contem facilidades para fazer seu próprio gerenciamento de memória.

IV.1.2.3 - Transação

No projeto do INGRES existiam cinco escolhas básicas disponíveis para definição de transação como unidade atômica de trabalho:

a) alguma coisa menor que um comando INGRES;
b) um comando INGRES;
c) uma coleção de comandos INGRES sem intervenção de código C;
d) Uma coleção de comandos INGRES com código C mas sem chamadas do siçtema;
e) um programa EQUEL arbitrário.

Se a opção a) fosse escolhida, INGRES não poderia garantir que dois comandos de atualização executando concorrentemente tivessem o mesmo resultado como se fossem executados sequencialmente numa coleção de processos INGRES.

A opção c) poderia ser obtida com uma simples estensão dos algoritmos e poderia ser implementada se assim fosse necessário.

Para suportar a opção d) a complexidade do sistema aumentaria muito para o que é percebido pelos projetistas como sendo uma pequena generalização.

A opção e) é na opinião dos projetistas do INGRES como sendo impossível de ser suportada.

Assim sendo, os projetistas do INGRES escolheram a opção b), um comando INGRES como transação.

IV.1.2.4 - Diário

O processo 3 transforma todas as operações de atualização (REPLACE, UPDATE e DELETE) em um comando **RETRIEVE** para isolar as tuplas a serem modificadas e as grava em um arquivo temporário. Este arquivo e passado para o processo 4 para que este execute as alterações solicitadas e altere as índices secundários que forem necessários. O conjunto de arquivos temporários forma um arquivo diário, que contém informações sobre todas as transações executadas.

IV.I.2.5 - Cópias do Banco de Dados

O administrador do banco de dadas chamado no **INGRES** de "superuser", pode copiar o banco de dados para fita através do esquema de "backup" do UNIX.

IV.1.2.6 - Pontos de Verificação

No INGRES o administrador do banco de dados ("supersuser") é o único que pode fazer um ponto de verificação. Nesta operação, ele copia o banco de dados para fita, r quando for necessário voltar ao último ponto de verificação do sistema, ele volta ao estado em que o banco de dados estava quando foi tirada sua última cópia.

IU.1.2.7 - Falhas

No INGRCS dificilmente ocorrerão falhas de transação porque os processos 2 e 3 verificam a existência de problemas que podem causar este tipo de falha, não permitindo, neste caso, que seja criado o arquivo temporário. Se um usuário quiser cancelar uma transação,

ele simplesmente aperta uma tecla específica. Se o arquivo temporário ainda não começou a ser executado pelo processo 4, ele será destruido. Se o arquivo já começou a ser executado, deve ser totalmente reexecutado para verificar se foram feitas todas as alterações e carrigí-las se necessário.

Quando ocorre alguma falha do sistema que exija reconstrução, é acionado o comando RESTORE. Só o administrador do banco de dados pode utilizar este comando. 0 arquivo temporário de alterações será tratado como no caso do usuário interrompeu a execução de uma transação.

Se após uma falha do sistema o catálogo se encontrar inconsistente, é usado um algorítmo que percorre o catálogo procurando as inconsitências e tomando ações apropriadas rara restaurá-lo.

Em caso de falhas de dispositivos, o administrador do banco de dados possui uma cópia do banco de dados em fita e o arquivo temporário (IV.1.2.4) do INGRES com as alterações feitas, podendo obter com isso um estado consistente do banco de dados.

IV.1.2.8- Controle de Concorrência

Existe no processo 2 um módulo encarregado da gerência dos bloqueios das transações que bloqueia as relações fisicamente. 0 processo 3 não pode iniciar sua execução sem sue antes todos os recursos necessários estejam bloqueados. Com isso, "deadlocks" são evitados. Existe uma lista de bloqueios na qual é inserida uma tupla para cada bloqueio solicitado. Toda a lista e bloqueada fisicamente e e verificado se existe conflito entre os bloqueios. Se existir, o processador de concorrência espera um tempo determinado e então tenta bloquear a relação novamente. Se não existir, todos os bloqueios já estão inseridos. Os

bloqueios são matidos pelo processador de concorrência ate este receber uma mensagem de fim atraves do "pipe" E.

IV.2 - Banco de Dados Orientados a Objetos e seus Sub-Sistemas de Reconstrucão.

Nesta seção, antes de descrevermos as três propostas de sistemas de banco de dados orientados a objetos (SBDOOs), achamos que seria importante fazer uma breve introdução sobre essa área emergente para que as pessoas que ainda não tiveram oportunidade de estudar o assunto possam se localizar no contexto. Assim, primeiro abordamos o paradigma da orientação a objetos em sistemas de banco de dados e depois descrevemos três propostas de sistemas de banco de dados orientados a objetos.

É importante salientarmos sue o material encontrado sobre as propostas dos SBDOOs se preocupam muito mais com as características dos modelos do que com detalhes de implementação. Assim sendo, o estuda sobre gerência de transação, controle de concorrência e reconstrução é bastante limitado. Porém, pode-se compor uma idéia de como esses assuntos estão sendo tratados nesses tipos de sistemas.

IV.2.1 - Sistemas de Banco de Dados Orientados a Objetos

Segundo Dittrich em (DITTRICH 1986), Sistemas de Banca de Dados Orientados a Objetos (SBDOOs) devem fornecer um modelo de dados capaz de representar uma entidade do mundo real, exatamente através de um objeto no banco de dados, além de permitir a definição de operadores apropriados para explorar a sua estrutura. Isto deve ser independente da complexidade e da estrutura da entidade na munda real.

SBDOOs devem fornecer acesso compartilhado aos dados num ambiente multiusuário, incorporando todas as facilidades usuais de banco de dados como controle de concorrência, recuperação, autorização e funções de gerência do sistema.

Segundo Heiler em (HEILER 1987), a aplicação do paradigma da orientação a objetos a sistemas de banco de dados oferece vários benefícios, entre eles temos:

a) Em um banco de dados orientado a objetos é utilizado um único modelo de representação. Tanto o mapeamento entre o modelo mental do usuário e o modelo externo do sistema, quanto o mapeamento entre os modelos externo e interno do próprio sistema são simplificados através do modelo comum.

b) Nos bancos de dados orientados a objetos encontra-se normalmente uma interface uniforme para os objetos e as operações disponíveis em cada nível de detalhe. Com isto é possível aumentar-se a capacidade de troca e de compartilhamento de informações entre usuários. Fica também mais fácil a interacão entre diferentes ferramentas.

c) Em um banco de dados orientado a objetos a representação e implementação dos objetos é ocultada através de um mecanismo de abstração que fornece uma visão externa destes e de suas operações. No caso particular dos objetos complexos, normalmente e fornecida uma visão do objeto em termos de suas caracterksticas agregadas e ocultada a sua estrutura interna em termos de seus componentes. Assim, objetos complexos são representados e manipulados de forma mais natural aos usuarios (manipulação direta do objeto).

d) A integração entre o programa de aplicação e o sistema de gerência de banco de dados durante a manipulação de objetos complexos também e simplificado, o envio de uma mensagem é suficiente para executar o equivalente a um

conjunto de consultas já que, como consequência desta mensagem, outras podem ser automaticamente disparadas e executadas.

e) A habilidade de se descrever objetos em múltiplos níveis de abstração permite a decomposição de um problema em sub-problemas independentes e menores, nos quais podem ser suprimidos detalhes de níveis inferiores e de implementação. Isto tende a facilitar a modelagem dos objetos e a simplificar a sua manutenção no banco de dados.

$1 A possibilidade de se definir as características dos tipos de objetos e a heranca destas definições pelos seus subtipos leva a uma economia de especificação, reduz a complexidade do banco de dados e garante a sua Integridade.

g) Outra importante aquisição obtida com a incorporação do paradigma da orientação a objetos aos sistemas de banco de dados é a flexibilidade, que pode ser percebida em funcão da capacidade de:

g.1) adição de novas classes de objetos e operações;

g.2) modificação das representações físicas dos objetos sem causar impactos nas operações sue os acessam (independência física);

g.32 modificação da implementação das operações sem afetar as aplicações que as utilizam.

Ainda segundo Heiler em (HEILER 1987), algumas características do paradigma da orientação a objetos devem ser respeitadas num sistema gerenciador de objetos, entre elas podemos citar:

a) Objetos existem e podem ser identificados univocamente independente de suas representações ou qualquer de seus atributos.

b) Classes de objetos encapsulam a definição do objeto com suas propriedades, restrições e operações. Conceitos como hierarquia de generalização e herança de tipos provêem semântica adicional para a organização de classes de objetos. Somente operações definidas para o tipo podem ser aplicadas para instâncias do tipo.

c) Objetos e operações sobre eles podem ser expressos independentemente da representação da objeto ou da implementação da operação.

d) Todas as operações podem ser invocadas de uma maneira uniforme, independente da operação ou do objeto no qual a operação esta sendo aplicada. Passagem de mensagens tem sido vista como parte integrante da abordagem orientada a objeto, mas os mecanismos de chamadas de procedimentos de linguagem de programação tradicionais também podem ser usados.

IU.2.2 - Descrição de Três Propostas de Sistemas de Banco de Dados Orientados a Objetos e de seus Sub-Sistemas de Reconstrução.

Nesta seção são apresentadas as principais características de três propostas de sistemas de bancos de dados orientadas a objetos: CACTIS, GemStone e POSTGRES. A escolha dessas três propostas foi proposital, pois cada uma delas é um exemplo de uma estratégia diferente para implementação de um SBDOO. O CACTIS e proveniente de um modelo semântico, o GemStone é uma extensão de uma linguagem de programação orientada a objetos e o POSTGRES é uma extensão de um sistema de banco de dados relacional.

A descrição das três propostas foram adaptadas dos trabalhos de Blum, Goncalves e Rossato (BLUM 1988) e de Gonçalves e Mattoso (GONÇALVES 1988), que são frutos das cadeiras de Tópicos Especiais em Ambientes de Desenvolvimento de Software e de Banco de Dados Não Covencionais realizadas ao longo de 1988 na COPPE/Sistemas.

IV.2.2.1- CACTIS

O CACTIS (HUDSON 1986, HUDSON 1987, HUDÇON 1988 e KING 1987), Coi desenvolvido na Universidade do Colorado, e seu modelo e visto como uma coleção de objetos da mesma forma que o modelo do SmallTalk. Possui um modelo semântico de dados onde relacionamentos são usados para construir agregações ou objetos complexos. Pode ser utilizado para aplicações de Engenharia de Software em estações de trabalho Sun. O sistema opera, atualmente, com um armazenamento centralirado de dados. Entretanto, são suportados acessos concorrentes por múltiplos usuários.

IU.2.2.1.1- Modelo de Dados

Um banco de dados CACTIS é visto como uma coleção de objetos, de forma semelhante ao SmallTalk. Cada objeto tem a si associado um grupo de valores de dados, chamados atributos, que são determinados pelo seu tipo. O tipo do objeto pode ser estabelecido estática ou dinamicamente.

CACTIS possui um modelo semântico de dados onde relacionamentos são usados para construir agregações ou objetos complexos. Para formar objetos de níveis mais altos, ou objetos abstratos, os objetos do banco de dados podem ser conectados recursivamente por tipificação ou por relacionamentos diretos. Segundo Hudson em (HUDSON 1988) o modelo de dados do CACTIS e um mecanismo natural para

especificar o conjunto de dados necessários a um ambiente de engenharia de software.

IV.2.2.1.2 - Linguagem de Definição e Manipulacão de Dados

A linguagem de dados do CACTIS permite que funções de derivação sejam construídas usando todos os operadores aritmeticos e lógicos da linguagem C. Podem também ser usados diversos construtores condicionais e interativos.

Uma linguagem formal de manipulação de dados não foi ainda apresentada. CACTIS provê uma serie de primitivas de manipulação de dadas que podem ser usadas para construir uma 'linguagem. Estas primitivas incluem:

a) criação e remoção de instâncias de dados;

b) estabelecimento e quebra de relacionamentos entre instâncias;

c) recuperação e movimentação de valores de atributos

IV.2.2.1.3- Gerência de Dados

Em CACTIS, sempre que um bloco do disco é lido para a memória, todos os processos que são associados com alguma instância armazenada no bloco são promovidos a uma fila especial de alta prioridade. Um esquema "hashing" e usado para indexar os processos pendentes pelas instâncias que contêm um atributo associado a eles.

Quando novos processos são escalonados, primeiro e verificado se a instância associada ao processo já esta na memória. Se já estiver, ele e escalonado na fila de alta prioridade. Desde que possam ser executados sem acesso adicional a disco, processos na fila de alta prioridade têm sempre prioridade sobre outros processos.

Para promover a localização de referências de dados, estes são mantidos em "clusters", que são baseados em padrões de utilização. Um contador é mantido com o número total de veres que cada instância é acessada no banco de dados, r com o número de vezes em que se cruza um relacionamento entre instâncias no processa de avaliação de atributos. O banco de dados é reorganizado periodicamente baseado nestas informações.

Uma vez que todos os processos na memória tenham sido executados, é preciso que se escolha o próximo a ser executado. Os processos escolhidos são aqueles que causam o menor número de acessos a disco. Para tanto, são mantidas informações históricas sobre o comportamento passado. Assim, o banco de dados se torna auto-adaptável, permitindo que mudanças na sua estrutura sejam o reflexo das medias das mudanças escalonadas

IV.2.2.1.4- Gerência de Transações, Controle de Concorrência e Recuperação

Devido à natureza complexa dos dados encontrados neste banco de dados, o sistema de banco de dados deve suportar formas não usuais de transações, além de suportar transações longas e complexas.

As transações longas precisam de mecanismos de restrição mais flexíveis e poderosos. Restrições podem estar amarradas à finalização de uma transação ou ao "checkpoint" de um grupo de módulos. Para conseguir controle e precisão sobre as transações longas e interativas, restrições podem ser forçadas ou validadas continuamente.

O banco de dados CACTIS é projetado para ser usado de uma maneira exploratória (HUDSON 1986), assim, ele precisa de mecanismos de retorno e recuperação.

O ponto chave das funções de retorno e recuperação no CACTXS é sue todas as mudancas para dados derivados ocorrem como resultado do processo de avaliação automática de atributos. Este processo é invocado sempre que atributos intrínsicos são alterados. O valor antigo de um atributo pode ser recuperado a fim de se reverter os efeitos de uma alteração.

O mesmo processo de avaliação de atributos e usado para informações deduzidas. Pode-se desfazer as deduções quando recuperação e retorno são realizadas. Por isso, o sistema mantém o valor antigo de qualquer atributo alterado.

Retorno e recuperação usam os mesmos mecanismos do processo de atualização. O sistema de banco de dados precisa apenas manter uma série de valores antigos e a localização das mudancas, com um conjunto de marcadores que delimitem as fronteiras da transação. Quando uma transação falha e o usuário explicitamente requisita uma operação de "undo", o sistema restaura o estado do banco de dados ao seu estado anterior a partir dos valores antigos que foram salvos.

O controle de concorrência no CACTIS é feito através da técnica de "time-stamping".

IV.2.2.2- GEMSTONE

O GEMSTONE (COPELAND 2984, MAIER 1985, MAIER 1986a, MAIER 1986b, PURDY 1987, STEIN 1987) foi desenvolvido no Oregon Graduate Center. É um sistema que encapsula uma variedade de idéias das áreas de representação de conhecimento orientada a objetos, programação não procedural e modelagem de dados temporais. Possui uma linguagem chamada OPAL baseada na SmallTalk, e três tipos de interfaces. Pode ser utilizado a partir de computadores do tipo IBM PC conectados em rede a um DEC VAX.

IU.2.2.2.1- Modelo de Dados

O modelo de dados do Gemstone baseia-se no modelo de objetos do SmallTalk, em lugar de ter como base um dos tradicionais modelos orientados a registros (COPELAND 1984).

O modelo de dados Gemstone (ESDM) permite que as aplicações gerenciem informações não tradicionais como documentos, figuras e sons. O modelo foi projetado para permitir modelar diretamente objetos complexos e relacionamentos.

O modelo captura explicitamente a história dos estados do banco de dados como parte do modelo de dados, e suporta consultas que acessem estados passados. Isto e obtido através da não remoção de objetos do banco de dados, sendo estes ali mantidos de forma histórica.

Os três principais conceitos do modelo de objetos são: objeto, mensagem e classe.

Um objeto é uma memória privativa que possui uma interface pública. Esta memória é estruturada como uma lista de variáveis de instância cujos valores são referências a outros objetos.

Objetos comunicam-se entre si através da passagem de mensagens. O conjunto de mensagens que um objeto entende e consegue respondeu descreve o seu comportamento, e é chamado de protocolo. Cada objeto responde a uma mensagem executando um método escrito na linguagem OPAL.

Os objetos que compartilham os mesmos formatos e métodos são grupados numa classe e chamados de instâncias da classe. Classes são arrumadas segundo uma hierarquia, onde

cada subclasse herda a estrutura e o comportamento de suas superclasses. Quando uma mensagem é recebida por um objeto este consulta sua classe e suas superclasses para localizar o método adequado para a execução. Também é possível representar-se objetos compostos sue são formados pela agregação de outros objetos.

O modelo e ainda capaz de capturar diretamente relacionamentos do tipo muitos-para-muitos (N:M), coleções e sequências.

IV.2.2.2.2 - Linguagem de Definição e Manipulação de Dados

A linguagem utilizada no Gemstone e chamada OPAL. Sua sintaxe e semântica são semelhantes as da linguagem SmallTalk (PURDY 1987).

Segunda Maier em (MAIER 1985), a escolha de uma linguagem com base no SmallTalk deve-se muito ao fato deste possuir uma base de literatura e divulgação já estabelecida, reduzindo assim o seu custo de desenvolvimento e divulgação.

OPAL é uma linguagem de banco de dados orientada a objetos que é usada para definição e manipulação de dados, e para computação em geral. Oferece facilidades para criação de tipos abstratos de dados flexíveis e para encapsulamento de dados e operações através de mensagens.

Foram adicionados a linguagem OPAL classes e métodos primitivos para fornecer controle de transações, sugestões de armazenamento e requisições de replicação de dados.

IV.2.2.2.3- Gerência de Dados

Todos os objetos no sistema residem num espaço de objetos em disco que é dividido em repositórios. Um repositório representa uma partição do espaço de objetos e é implementado, sobre o sistema operacional considerado, como um arquivo de acesso direto em disco.

Repositórios são divididos em regiões disjuntas chamadas de segmentos, com propósitos de controle de autorização e de concorrência.

O gerenciador de objetos GOS ("GEMSTONE Object Server"), envolve vários Gem Process e um Stone Process (MAIER 1986a). O Stone corresponde à memória de objetos e o Gem à máquina virtual. Stone fornece:

a) gerência de armazenamento secundário;

b) autorização e controle de concorrência;

c) suporte a transações e recuperação;

d) suporte a acesso associativo;

e) gerência das estações de trabalho para sessões ativas.

IV.2.2.2.4 - Gerência de Transações, Controle de Concorrência e Recuperação

O controle de transações em GemStone usa uma abordagem na qual é dada prioridade às transações de leitura sobre as transações de leitura/gravação (MAIER 1985). O sistema supõe que as transações de leitura sejam mais frequentes do que as de leitura/gravação.

Os repositórios podem ser replicados no disco para evitar falhas. A replicação foi escolhida, em detrimento

dos tradicionais arquivos diário de transações, por ser mais viável para implementar dentre das restrições do gerenciador de armazenamento do GemStone.

Como os dados de um banco de dados GemStone são compartilhados por múltiplos usuários, o sistema deve possibilitar acessos concorrentes (MAIER 1986a). Cada usuário deve poder ver uma versão consistente do banco de dados, mesmo que outros usuários o estejam utilizando simultaneamente.

Usuários podem fazer modificações no banco de dados sue não devem permanecer neste. Para tanto, GemStone suporta múltiplas estações de trabalho, nas quais as modificações propostas para o banco de dados podem ser então eliminadas ou a este agregadas. O sistema deve portanto gerenciar os vários espaços.

A granularidade do controle de concorrência é a nível de segmentas. Isto é justificado pelo fato de que o controle sobre objetos individuais poderia prejudicar o desempenho do sistema.

Stone fornece a cada sessão de usuário uma área de trabalho que contem uma cópia da tabela de objetos, derivada da última versão da tabela de objetos chamada "shared table" (MAIER 1986a). Quando uma sessão modifica um objeto, uma nova cópia daquele objeto e criada, ficando Inacessível para outras sessões. A cópia da tabela de objetos é então atualizada.

Foi escolhido um esquema de controle de concorrência no qual conflitos de acesso são verificados em tempo de "commit", prevenindo-se assim a ocorrência de "deadlocks".

Para cada transação, em tempo de "commit", Stone verifica conflitos com transações de leitura/gravação e gravação/gravação desde o começo da transacão. Se não

existir qualquer conflito, é permitido à transação efetuar as alterações desejadas.

Com o esquema escolhido, certifica-se que transações de leitura nunca conflitem com outras transações. Assim, apenas as transações de gravação podem conflitar com outras.

Em GemStone, a recuperação de falhas não requer grandes mecanismos adicionais sobre os que existem para o controle de concorrência.

R unidade de recuperação é a transação. Mudanças feitas por transações compromissadas são mantidas, enquanto que mudanças ainda não compromissadas são perdidas.

Uma vez que versões compartilhadas do banco de dados nunca são regravadas, não é preciso muito esforço no sentido de levar o banco de dados para um estado consistente. A única preocupação que se deve ter e a ocorrência de falhas no processador enquanto uma nova raiz do banco de dados estiver sendo gravada.

Para suportar esta eventualidade, são mantidas duas cópias da raiz, que ficam residentes em um lugar específico. A restauração de um banco de dados a um estado consistente, após uma falha, consiste simplesmente da verificação destas duas páginas. Se estiverem diferentes, a que for determinada como não corrompida é copiada para cima da outra. O trabalho real de recuperação é a "coleta de lixo". Nesta são removidas os detritos das transações que não foram completadas até a ocorrência da falha.

Como precaução contra falhas dos dispositivos de armazenamento, foi introduzida uma estrutura chamada repositório. O repositório é a a unidade de armarenamento e também unidade de replicação. Partições dos segmentos dos repositórios e todos os objetos dos segmentos são armazenados no repositório do segmento.

Uma instância de repositório pode responder a uma mensagem do tipo "replicar" que significa que duas cópias do repositório devem ser mantidas, usualmente em diferentes dispositivos externos. As cópias têm conhecimento umas sobre as outras, e se uma falhar a outra permanece disponível.

IU.2.2.3 - POSTGREÇ

O POSTGRES (ROWE 1986, ROWE 1987, STONEBREAKER 1984, STONEBREAKER 1986a, STONEBREAKER 1986b e STONEBREAKER 1987) foi desenvolvido na Universidadey de Berkeley. é uma extensão orientada a objetos do sistema relaciona1 INGRES (seção IV.1.2). Possui uma linguagem de consulta chamada Postquel, e suporta objetos complexos através de tipos abstratos de dados e procedimentos. Suporta bancas de dados ativos através de gatilhos, alertas e inferências. A extensibilidade é permitida através de novos tipos de dados, operadores, métodos de acesso e um mecanismo de recuperação simplificado.

IU.2.2.3.1 - Modelo de **Dadas**

O modelo do POSTGRES suporta a gerência de objetos através de uma extensão do modelo relacional, oferecendo para tal duas facilidades: tipos abstratos de dados e procedimentos como tipos de dados (ROWE 1987).

O banco de dados é composto de relações que contém tuplas. Uma relação tem atributos de tipos fixos, que representam propriedades das entidades e dos relacionamentos, e uma chave primaria. A chave primária e uma sequência de atributos da relação que juntos identificam univocamente cada tupla. Tuplas precisam ter um valor para todas os atributos sue pertençam a sua chave.

Heranças de dados podem ser especificadas. Atributos podem ser definidos em cada relação ou herdados da definição de outras relacões.

Uma relação herda os atributos de seu pai a menos que estes sejam sobrepostos na sua definição. Podem ser herdados atributos de vários pais (herança múltipla).

Em POSTGRES são suportadas versões de relações. Atualizações em uma versão não modificam a relaçao envolvida. Atualizações na relação envolvida são visíveis através de suas versões desde que o valor seja também modificado nestas.

No modelo de dados do POSTGRES e provida uma coleção de tipos atômicos e estruturados. Há tipos pré-definidos e o usuário pode estender o sistema adicionando novos tipos. Para tanto podem ser usados tipos abstratos de dados e novos operadores sobre os novos tipos.

O modelo para heranca de procedimentos e quase idêntico à herança de métodos em linguagens de programação orientada a objetos. A heranca de procedimentos usa a hierarquia e as regras de heranca de dados. Cada relação tem a si associada uma lista de precedência de herança que é usada para resolver conflitos.

IV.2.2.3.2- Linguagem de Definição e Manipulação de Dados

A linguagem de consulta do Postgres é uma extensão da linguagem QUEL, chamada POSTQUEL (ROWE 1987), (STONEBREAKER 1986a), (STONEBREAKER 1986b).

Embora as estruturas básicas da linguagem POSTQUEL sejam bem similares às da QUEL, com as varias extensões feitas pretende-se suportar:

a) objetos complexos;

b) tipos de dados e métodos de acesso definidos pelo usuário;

c) dados temporais;

d) alertas, gatilhos e regras

POÇTQUEL provê também um conjunto de operadores de comparação e um construtor de relações que pode ser usado para especificar consultas complexas.

A linguagem POSTQUEL trabalha com três níveis de hierarquia de memória que são: memória principal, secundária (disco magnético) e terciaria (disco ótico). Dados correntes são armazenados na memória secundaria e dados históricos na memória terciaria. Entretanto, os dados são consultados pelos usuários sem que estes saibam onde estão armazenados.

POSTQUEL permite ainda atualizações de versões e a criação de uma versão a partir de uma relação ou de uma imagem instantânea.

IV.2.2.3.3- Gerência de Dados

Quando uma relação é criada, um arquivo é alocado para manter os registros da relação. Estes não precisam ter um tamanho limitado. O administrador de armazenamento é preparado para processar registros que vão além das fronteiras dos blocos de disco. Isso é feito por alocação contígua de registros e ligação por listas encadeadas.

Inicialmente, Postgres usa arquivos convencionais providos pelo sistema operacional UNIX. Entretanto, se o

espaço em arquivos terminar, Postgres estende o arquivo por páginas de tamanho múltiplo de 8 Kbytes.

Se o usuário desejar que os registros de uma relação sejam concentrados ("cluster") em um determinado campo, ele declara sua intenção indicando o campo apropriado.

O banco de dados é parcialmente armazenado em disco magnético e parcialmente em disco ótico. Os dados do disco magnético incluem todos os índices secundários e tuplas, utilizando o sistema de arquivo padrão do UNIX com uma relação pur arquivo recente (Última versão) do banco de dados. O disco dtico é reservado como um arquivo de armarenamento contendo tuplas históricas e é organizado como um grande repositório com tuplas de várias relações misturadas.

IU.2.2.3.4- Gerência de Transações, Controle de Concorrência e Recuperação

O sistema Postgres suporta quatro modos de atualização: cópia local, atualização direta, atualização deferida e atualização de objeto.

Nu modo de cópia local uma cópia do objeto é feita em memória cachê. Atualizações para o objeto não são propagadas para o banco de dadus e atualizações por outros processos em outras cópias não são propagadas para a cópia local. Neste modo de atualização, mudanças são válidas somente para a sessão corrente.

No modo de atualização direta, cada atualiracão para o objeto é propagada imediatamente para o banco de dados. Em outras palavras, atualizando uma variável de instância num objeto causa a execução de uma operação de atualização sobre a relação que representa instâncias do objeto. Para estas atualizações é usado um modelo convencional de

controle de transações num banco de dados. Bloqueios para gravação são aceitos quando consultas de atualização são executadas e são tirados quando estas são encerradas.

No modo de atualização deferida salva-se as atualizações de objetos antes que o sistema, em tempo de execução, peca explicitamente que estas sejam propagados para o banco de dados. Um modelo de transações convencional é usado para especificar limites de atualizações. Uma operação de inicio de transação pode ser executada por um objeto específica. Subsequentes acessos a variáveis indicarão bloqueios apropriados de leitura e gravação, para assegurar atomicidade e recuperabilidade de transações. Uma transação é compromissada quando uma operação de fim de transação é executada sobre o objeto.

O Último modo de atualização suportado pelo sistema e a atualização de objeto. Este modo trata todos os acessos para o objeto como uma transação simples. Um bloqueio de "intenção-de-gravação" é aceito sobre um objeto quando ele é recuperado a primeira vez do banco de dados. Outros processos podem ler o objeto, mas não podem atualizá-lo.

Neste modo de atualização, atualizações de objetos são propagadas para o banca de dados quando o objeto e liberado da memória cachê. Assim, reduz-se o nível de concorrência porque todo objeto é bloqueado enquanto estiver na memória cachê de objetos.

Quando um processo recupera um objeto, um "alertador" do banco de dados é estabelecido sobre o objeto. Este "alertador" notificara ao processo quando o objeto for atualizado par outro processo.

Quando o "alertador" é disparado par outro processa, a processo que estabeleceu o "alertador" é notificado. O valor retornado pelo "alertador" para o processo que o estabeleceu é o valor atualizado do objeto.

O Postgres usa um esquema de controle de concorrência baseado na ordenação de selos temporais ("time-stamping") e uma política de bloqueio de duas fases que é implementado na tabela de bloqueios da memória principal.

Postgres contém uma relação TIME na qual é guardado o instante em que cada transação foi encerrada. Cada relação em um banco de dados Postgres é "etiquetada" (identificada) no momento que é criada.

Existe um problema na implementação de hierarquias de objetos em sistemas de banco de dados relacionais, o tempo necessário para acessá-lo. Isto é devido ao fato de que consultas são executadas para buscar e atualizar objetos que encontram-se decompostos e armazenados em várias relações. Estas relações precisam ser reunidas ("join") para implementar a recuperação.

São utilizadas três estratégias para acelerar o tempo de busca: "caching", pré-computação e pre-busca ("pre-fetching").

Processos de aplicação guardam na memória cachê os objetos obtidos do banco de dados. Um objeto indexado é mantido na memória principal para permitir que o sistema, em tempo de execução, determine rapidamente se um objeto referenciado está na memória cachê ou não. Cada entrada no índice contem as tuplas relacionadas com os objetos e os seus endereços na memória.

IU.3 - Bancos de Dados Distribuídos

Um banco de dados distribuído (BERNSTEIN 1987), (CASANOVA 1985), (CERI 1984), (DATE 1988), (IBM 1988) e (KORTH 1989), é uma coleção de dados distribuídos por vários computadores interligados em rede. Cada computador (nó da rede) tem capacidade de processamento autônoma e

pode tratar aplicações locais. Cada computador também participa na execução de pelo menos uma aplicação global que exige acesso a dados localizados em mais de um computador da rede.

O objetivo principal de um banco de dados distribuído (BDD) e fornecer o que normalmente se denomina por transparência de localização. A transparência de localização significa que os usuários e os programas de usuários não deveriam precisar saber onde (isto e, em que nó) qualquer item particular de dados esta localizado. Ao invés disso, todas essas informações de localização devem ser mantidas pelo sistema como parte de seu catálogo, e todas as solicitações de dados, por usuários, deveriam ser interpretadas pelo sistema de acordo com aquela informação.

Outro conceito importante em BDD é a replicação de dados. A replicação de dados significa que determinado objeto de dados lógicos pode possuir diversos representantes armazenados, em diferentes nos. A vantagem de tal disposição é possibilitar uma melhoria tanto na desempenho como na disponibilidade: as operações podem operar com réplicas locais, ao invés de terem de se comunicar com nós remotos, e um certo objeto de dados permanece disponível para processamento, desde que pelo menos uma réplica esteja disponível, isto e, desde que pelo menos um nó que detenha uma cópia daquele objeto esteja acessível. Naturalmente existem desvantagens também: a atualização de um objeto requer que ela seja propagada para todas as cópias existentes em todos os nós, e obviamente exige mais espaço de armazenamento e atividade de I/O.

A transparência de localização e a replicação, indicam que (idealmente) o sistema distribuído deve Parecer-se com um sistema centralizado para o usuário. O usuário deverá raciocionar somente em termos de objetos de dados lógicos, e não se preocupar cõm onde ou quantas vezes esses objetos são fisicamente armazenados.

Entre as razões para se usar um BDD podemos citar:

a) Organizacionais

O empreendimento a que o banco de dados serve muitas vezes é logicamente distribuído (divisões, departamentos, projetos, etc.) e provavelmente, será fisicamente distribuido também (fábricas, armazens, escritórios, etc.). Assim, um BDD descreve e suporta mais naturalmente a sua estrutura organizacional.

b) Econômicas

Os investimentos necessários em processamento de dados ("software" e "hardware") são menores e mais bem utilizados. A principal razão e o compartilhamento de recursos.

c) Capacidade

Expansões constantes das aplicações numa organização excedem rapidamente a capacidade de um único processador. Como o custo de varias máquinas menores e da comunicação entre elas tem decrescido nos Últimos anos, isso as torna competitivas em relação as máquinas de grande porte.

d) Crescimento

é muito mais fácil adicionar novos processadores (nós) num sistemas distribuido do que trocar um sistema centralizado por outro maior. O impacto do crescimento num sistema distribuído e bem menor.

e) Autonomia

A distribuição do sistema permite que grupos individuais, dentro da empresa, exercitem o controle local sobre seus próprios dados, com responsabilidade local; tornando-os menos dependentes de outros centros de processamento de dados. Ao mesmo tempo, a distribuição permite que esses grupos locais acessem dados em outras localizações, quando necessário.

f) Disponibilidade

BDDs, especialmente empregando replicação de dados, são usados para obter alta disponibilidade. Nesses ambientes, o impacto de uma falha na maquina, ou numa ligação entre máquinas, é reduzido. Com replicação de dados e programas, sistemas agem como "backups" entre si durante períodos de manutenção ou falhas.

g) Custo e Performance

A distribuição pode reduzir custos de comunicação e tempo de resposta, colocando os dados e a capacidade de processamento próximo a seu ponto normal de utilização (a maioria das acessos é 'local). Também deve ser levado em conta o alto grau de paralelismo real existente.

Com esta visão de BDD abordaremos agora o tópico de interesse neste trabalho.

IV.3.1 - Gerenciamento de Transações

A finalidade básica de qualquer sistema de banco de dados é executar transações. No entanto, a noção de transação requer definição cuidadosa no ambiente distribuído porquê pode envolver a execução de código em múltiplos nós do sistema. Segundo Gray em (GRAY 1980), transação é uma unidade de recuperação (unidade lógica de trabalho), e o agente é a execução do processo por conta de alguma transação em particular, de algum nó em especial (isto e, um representante daquela transação, naquele no). O inicio da transação faz com que o agente comece a executar em algum nó. A medida que esse agente executa, isso pode fazer com que outros agentes comecem a executar em outros nós, e assim por diante.

Cori em (CERI 1984), assume que cada nó (processador local) tem seu gerenciador de transação local, o qual é capar de implementar transações locais. Com isso, pode-se aproveitar as técnicas e as sistemas existentes que implementam transações não distribuídas.

O problema de recuperação em sistemas distribuídos é um pouco mais delicado. Em cada nó existe um gerenciador de transações local que utiliza as mesmas técnicas de reconstrução local descritas anteriormente (seção III.6). São utilizados tambem gerenciadores de transação distribuídas que rara existirem necessitam que os gerenciadores de transação local sejam aptos a:

a) assegurar a atomicidade de uma transação;

b) gravar alguns registros em memória estável em nome do gerenciador de transação distribuída (necessidade de gravar novos tipos de registros no diário que devem sobreviver a falhas).

Num sistema de gerenciamento de BDD, o problema da indivisiblidade da transação é especial, porque uma Única transação pode envolver a execucão de código em múltiplos nós e tratar com atualizações em múltiplos nós. Exatamente

como em sistemas de gerenciamento de banco de dados tradicionais, a transação é e continua sendo a unidade de execução, ou seja, a unidade de trabalho. Por causa disto, transações fornecem integridade ao banco de dados. Os problemas de recuperação em ambientes de BDD são discutidos a seguir.

IV.3.2 Falhas num Sistema Distribuído

Um sistema distribuído consiste de dois tipos de componentes: nós, que processam informações, e elos de comunicação, que transmitem informações entre os nós. Um sistema distribuído é comumente representado por um grafo onde os nódulos são nós e os arcos não direcionados são elos de comunicação bidirecionais (figura IV.5).

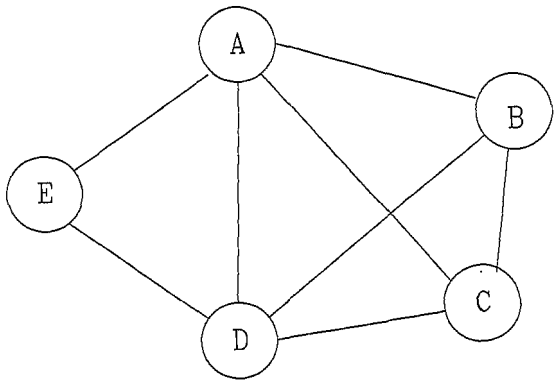


Figura IV.5. Exemplo de uma rede.

Se assumirmos que este grafo é conectado, significa dizer que existe um caminho de cada nó para outro. Assim, cada dois nos podem se comunicar diretamente através da ligação entre eles, ou indiretamente através de uma cadeia de ligações. A combinação de "hardware" e "software" responsável por mover mensagens entre nós e chamada de "rede". Em (KORTH 1989) e (CERI 1984) podemos encontrar mais informações sobre topologia de rede.

IV.3.2.1 Falhas em Nós

Mesmo se cada nó estiver funcionando apropriadamente ou tiver falhado, diferentes nós podem estar em diferentes estados. Uma falha parcial e uma situação onde alguns nos estão operacionais enquanto outros não. Uma falha total acorre suando todas os nós não estão operacionais.

é complicado tratar falhas parciais. Isto porque os nós operacionais podem estar incertos sobre o estado doç nós que falharam. Como podemos ver, nós operacionais podem se tornar obstruídos, incapazes de compromissar ou abortar transações, ate que tal incerteza seja resolvida. Um objetivo importante no projeto de protocolos de comprometimento atômico e minimizar o efeito da falha de um nó sobre a habilidade de outros nós continuarem o processamento.

IU.3.2.2 Falhas de Comunicação

Clos de comunicação também são objetos de falhas. Tais falhas podem impedir processos em diferentes nós de se çomunicarem. Uma variedade de falhas de comunicacão são possíveis, entre elas:

a) uma mensagem pode ser corrompida por causa de ruídos na ligação;

b) uma ligação pode funcionar mal temporariamente, causando a perda completa da mensagem, ou;

c) uma ligação pode ser interrompida por um tempo, causando a perda da mensagem enviada.

Corrupção na mensagem pode ser suportada através do uso de código de detecção de erros, e pela retransmissão da

mensagem na qual o receptador detectou um erro. Perda de mensagens ocasionadas por falhas de ligações temporárias podem ser suportadas pela retransmissão da mensagem perdida. A probabilidade de perda de mensagens ocasionadas por interrupção (quebra) na ligação pode também ser resolvida por reenvio de mensagens.

Infelizmente, mesmo com um reenvio automático de mensagens, uma combinacão de falhas de ligação com falhas no nó pode inutilizar a comunicação entre nós. Isto pode acontecer se todos os caminhos entre dois nós contém um nó com falha ou uma ligação interrompida. Este fenômeno é chamado de "partição na rede".

Assim que os nós se recuperarem e as ligações forem reparadas, a comunicação entre os nós é reestabelecida permitindo a troca de mensagens.

Pode-se reduzir a probabilidade de ocorrer uma partição de rede projetando-se uma rede altamente conectada, isto é, uma rede onde a falha de alguns nós e ligações não interrompa todos os caminhos entre qualquer par de nós. Entretanto, fazer uma rede altamente conectada requer o uso de mais componentes significando maiores custos. Alem disso, a topologia da rede é frequentemente restringida por outros fatores, como geografia ou meio de comunicação. Assim, a habilidade de se evitar partições é limitada.

IV.3.2.3 Detectando Falhas por "Timeouts"

Para se detectar que um nó não pode se comunicar com outro, usualmente se usa o "timeout". Um nó A envia uma mensagem para o nó B e espera uma resposta com um período de tempo pré-determinado, chamado de período de "timeout". Se a resposta chega, claramente A e B podem çe comunicar. Se o período pré-determinado termina e A ainda não recebeu a resposta, A conclui que não pode se comunicar com B.

Se o período de tempo é subestimado, o no pode pensar que não pode se comunicar com o outro quando, de fato, ele pode. Tais erros são chamados de "falhas de timeout".

IV.3.3 Comprometimento Atômico

Toda vez que a execução de uma transacão envolver mais de um banco de dados local, o SGBD distribuído deve assegurar que a finalização da transação é consistente. Isto significa que ou todos os nós se comprometem ou todos eles abortam as efeitos da transação.

Para facilitar, é assumido que para cada transação distribuida T, existe um processo em cada nó onde T é executado. Este processo realiza o protocolo de comprometimento atômico para T. O processo no "nó casa" (o nó que originou a transação) de T e chamado de coordenador de . Os processos restantes são os participantes de T. Os participantes sabem o nome do coordenador, mas não necessariamente sabem um do outro.

O que normalmente ocorre é assumir que cada nó contém um diário de transações distribuídas (diário TD) onde coordenadores e participantes naquele nó podem gravar informações sobre transações distribuídas. O diário TD deve ser armazenado em memória estável, porque seu conteúdo deve sobreviver a falhas no nó. Na pratica o diário TD pode ser parte do diário do gerenciador de dados do nó.

Aproximadamente falando, um protocolo de comprometimento atômico (PCA) é um algorítmo para o coordenador e participantes tal que todos eles comprometem ou abortam a transação. Cada processo deve votar: Sim ou Não, podendo alcançar exatamente uma da duas decisões: Compromissar ou Abortar. O PCA é um algorítmo para os processos alcançarem decisões tais como:

CA1: Todos os processos que alcançarem uma decisão alcançam a mesma.

CA2: Um processo não pode reverter sua decisão depois de atingi-la.

CA3: A decisão de compromissar pode somente ser alcancada se todos os processos votarem SIM.

CA4: Se não existirem falhas e todos os processos votarem Sim, então a decisão será de Compromissar.

CA5: Considerando qualquer execução contendo somente falhas que o algorítmo e projetado para tolerar. A qualquer ponta nesta execucão, se todas a5 falhas existentes são reparadas e nenhuma nova falha ocorre por um tempo suficientemente longo, então todos os processos eventualmente alcançarão uma decisão.

IV.3.3.1 Protocolo de Comprometimento em Duas Fases (2PC)

O mais simples e mais popular PCA e o protocolo de comprometimento em duas fases (2PC). Assumindo que não ocorra falhas, ele funciona aproximadamente assim:

a) O coordenador envia uma mensagem de pedido de voto para todos os participantes.

b) Quando o participante recebe o pedido de voto, ele responde enviando ao coordenador uma mensagem contendo seu voto: Sim ou Não. Se o participante votar Não, ele decide Abortar e parar.

c) O coordenador coleta as mensagens de votos de todos os participantes. Se todas elas forem Sim e o coordenador

também votar Sim, então o coordenador decide Comprometer e envia uma mensagem de Comprometimento para todos os participantes. Caso contrario, o coordenador decide Abortar e envia uma mensagem de Aborto para todos os participantes que votaram Sim. Em ambos os casos, o coordenador então pára.

d) Cada participante que votou Sim espera por uma mensagem de Comprometimento ou Aborto do coordenador. Quando ele recebe esta mensagem, ele decide de acordo com ela e pára.

As duas fases deste protocolo (2PC) são a fase de voto a) e b) e a fase de decisão c) e d). O período de incerteza dos participantes começa quando eles enviam um ÇIM para o coordenador (passo "b") e termina quando eles recebem um Comprometimento ou Aborto (passo "d"). O coordenador não tem período de incerteza uma vez que ele decide assim que ele vota (com o conhecimento, naturalmente, dos votos dos participantes).

É importante enfatizar que enquanto os participantes estão esperando pela decisão do coordenador, eles não tem permissão para o comprometimento ou para o aborto de uma transacão unilateralmente. Neste estado, todo o controle local sobre os recursos seguros pela transação estão submetidos ao coordenador do comprometimento em duas fases.

Para este protocolo satisfazer a condição CA5, deve-se fornecer um "timeout" conveniente às ações para cada passo do protocolo no qual um processo esta esperando por uma mensagem. Além disso, deve-se indicar qual a informação a ser mantida no diário TD e como usa-la na recuperação. A seguir, o segundo requisito da condição CA5 (recuperação) será melhor explicado, uma vez que faz parte diretamente do assunto deste trabalho.

A integridade de cada banco de dados envolvido num processamento de transação é garantida através do processo

de comprometimento em duas fases. Como sabemos, os estados alcançados pelos participantes e pelo coordenador são gravadas em diários. Se nenhuma informação do diário for perdida, o protocolo de comprometimento em duas fases e capaz de garantir a integridade da unidade de trabalho através de qualquer falha que possa acontecer durante seu processamento.

IV.3.4 Reconstrução

Para relembrar seu estado na hora da -Falha, cada processo deve manter alguma informação no diário TD do seu nó que sobrevive a falhas. Naturalmente, cada processo tem acesso somente ao seu diário TD local. Assumindo-se que o protocolo de terminação cooperativa e usado, abaixo e mostrado como a diário TD é gerenciado.

a) Quando o coordenador envia uma mensagem de pedido de voto, ele grava um registro início-2PC no diário TD. Este registro contém os identificadores dos participantes e pode ser gravado antes ou depois do envio da mensagem.

b) Se um participante vota Sim, ele grava um registro SIM no diário TD, antes de enviar o SIM para o coordenador. Este registro contem o nome do coordenador e a lista dos outros participantes (que e fornecida pelo coordenador na mensagem de pedido de voto). Se o participante vota Não, ele grava um registro de aborto ou anteç ou depois do participante enviar Não para o coordenador.

c) Antes do coordenador enviar a mensagem de Comprometimento para os participantes, ele grava um registro de comprometimento no diário TD.

d) Quando o coordenador envia uma mensagem de Aborto para os participantes, ele grava um registro de aborto no

diário TD. O registro pode ser gravado antes ou depois do envio da mensagem.

e) Depois de receber a mensagem de Comprometimento (ou Aborto), um participante grava um registro de Comprometimento (ou Aborto) no diário TD.

A gravação de um registro de comprometimento ou aborto no diário TD e o ato pelo qual o processo decide o comprometimento ou aborto.

Quando um nó S recupera-se de uma falha, o destino de uma transação distribuida executando em S pode ser determinado examinando-se seu diário TD:

a) Se o diário TD contém um registro de início-2PC, então S foi o hospedeiro do coordenador. Se ele também tiver um registro de comprometimento ou aborto, então o coordenador decidiu antes da falha. Se nenhum dos dois registros e encontrado, o coordenador pode agora unilateralmente decidir pelo Aborto inserindo um registro de aborto no diário TD. Para este trabalho, é crucial que o coordenador primeiro inclua o registro de comprometimento no diário TD e depois envie a mensagem de Comprometimento.

b) Se o diário TD não contem um registro de início-2PC, então S foi hospedeiro de um participante. Existem três casos para considerar:

b.1) O diário TD contem um registro de comprometimento ou aborto. Então o participante alcançou sua decisão antes da falha.

b.2) O diário TD não contem um registro Sim. Então ou o participante falhou antes de votar ou votou Não (mas não gravou um registro de aborto antes da falha. Esta e a

causa de se gravar o registro Sim antes de enviá-lo). Ele pode portanto abortar unilateralmente através da inserção de um registro de aborto no diário TD.

b.3) O diário TD contem um registro Sim mas nenhum registro de comprometimento ou aborto. Então o participante Calhou enquanto estava no período da incerteza. Ele pode tentar alcançar uma decisão usando o protocolo de terminação. Relembramos que o registro Sim incluiu o nome do coordenador e dos participantes, que são necessários para o protocolo de terminação.

Para evitar confusão de registros de diferentes transações, os registros de "início-2PC", "sim", "comprometimento", e "aborto" devem conter o nome da transação à que eles se referem.

Outro ponto importante é fazer o recolhimento de lixo no diário TD para tirar as informações desnecessárias. Existem dois principios básicos considerando este recolhimento de lixo:

RLI: Um nó não pode apagar registros de uma transação T do seu diário TD até que pelo menos seu gerenciador de recuperação tenha processado a operação de comprometimento ou aborto.

RL2: Pelo menos um nó não deve apagar os registros da transação T do seu diário TD até que este nó tenha recebido mensagens indicando que a operação de comprometimento ou aborto tenha sido processada por todos os outros nós onde T foi executado.

RL1 declara que um nó envolvido na execução de T não pode se esquecer de T antes que seus efeitos naquele nó tenham sido realizados. RL2 diz que algum nó envolvido na execução de T deve lembrar do destino de T até que aquele nó saiba que os efeitos de T foram realizados em todos os nós. Se Isto não for verdade e um nó recuperar-se de uma falha e encontrar-se na dúvida sobre o destino de T, poderá nunca ser capaz de saber o que decidir sobre T, violando CA5.

IU.3.5 Impasses ("deadlocks") em Bancos de Dados Distribuídos

O tratamento de "deadlocks" em SBDDs e muito semelhante ao caso centralizado (seção III.3.3.4). Os algorítmos de prevenção e detecção existentes para os sistemas centralizados podem ser usados em sistemas distribuídos, desde que algumas modificações sejam feitas.

A prevenção de impasse pode resultar em algumas esperas desnecessárias e repetidas. Alem disso, algumas das técnicas de prevenção de "deadlocks" podem exigir mais locais sendo envolvidos na execução de uma transação, do que de outro modo.

Se "deadlocks" forem permitidos contando com a estratégia de detecção de "deadlocks", o principal problema em um sistema distribuido é decidir como manter o grafo de espera. Diversas técnicas existem na literatura para tratar deste assunto, geralmente elas exigem que cada local mantenha um grafo de espera local. Os nós do grafo correspondem a todas as transações (tanto local como não local) que estão correntemente retendo ou requisitando qualquer dos itens daquele local. Por exemplo, na Figura IV.6, temos um sistema consistindo em dois locais, cada um mantendo seu grafo de espera. As transações T2 e T3

aparecem em ambos os grafos, indicando que as transações são itens de requisição em ambos os locais

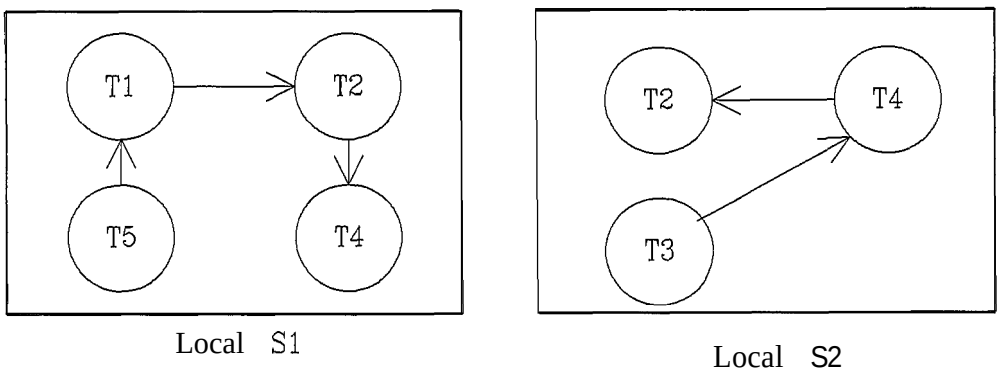


Figura IV.6 - Grafos de espera local.

Estes grafoç de espera local são construidos de maneira usual para locais e itens de dados. Quando uma transaçao Ti no local S1 precisa de um recurso retido pela transação Tj no local 52, uma mensagem de requisição e enviada por Ti para o local S2. A aresta T1 ---> Tj é então inserida no grafo de espera local de S1.

é claro que se algum grafo de espera local possui um ciclo acorre um "deadlock". Por outro lado, o fato de não haver ciclos em qualquer um dos locais dos grafos de espera não significa que não hajam "deadlocks", pois a união dos grafos de espera local pode conter um ciclo (figura IV.7).

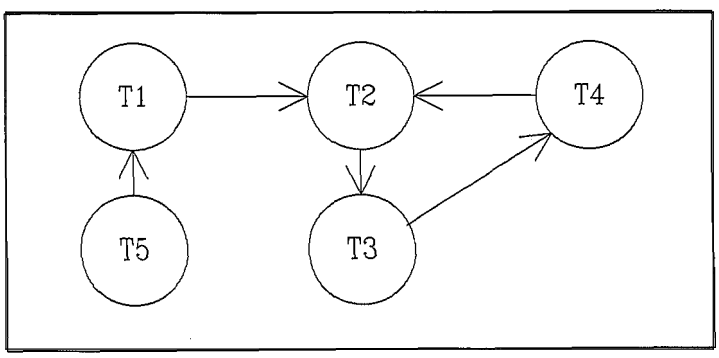


Figura IV.7 - Grafo da espera global.

Em (KORTH 1989) são apresentados vários métodos para organizar o grafo de espera em srstemas distribuidos.

IV.3.6 "Timestamping"

A idéia central no esquema de "timestamping" é sue a cada transacão é dado um unico "timestamp", que é usado para decidir a ordem de sequência das transações. Para generalizar o esquema centralizado para um esquema distribuido, deve-se desenvolver um esquema geral de "timestamps" Únicos. Uma ver que isto seja feito, o protocolo prévio pode ser dirntamente aplicado ao meio não replicado.

A questão básica e que cada local deve gerar um único "timestamp" local usando ou um contador lógico ou um relógio local. O "timestamp" Único é obtido pela concatenação do "timestamp" local unico com o identificador de local, o qual precisa ser Único (Figura IV.8). A ordem de concatenação é importante. O identificador de local é usado na última posição significativa, para assegurar que o "timestamp" gerado em um local que não seja sempre maior do sue aqueles gerados em outro local.

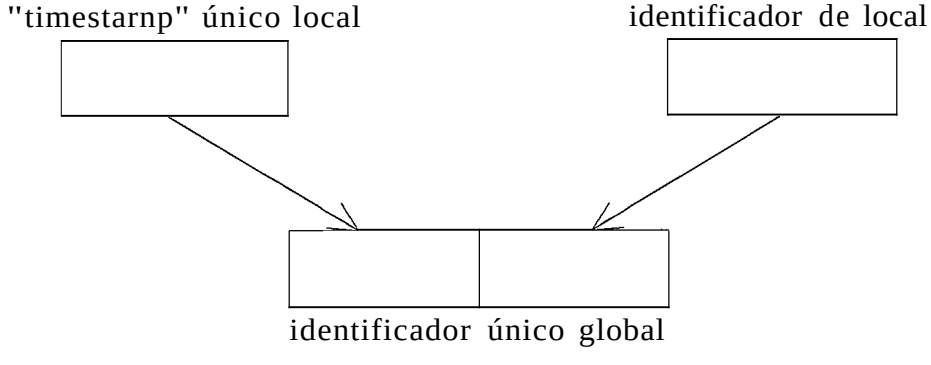


Figura IV.8 - Geração do "timestamp" unico

A vantagem desta abordagem é que nenhum bloqueio é feito e portanto, "deadlocks" não ocorrem. Com isso, os Custos de comunicação de bloqueios e detecção de "deadlocks" são evitados.

Para Date em (DATE 1988), há algum sentimento de que as técnicas baseadas em "timestamping" possam ser mais apropriadas para resolver problemas de concorrência e recuperação nos sistemas distribuídos. é possível também, que tais técnicas tornem-se mais amplamente utilizadas num futuro próximo. Contudo, mesmo que isso aconteça, a5 técnicas de bloqueio continuarão a ser importantes para os sistemas centralizados e também para os processadores locais individuais dentro de um sistema distribuido.

CAPÍTULO V

PROPOSTA DE UM SUB-SISTEMA DE RECONSTRUÇÃO PARA O SGBD COPPEREL-PC

Neste capitulo descrevemos uma proposta de implementação de um sub-sistema de reconstrução para o SGBD COPPEREL-PC. No entanto, para uma melhor compreensão da proposta é essencial que descrevamos, mesmo que sucintamente, os nódulos componentes do sistema COPPEREL-PC mais importantes e a sua arquitetura. Maiores informações a respeito deste assunto podem ser encontradas em (MATTOSO 1985a), (MATTOSO 1985b), (MATTOSO 19871, (PALERMO 1985) e (ZAKIMI 1982).

V.I - A Arquitetura do COPPEREL-PC

Esta seção será extraída quase que inteiramente do trabalho de Mattoso (MATTOSO 1987). Isto porquê o referido trabalho e a mais completa e atual fonte de referência sobre o sistema COPPEREL.

Mattoso nos mostra o sistema COPPEREL (figura V.ll como sendo a composição de três módulos principais independentes:

BSTRAP - responsável pela inicialização e configuração do SGBD (MATTOSO 1985a);

SUPCSQ - responsável pela administração do banco de dados (MATTOSO 1985b);

COPPEREL - é o SGBD propriamente dito.

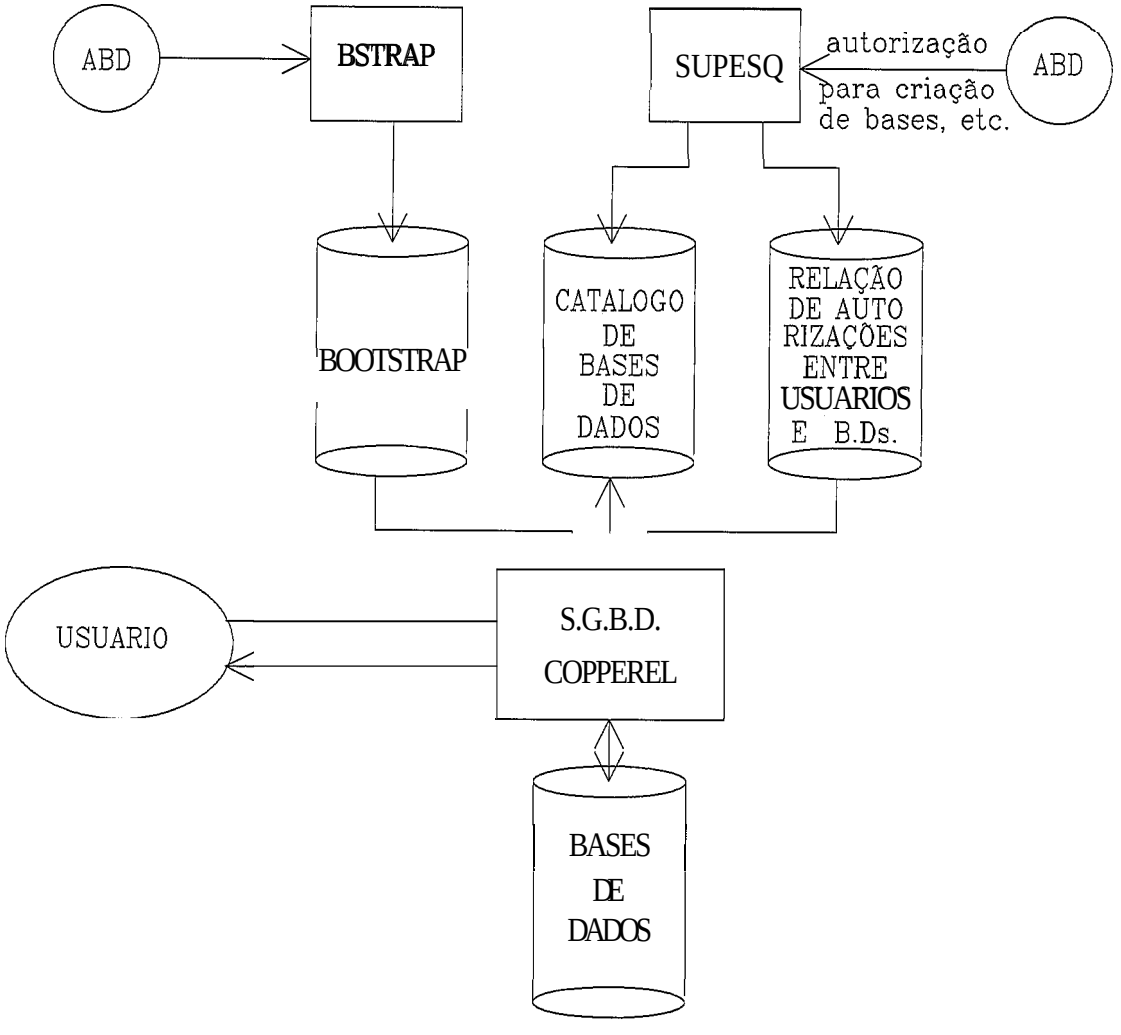


Figura V.9 - O sistema COPPEREL

V.1.1 - O Componente BSTRAP

O compunente BSTRAP é responsável pela geração do arquivo "bootstrap" que contem as informações que descrevem a representação dos dados no nível conceitual, ou seja, o esquema (MATTOSO 1985a). Este componente calcula os endereços e aloca o espaço físico das tabelas do esquema e dos seus respectivos índices.

E através do arquivo "bootstrap" que a base de dados é confiyurada, sendo carregada com as relações do esquema se autodescrevendo. **As** tabelas do esquema só podem ser alteradas pelo próprio sistema, embora como consequência de comandos do usuário.

V.1.2 - O Componente SUPESQ

O componente **SUPESQ** tem como objetivo gerar o super-esquema do COPPEREL, composto por duas tabelas: TAB-BASE-DADOS (TBD) **E** TAB-USUARIO-BASE (TUB), que descrevem as bases gerenciadas pelo sistema formando o chamado "super-esquema". Essas duas tabelas, **TBD** e TUB, se diferenciam das demais relações do esquema por não serem especificas de uma única **base** de dados e por isso não se materializarem nas bases de dados, residindo em arquivos distintos conforme a figura V.1. **A** tabela TAB-USUARIO-BASE relaciona usuários com as base de dados as quais têm acesso. **A TAB-BASES-DADOS** descreve as base de dados cadastradas. Essas duas tabelas controlam a criação, abertura e fechamento das bases de dados e sua manutenção é responsabilidade do administrador do banco de dados (ABD). Mais informações sobre o componente **SUPESQ** e sobre as tabelas do super-esquema (TBD e TUB) podem ser encontradas em (MATTOSO 1985b).

V.1.3 - O Componente SGBD COPPEREL

O componente SGBD **COPPEREL,** em seu nível mais alto, pode ser visto conforme o diagrama do projeto modular do SGBD (figura V.2) encontrado em **(MATTOSO** 1987). Porém, os módulos mostrados na figura são na verdade compostos de outros módulos mais básicos.

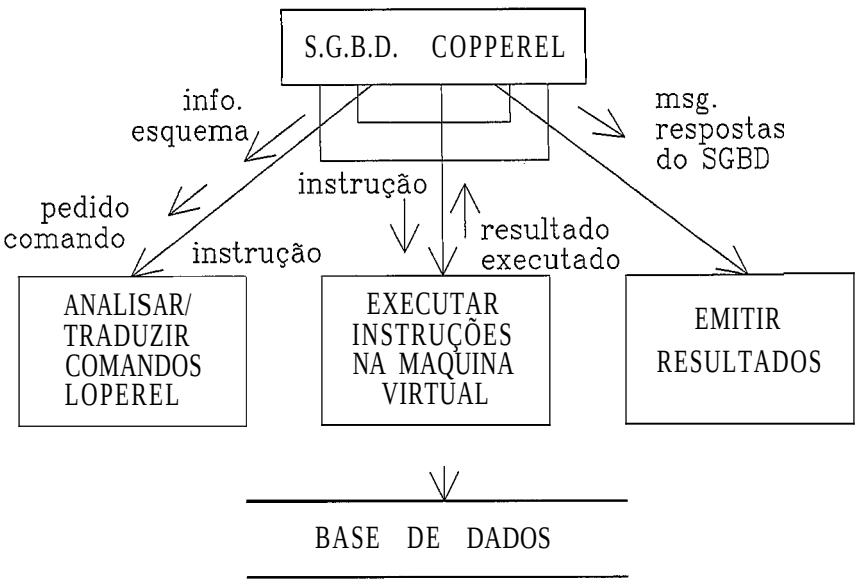


Figura V.2 - Projeto modular do SGBD COPPEREL.

O módulo principal gerencia a interação do usuário com o banco de dados. O SGBD pede comandos ao usuário que os envia linha a linha para o "analisador/tradutor". O analisador gera, para cada comando, instruções em código intermediário a serem executadas na máquina virtual COPPEREL. Os resultados destas instruções serão utilizadas pelo analisador manipulando informações do esquema conceitual (laço mais interno) ou exibidos ao usuário através do módulo de emissão de resultados, como resposta ao comando submetido pelo usuário. Em seguida o controle volta ao módulo principal que novamente pede comandos ao usuário (laço mais externo).

O detalhamento dos módulos máquina virtual COPPEREL e analisador/tradutor podem ser encontrados no trabalho de Mattoso (MATTOSO 1987) nos capítulos III e IV respectivamente.

V.1.4 - Níveis de Representação

O sistema COPPEREL tem três níveis de representação de dados: o nível conceitual, o nível externo e o nível interno.

No nível conceitual temos registros COPPEREL, que em conjunto compõem uma "relação" da base de dados. Uma relação da base de dados, é aquela que possui uma existência independente, ao contrario do que ocorre com uma "vista", que é derivada de outras relacões da base. Existe um esquema conceitual para cada base de dados gerenciada pelo COPPEREL. As relações desses esquemas são criadas automaticamente pelo sistema quando o usuário cria uma nova base de dados, já contendo suas próprias descrições. O usuário pode consultar qualquer uma dessas relações desde que autorizado para isso, mas não atualiza diretamente nenhuma delas (seção V.1.1).

No nivel externo o usuário tabalha com relações que tanto podem ser "vistas" quanto relações da base. Uma vista no sistema COPPEREL é uma relação cuja existência depende de outras relações que podem ser vistas ou relações da base.

No nível interno todas as relações da base são representadas em um único arquivo físico chamado de base de dados nas figuras V.l e V.2. Um registro da base de dados pode conter uma ou mais tuplas de uma ou mais relações da base. Uma relação pode ter índices sobre todos as seus atributos e eles também são armazenados no arquivo que contém as relacões da base.

V.I.4.1 - A Base de Dados COPPEREL

Uma base de dados equivale a um único arquivo físico em disco do ponto de vista do sistema operacional. A maquina COPPEREL divide esse arquivo logicamente, de modo a conter as relações (do esquema e do usuário), índices criados

cobre essas relações, área de trabalho e listas de ordenações sobre relações. A figura V.3 mostra em linhas gerais a estrutura interna de uma base de dados.

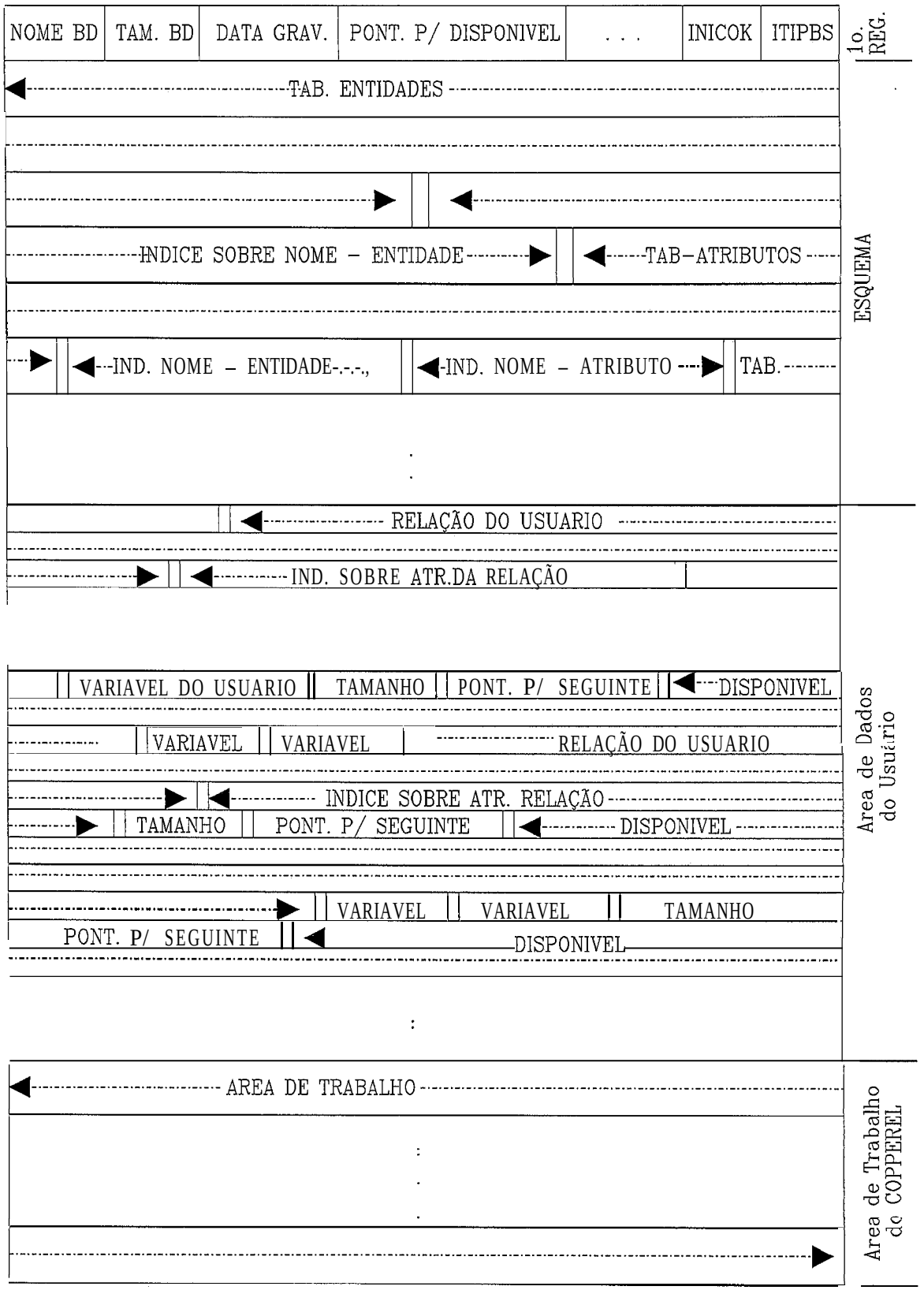


Figura V.3 - Estrutura física da base de dados COPPEREL.

Tanto o tamanho do registro da base de dados, quanto o numero de registros desse arquivo são parametrizados e podem ser definidos no momento da criação da base de dados.

O primeiro registro deste arquivo contem características da base de dados que ele representa.

V.1.4.2 - A Memória de Paginasão

Para melhor aproveitamento da memória principal, o sistema possui um método próprio de paginação, que controla a alocação/liberação de "buffers" internos, cada um contendo uma ou mais tuplas. O critério para escolha de "buffers" a serem substituídos é baseado em características de localidade e sequencialidade da tuplas acessadas, o que garante um gasto mínimo de tempo de paginação. As páginas que sofrem alterações na área de "buffers" são marcadas, de modo que quando forem substituídas no processo de paginação possam ser reescritas na base de dados ou simplesmente ignoradas, conforme estejam marcadas ou não. Cada pagina lógica do sistema equivale a um registro físico no arquivo definido para a base. A área de paginação é utilizada inclusive pelas tuplas das relações do esquema, visto que são tratadas da mesma forma sue as do usuário.

A estratégia de substituição de paginas adotou uma política semelhante a NRU ("not recently used"), ou seja, a página a ser retirada da memória deve ser a que menos vem sendo usada ultimamente. Para implementar a política de NRU, foi adotado uma espécie de rodízio entre as páginas dos "buffers" e a tabela de mapeamento passa a ser também uma tabela de envelhecimento que indica através da posição da pagina na tabela se ela tem sido usada recentemente ou não.

Quando uma página é referenciada e não se encontra na memória, ela e copiada para um "buffer" livre e entra na

primeira posição da tabela de envelhecimento, sendo as páginas que ocupavam a tabela anteriormente, deslocadas para baixo de uma posição.

Quando a tabela está cheia, o algoritmo verifica se a página que esta no "buffer" da Última posição da tabela foi allerada ou não. Se foi alterada, essa pagina é reescrita no disco, caso contrário esse "buffer" é considerado livre, recebe a página desejada e essa entrada na tabela é copiada para a primeira posição, sendo feito o deslocamento adequado.

Para evitar que as paginas que vem sendo referenciadas com frequência, mas que já estão na memória, cheguem a Última posição da tabela e portanto sejam retiradas da memória, o algoritmo avança a pagina de uma posição na tabela de envelhecimento sempre que a página é referenciada.

Existe ainda um meio de influir nessa política e alterar a localização que a pagina recebe no seu referenciamento. A primitiva que faz a gerência dos "buffers", possui um parâmetro de entrada que impõe a posição da tabela onde a página deve "envelhecer".

De acordo com a filosofia de modularidade e portabilidade do COPPEREL, o número de "buffers" da memória de paginação pode ser fixado na instalação do sistema, assim como o tamanho máximo de uma base de dados, o tamanho de uma página, a cardinalidade máxima de cada relação do esquema e o limite do espaço na base de dados reservado para o esquema conceitual e para relações do usuário.

U.2 - Visão Geral da Proposta

O primeiro capítulo deste trabalho aponta como principal fator de motivação para a realização do mesmo o fato do

COPPEREL-PC ainda não possuir dois importantes conceitos para SGBDs. O conceito de transação como unidade lógica de trabalho ou unidade de consitência, no sentido em que deve transformar a base de dados de um estado consistente inicial para um novo estado consistente; e o conceito de reconstrução, ou seja, procedimentos de recuperação de falhas sue assegurem a consistência dos dados e a integridade do banco de dados.

Antes de falarmos da proposta em sí, e necessario relembrarmos de algumas características do sistema em questão. O COPPEREL-PC é um sistema mono-usuário com um esquema de propagação NÃO ATÔMICA (seção III.5.4), onde a propagação indivisível de um conjunto de paginas não é permitida. Sua estratégia de substituição, ROUBAR, permite que páginas modificadas pertencentes a uma transacão não encerrada sejam escritas e/ou propagadas a qualquer tempo (seção III.5.4). O sistema possui ainda um critério de manusear buffers de FORÇAR, onde todas as páginas modificadas são escritas e/ou propagadas durante o proceçsamento de fim de transação (seção III.5.4).

O conceito de transação será implementado conforme havia sido estipulado (mas não implementado) na linguagem LDPEREL. Na proposta, uma transação inicia com uma operação de ABRIR BASE DE DADOÇ nome-da-base ou de CRIAR BASE DE DADOS nome-da-base e termina com uma operação de FECHAR BASE DE DADOS ÇSEM ATUALIZARI. Todo e qualquer comando que esteja dentro deste bloco pertence a essa trasação. A extensão [SEM ATUALIZARI significa um "rollback" da transação, ou seja, uma falha de transação.

Ao longo deste trabalho ficou claro que nenhuma técnica de reconstrução isolada possibilita uma reconstrução plena de um banca de dados. Assim, decidimos usar duas técnicas de reconstrução: diário ("log") e descarga ("dump").

O uso destas duas técnicas é muito frequente em SGBDs e justifica-se por ser a maneira mais comum de se conseguir

manter um banco de dados num estado correto, ou seja, no estado imediatamente anterior a ocorrência de uma falha.

O diário será composto por partes de representações físicas (páginas), por isso, seguindo a classificação dada em IV.3.3.1 será um "diário físico a nível de página". As páginas modificadas do banco de dados serão gravadas no diário imediatamente antes ("before images") e imediatamente depois ("after images") de serem gravadas fisicamente no banco de dados. Junto com as paginas gravadas no diário serão gravadas algumas informações importantes para a reconstrução, como o estado do banco de dados no ato do fechamento (se foi normal ou ocorreu algum problema), registros referentes ao super-esquema, e também registros de comprometimento da transação.

A localização do diário será em disco, com o propósito de impactar o menos possível a performance do sistema, proporcionar reinícios rápidos e transparentes ao usuário em caso de falhas de transação e falhas de sistema, e usar a mínimo de recurso possível do ambiente no qual esta sendo executado.

O ponto de verificação será o orientado a transações (TOC - "Transaction Oriented Checkpoint") descrito na seção III.6.3.2. Uma vez que a estratégia de FORÇAR é utilizada pelo sistema, um registro de fim de transação escrito no diário nos indicara que a transação e durável, podendo ser interpretado como um ponto de verificação orientado a transações (HAERDER 1983).

A segunda técnica escolhida, a técnica de descargas ("dumps"), fará uso de utilitarios existentes apropriados para esta função. As descargas serão geradas periodicamente a critério do usuario, o que possibilitará a reconstrução da **base** de dados no caso de ocorrer uma falha no meio de armazenamento no qual a base de dados estava localizada. O critério de deixar a descarga sob responsabilidade do usuário da a ele total liberdade de escolher o momento

ideal de fazer a descarga. Outro ponto positivo nesta decisão é que também tira de cima do administrador do sistema a responsabilidade por eventuais perdas por falta de descarga.

As falhar de transação (pseudo-falhas) bem como as falhas de sistema usarão um algoritmo de "UNDO" (seção V.5) que ira desfazer o efeito da transação em questão utilizando para isso as imagens anteriores ("before images") armazenadas no diário. Falhas de meio serão resolvidas em dois passos: O primeiro passo será recuperar a banco de dados a partir da Última descarga realizada, o que nos dará um banco de dados consistente em algum tempo no passado. O segundo passo será usar um algoritmo de "REDO" (seção V.5) que ira refazer todas as transações realizadas no banco de dados após a ocorrência da Última descarga, utilizando para isso as imagens posteriores ("after images") armazenadas no diário.

V.3 - Restrições Existentes para a Implementação

Uma avaliação mais cuidadosa do ambiente e do sistema em que se dará a implementação mostra-nos que algumas restrições irão aparecer na implementação:

a) quanto ao tamanho do código do COPPEREL-PC. Atualmente seu código executável se encontra com aproximadamente 600 K, o que nos deixa uma margem muito pequena para incluir ou modificar seu código sem que seja necessário usar a técnica de "overlay".

b) quanto ao tempo de execução. O sub-sistema de reconstrução proposto deve preocupar-se em aumenta-lo o mínimo possível.

c) quanto a localização de diário. Se quisermos capacitar o sistema a fazer reconstrução de falha de meio é

necessário que o diário (pelo menos a parte referente a imagens posteriores "after images") seja localizado num dispositivo diferente do dispositivo que contem a base de dados. A localização do diário, ou parte dele, num disco flexível poderá causar uma queda de performance significativa por motivos abvios.

V.4 - Componentes do Sub-Sistema de Reconstrução do COPPEREL-PC

No capitulo III abordamos os principais elementos de um sistema de reconstrucão na tentativa de falar dos conceitos e eventos encontrados com maior frequência nos sistemas de reconstrução existentes. Aqui, mostraremos somente os elementos envolvidos com o sub-sistema de reconstrução proposto para o COPPEREL-PC, o que não inclui controle de concorrência.

V.4.1 - Transação

O conceito de transação como unidade lógica de trabalho, unidade de consistência e unidade de recuperação é essencial num SGBD. Na ausência desse conceito o usuário não e capaz de definir quais ações devem ser executadas de forma atômica (ou todas ou nenhuma), a integriade e a consistência do banco de dados pode ser comprometida pela ocorrência de possíveis falhas, alem de não existir a possibilidade do usuário desistir de efetuar uma operação ("rollback") sem que ela afete o banco de dados equivocadamente.

Nossa proposta de implementar o conceito de transação, como já falamos na visão geral da proposta (seção V.2), segue a definição já existente mas que não foi implementada. Na LOPEREL (ZAKIMI 1985), uma sessão é

constituída de uma sequência de transações, como no exemplo abaixo.

```
ABRIR SESSAO PARA USUARIO ALUNO/SENHA;

ABRIR BASE DE DADOS EXEMPLO;
(* comandos da transação sobre a base de dados EXEMPLO *)


FECHAR BASE DE DADOS;

ABRIR BASE DE DADOS BD1;
(* comandos da transação sobre a base de dados BD1 *)


FECHAR BASE DE DADOS;

FECHAR SESSAO;

ENC
```

Por sua vez, uma transação é um conjunto de gerências e comandos sobre uma determinada base de dados aberta. Assim, o usuário corrente deverá estar habilitado a abrir a base de dados ou a criar uma nova base de dados. Uma transação pode então ser iniciada através de dois comandos:

a) **CRIAR BASE DE DADOS** nome-da-base-de-dados **EM** dispositivo1 E dispositivo2 **PARA** inteiro REGISTROS **POR** ARQUIVO;

b) ABRIR **BASE DE DADOS** nome-da-base-de-dados;

O comando que encerra uma transacão é o seguinte:

FECHAR BASE DE DADOS [SEM ATUALIZARI;

Este comando sem a opção [SEM ATUALIZARI fecha a base de dados tornando todas as modificacões realizadas dentro da transação executada permanentes na base de dados. Se a opção [SEM ATUALIZARI for usada a base de dados será fechada sem manter as modificações realizadas dentro da transação executada, ou seja, a base de dados ficara como se a transação não tivesse sido executada ("rollback").

V.4.2 - Diário ("log")

Na visão geral da proposta (seção V.2) justificamos por que o diário foi escolhido como uma das técnicas de reconstrução, porém, não comentamos a escolha de usar um "diário físico a nível de pagina".

Na classificação de dados do diário da seção III.6.3.1 encontrada em (HAERDER 1983), temos cinco classificações de diário: diário de estado físico a nível de página, diário de transição física a nível de pagina, diário de estado físico a nível de caminhos de acesso, diário de transição a nível de caminhos de acesso e diário lógico orientado a registros.

Dessas técnicas, escolhemos para implementar no COPPEREL-PC a primeira, "diário de estado físico a nível de pagina". Essa opção possui as seguintes vantagens:

a) a unidade de gravação/recuperação do COPPEREL é a página, o que torna a escolha um caminho natural;

b) o diário usa as informações armazenadas nos "buffers" do banco de dados, que são sempre uma pagina e não um registro. Se escolhessemos uma técnica envolvendo

registros, haveria necessidade de criar algoritmos mais sofisticados para extrair somente os registros necessários do "buffer" ;

c) Se registros fossem gravados no diário seria necessária uma rotina para simular a gravação de registros de tamanho variável.

Como principal desvantagem desta tecnica temos a necessidade de um maior espaço em disco para gravar todas as páginas que sofrerem modificações ("before" e "after images").

A segunda tecnica, "transição física a nível de pagina", em vez de gravar as páginas **antes** e **depois** da alteração, grava a diferença entre elas. A vantagem desta técnica em relação a primeira é justamente diminuir o espaço necessário para o diário. Em contrapartida, ela exige um algoritmo para calcular essa diferença, e mais, esse algoritmo teria que ser aplicado cada vez que uma pagina fosse alterada ou quando o sub-sistema de reconstrução necessitasse voltar a pagina a seu estado anterior, aumentando com isso o custo durante o processamento normal.

A terceira e a quarta tecnica, "estado físico a nível de caminhos de acesso" e "transição a nível de caminhos de acesso", foram descartadas, pois apesar de diminuirem a quantidade de informações gravadas no diário e com isso necessitarem de menos espaço para o diário, trabalham com entradas (declarações) em estruturas de armazenamento (seção III.6.3.1), o que tornaria a implementação muito mais complexa.

Como Ultima opção, teríamos a tecnica de "diário lógico orientado a registros". Essa técnica é bastante interessante e muito usada, pois em vez de gravar representações físicas ou entradas em estruturas de armazenamento, ela grava descrições de operações de mais alto nível, ou seja, comandos de alteração e seus

parâmetros. Isto possibilita uma diminuicão significativa no espaço necessário para o diário mas exige uma complexidade maior na interpretação das entradas do diário por parte do algoritmo de reconstrução. Entre as razões que nos levaram a decidir não usar esta técnica temos:

a) para implementar um diário lógico e necessário expandir o repertório das operações de atualização do sub-sistema de reconstrução. Além da simples operação de gravação ("write"), seriam necessárias outras operações para que o sub-sistema de reconstrução pudesse desfazer e refazer uma operação.

b) a justificativa de a) torna-se mais seria suando sabemos que a LOPEREL (ZAKIMI 1985) é uma linguagem não só de manipulação de dadas, mas também de definição de dados. Isto significa aumentar ainda mais o repertório, ou seja, para cada comando de definição que implique em alteração no banco de dados, teríamos que prover o sub-sistema de reconstrução deste comando e de seu inverso.

c) nesta técnica algumas ações de desfazer e refazer correspondentes a registros do diário lógico só podem ser aplicados aos itens de dados em questão quando estes ainda estiverem na mesmo estado (lógico) de quando os registros do diário foram criados. A justificativa é simples (BERNSTEIN 1987), paginas modificadas que não sofreram um "flush" para o banco de dados estavel antes do sistema falhar podem corromper o banco de dados, uma ver que no reinício o sistema poderá tentar desfazer as modificações sem que elas realmente existam no banco de dados estável. Em Bernstein (BERNSTEIN 1987), são apresentadas três maneiras de contornar este problema, entre elas o usa do algoritmo chamado de "LSN-based logging algorithm".

Com tudo isso, chegamos a conclusão sue um diário físico a nível de paginas seria a solução mais adequada para o COPPEREL-PC, tanto pelas características do mesmo quanta

pela facilidade de implementação. é claro que as restrições comentadas na seção U.3 também influenciaram esta decisão.

U.4.3 - Descarga ("dump")

Essa técnica de reconstrução tem fundamental importância contra falhas de meio sendo de longe uma das mais empregadas em SGBDs. A técnica de descarga se empregada isoladamente é de pouca utilidade, uma ver que ela somente pode reconstruir um banco de dados num momento passado, embora valido, o que significa a perda de todo o trabalho realizado desde o momento do inicio da Última descarga. Entretanto, essa técnica e essencial para complementar outras técnicas de reconstrução, mais especialmente o diário.

Como já falamos na visão geral da proposta, a descarga poderá ser feita a qualquer tempo, usando-se para isso utilitários próprios existentes.

A descarga será estática, ou seja, enquanto ela estiver sendo feita não é possível usar a base de dados em questão. Maiores detalhes sobre essa técnica de reconstrução são encontrados na seção III.6.2.

Os arquivos que deverão ser copiados e guardados são os seguintes:

BDTHABA - base de dados;

TBD - infomações sobre as bases de dados disponíveis;

TUB - informações sobre os usuários das bases de dados;

SUPDADOS - informações a respeito da base de dados e de autorizações que geram o super-esquema;

LOGDADOS - informações sobre o sub-sistema de reconstrução;

BSTRAP - configuração da base de dados;

TRACE - relatório de saída do SUPESQ e do sistema;

SOCORRO - assistência interativa ("help") para o sistema;

ERROS - arquivo de mensagem de erros.

W.5 - A Escolha do Algoritmo de Recuperação

Um sub-sistema de reconstrução necessita desfazer ("UNDO") transações se ele permite uma transação não compromissada gravar no banco de dados estavel valores que ele modificou (BERNSTEIN 1987). O sistema pode falhar neste ponto e a recuperação do banca de dados estável ira conter efeitos de transações não compromissadas. Estes efeitos devem ser desfeitos pelo reinício para restaurar a banco de dados estável para seu estado compromissado imediatamente antes da falha.

Do mesmo modo, necessita refazer ("REDO") se ele permite uma transação compromissar antes que todos os valores que ela modificou tenham sido gravados no banco de dados estável (BERNSTEIN 1987). O sistema pode falhar neste ponto e a recuperação do banco de dadas estavel irá perder alguns efeitos das transações compromissadas. Estes efeitos devem seu refeitas pelo reinício para restaurar o banco de dados estável para seu estado compromissado imediatamente antes da falha.

Regulando a ordem do comprometimento das transações relativa a gravação de seus valores no banco de dados estável, um sub-sistema de recontrução pode controlar quando ele exige desfazer ("UNDO") ou refazer ("REDO"). Assim, Bernstein em (BERNSTEIN 1987) classifica os algoritmos de reconstrução em quatro categorias:

1) aqueles que necessitam tanto do "undo" como do "redo";

2) aqueles que necessitam do "undo" mas não do "redo";

3) aqueles que necessitam do "redo" mas não do "undo"; e

4) aqueles que não necessitam nem do "undo" nem do "redo".

No paradigma de Haerder e Reuter (HAERDER 1983) estas categorias correspondem a ROUBAR/NÃO-FORÇAR (1), ROUBAR/FORÇAR (2), NÃO-ROUBAR/NÃO-FORÇAR (3) e NÃO-ROUBAR/FORÇAR (4).

Na visão geral da proposta vimos que o COPPEREL-PC tem uma estatégia de subsituição de ROUBAR e que seu critério de manusear buffers FORCA a escrita e/ou propagação de todas as páginas modificadas durante o processamento de fim de transação. Se pegarmos as características do COPPEREL-PC e tentarmos enquadrar num desses algoritmos chegaremos naturalmente ao algoritmo "UNDO/NO-REDO".

Com a escolha desse algoritmo evitamos grandes modificações no código, uma vez que suas características principais são preservadas. Entretanto, na prática, e necessário guardar as imagens posteriores ("after images") visando a recuperação de falhas de meio.

U.6 - Divisão do Diário

Na literatura normalmente encontramos exemplos sobre diário como este sendo um arquivo único que contem as imagens anteriores e posteriores de informações modificadas no banco de dados.

No COPPEREL-PC optamos por separar o diário em dois arquivos distintos. O arquivo de imagens anteriores denominado de BI ("before image") e o arquivo de imagens posteriores denominado AI ("after image").

Temos três justificativa para esta divisão:

A primeira e decorrente do algoritmo "UNDO/NO-REDO" escolhido para a implementação. Este algoritmo separa de maneira clara e natural o objetivo de cada uma das partes. As imagens anteriores servem para reconstrução a partir de falhas de transação e/ou de sistema. As imagens posteriores servem tão somente para reconstrução a partir de falhas de meio.

A segunda justificativa é a preocupação com o espaço em disco. Com a separação dos diários podemos reutilizar o diário de "before image" a cada transação, ou seja, o tamanho máximo atingido pelo BI seria o número de paginas modificadas numa transação.

A terceira justificativa, a mais seria, é decorrente das características do ambiente (máquina e periféricos) no qual o sistema é executado. Sabemos que normalmente computadores pessoais possuem somente um disco rígido, e pensando nisso, resolvemos colocar o arquivo de imagens anteriores (BI) no mesmo dispositivo onde a base de dados será armazenada (provavelmente num disco rígido). Esta restrição nos deixa com a possibilidade de localizar o arquivo de imagens posteriores (AI) noutro disco rígido (hipótese mais difícil) ou num disco flexível, uma vez que para ele cumprir seu papel na reconstrução e necessário que se localize num dispositivo físico que não seja o da base de

dados. Porem, *e* importante enfatizar que discos flexíveis tem performance *e* capacidade sofríveis.

CAPITULO VI

A IMPLEMENTAÇÃO DA PROPOSTA

Na capítulo v definimos as técnicas de reconstrucão sue compõem o sub-sistema de reconstrução do COPPEREL-PC, apresentamos as justificativas para as escolhas e tambem fizemos uma breve descrição de como as técnicas escolhidas seriam implementadas. Neste capítulo, descreveremos detalhadamente como essas técnicas foram realmente implementadas, assim como, onde e como o código do COPPEREL-PC foi modificado.

VI.1 - Estratégia de Implementação

Num primeiro momento, era prioritário conhecer um pouco da estrura do COPPEREL-PC como um todo e também sobre o FORTRAN Microsoft (linguagem, compilador e linkeditor). Para isso, além da leitura dos referidos manuais (MICSOFT 1987a, 1987b) junto com informações sobre o produto dadas por terceiros, algumas simulações de modificações no código foram realizadas visando uma ambientação com os produtos em questão.

Num segundo momento, objetivando descobrir no sistema COPPEREL-PC os arquivos utilizados e os programas que efetivamente gravassem informações nesses arquivos, realiramas um levantamento em todos os programas fontes tendo como resultado uma lista de oito 18) arquivos e quinze (15) programas.

O arquivos são os seguintes:

BDTRABA - é a própria base de dados. Tem coma número de unidade de E/S padrão no sistema o um (1);

TUB - contém a lista de usuários das diversas bases de dados existentes. Tem como numero de unidade de E/S padrão no sistema o dois (2);

TBD - contém a lista de bases de dados disponíveis para utilização. Tem como número de unidade de E/S padrão no sistema o três (3);

SUPDADOS- contém os dados de entrada que inicializam o super-esquema CTUB e TBD). Tem como número de unidade de E/S padrão no sistema o quatro (4);

TRACE - relatório de saída do sistema. Tem como número de unidade de E/S padrão no sistema o cinco (5);

BSIRAP - contem a configuração inicial da base de dados. Tem como número de unidade de E/S padrão no sistema o oito (8);

ERROS - contém as mensagens de erros do sistema. Tem como número de unidade de E/S padrão no sistema o doze (12);

SOCORRO - contém a assistência interativa ("help") do sistema. Tem como número de unidade de E/S padrão no sistema o dezesseis (16);

A seguir listaremos os programas com uma breve definição do seu objetivo e indicando entre parênteses quais são os arquivos que eles gravam:

INIBDT.FOR - inicializa a base de dados (1)

BSTRAP.FOR - faz a configuração da base de dados (5 e 8)

SUPESQ.FOR - inicializa o sistema conforme o super-esquema (2 e 3)

ABREBD.FOR - abre uma base de dados (1)

CRIABD.FOR - cria uma base de dados (1, 2 e 3)

FECHBD.FOR - fecha uma base de dados aberta (1, 2 e 3)

BUSPAG.FOR - faz a leitura e gravação das paginas em disco (1)

LMPBUF.FOR - faz a descarga das paginas alteradas em memória (1)

INTBUB.FOR - insere em TUB um novo registro (2)

RMTBUB.FOR - remove de TUB uma autorimacão de um usuário (2)

INIERR.FOR - inicializa o arquivo de erros (1)

RWERRO.FOR - devolve a mensagem de erro para um código (2)

ERREXC.FOR - determina um erro de execução (5)

TABPAL1.FOR- pesquisa tabela de palavras reservadas (5)

Cam este levantamento reduzimos bastante o universo de trabalho, o que possibilitou um estudo minucioso de quais destes programas deveriam ser modificados. Neste estudo descobrimos que a base de dados (BDTRABA) só era modificada por seis desses programas: INIBDT, ABREBD, CRIABD, FECHABD, BUSPAG E LMPBUF. Verificamos também que o objetivo do programa INIBDT era de inicializar a base de dados (BDTRABA) quando da implantacão do produto, não fazendo parte do código do COPFEREL-PC (CPREL.EXE).

Em seguida, ficou claro tambem que somente dois programas gravavam páginas alteradas na base de dados. O BUSPAG e o LMPBUF. Os restantes simplesmente atualiravam o primeiro registro da base de dados (seção V.1.4.1) para garantir a coerência com a resto da base.

No caso do BUSPAG, a página somente é gravada na base de dados se a tabela de envelhecimento (seção V.1.4.2) assim o exigir, ou seja, se todas as paginas da tabela de envelhecimento tiverem sofrido alteração e uma nova pagina precisar ser gravada na tabela de envelhecimento, forçando assim a gravação de uma página na base pertencente a uma transação que ainda não foi compromissada.

O programa LMPBUF, diferentemente do BUÇPAG, percorre toda a tabela de envelhecimento no final de uma transação e descarrega para a base de dados todaç as páginas modificadas. Com isso, sabíamos que estes dois programas também precisavam gravar páginas nos arquivos de "before images" (BI) e de "after images" (AI).

Neste ponto, decidimos que a implementação seria dividida em duas partes. A primeira seria o trabalho de implementar o conceito de transação e o algoritmo de reconstrução para falhas de transação e falhas de sistema. A segunda parte cuidaria da implementação de reconstrução para falhas de meio.

U1.2 - Implementando Transacão e Reconstrução para Falhas de Transação/Sistema

Depois de localizar os programas que possivelmente seriam alterados, precisávamos definir como implementar o conceito de transação. Resolvemos o seguinte: Quando um comando de CRIAR BASE DE DADOS nome-da-base-de-dados (programa CRIABD) ou de ABRIR BASE DE DADOS nome-da-base-de-dados (programa ABREBD) fosse disparado, seria feito uma atualização no primeira registro da base (o que guarda informações sobre a base de dados). Para isso, criamos mais um campo neste primeiro registro denominado INICOK com a intenção de sabermos se uma transação terminou normal to valor de INICOK é igual a zero no reinício, mostrando que a

transação foi compromissada) ou se foi abortada por uma falha de sistema (o valor de INICOK é igual a um no reinício, mostrando que a transação não foi compromissada).

UI.2.1 - Reconstrução para Falhas de Transação

As falhas de transação, também conhecidas como pseudo-Talhas ou "rollback", não seguem este raciocíno, uma vez que são causadas propositalmente e de maneira controlada pelo usuário através do comando FECHAR BASE DE DADOS SEM AYUALIZAR. Esse comando leva o sistema a recuperar do BI as imagens anteriores de todas as páginas modificadas dentro desta transação e substitui-las na base de dados, fazendo com que a base de dados volte ao seu estado anterior ao inicio da transação.

Já sabemos que uma transação pode iniciar tanto pelo comando de CRIAR BASE DE DADOS nome-da-base-de-dados (programa CRIABD) como pelo comando de ABRIR BASE DE DADOS nome-da-base-de-dados (programa ABREBD) (seção V.2). A diferença básica para a reconstrução é que quando uma transação inicia através do programa CRIABD, a base de dados ainda não existe e devera ser criada, diferentemente de uma transação iniciada através do programa ABREBD, onde já existe uma base de dados. Essa explicação é pertinente pelo fato de que quando uma falha de transação e decidida pelo usuário, o programa FECHBD, sue é quem realiza a reconstrução de falha de transação, deve saber se a transação foi iniciada pelo programa ABREBD ou pelo CRIABD. A justificativa para isso é simples, se a transação iniciar por um programa CRIABD, o sistema não precisa desfazer a transação, uma ver que a base de dados ainda não existia efetivamente. Para fazer esta distinção, criamos uma variável de memória denominada "CRILA", que é atualizada no programa CRPABD com o valor i e no programa ABREBD com o valor 0, e e testada no programa FECHBD somente se ocorrer uma falha de transação.

A definição de um "lay-out" para o arquivo BI seguiu o seguinte raciocínio: já sabíamos que uma cópia da página modificada anterior a modificação deveria ser armazenada no BI. Havia também a necessidade de guardar qual a página da base de dados que estava sendo modificada para possibilitar uma ação de desfazer posterior . Por Último, com o fato do BI ser reutilizado a cada transação, havia também a necessidade de diferenciarmos cada transação das anteriores para sabermos quais registros do BI pertencem a Yltima transação. Assim, surgiu mais um campo para guardar a identificacão da transação.

O "lar-out" definido para o BI foi o seguinte:

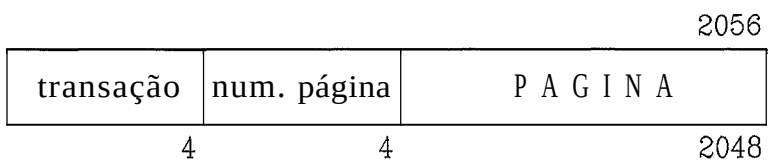


Figura VI.1 - O "lay-out" do arquivo BE.

Resolvemos que o acesso "direto" seria a melhor forma de organização para o arquivo BI. A justificativa para esta decisão é o fato de que um arquivo de acesso direto no FORTRAN pode ser tratado como um arquiva sequencial, mas o inverso não é verdadeiro. Assim, se no meio da implementação sentissemos a necessidade de mudar o acesso isso não impactaria muito no que já estivesse feito.

Por ocasião dos testes de falha de sistema surgiu a necessidade de identificarmos se os registros gravados pela Última vez no BI pertecem ou não a última transação. Isto porquê a Última transação pude não ter gravado nada no BI antes da falha.

O fato de que no momento da falha essa informação só se encontrava em memória volátil poderia levar um reinício a

recompor as paginas da transação anterior a ela equivocadamente. Assim, foi necessário criar mais um campo no primeiro registro da base de dados, denominado "ITIPBS", para armazenar o numero da ultima transação ativa no sistema. Esta informação é atualizada juntamente com o campo INICOK (seção VI.2) no início de cada transação, e testado somente no reiníco após uma falha de sistema.

VI.2.2 - Reconstrução para Falhas de Sistema

Uma falha de sistema, cama já mencionamos anteriormente, é detectada pelo "flag" INICOK=1 pertencente ao primeiro registro da base de dados (BDTRABA). Sabemos também que ao iniciar uma transação, tanto pelo CRIABD como pelo ABREBD, esse "flag" INICOK é inicialimado com o valor um (1), somente sendo atualizado para o valor zero (0) nu final do programa FECHBD (no final da transação), sinalizando com isso que a transação terminou normalmente. Assim, no inicio de uma transação a primeira coisa que ocorre é uma leitura no primeiro registro da base de dados para saber se a última transação realizada teve um fim normal ou não (testando o valor de INICOK). Se a Última transação não teve um fim normal, a mesma rotina de reconstrução usada no programa FECHBD é usada no inicio do programa ABREBD para recuperar do BI as imagens anteriores de todas as paginas modificadas dentro da Última transação, voltando a base de dados ao estado anterior ao Inicio da transação abortada. Como a rotina de reconstrução é igual para os dois tipos de Calhas (transação e sistema), optamos por escrever uma subrotina (RECUP1) em vez de duplicar o código.

VI.2.3 - 0 Problema do Super-Esquema

A essa altura da implementação, já sabiamos que a reconstrução no COPPEREL-PC envolveria mais do que

substituir paginas modificadas. Existia o problema de ocorrerem atualizações no super-esquema que também deveriam fazer parte da reconstrução. A explicação para isso e simples: durante uma transacão um usuário que seja devidamente autorizado pode autorizar e/ou revogar outros usuários a consultar e/ou atualizar uma determinada base de dados. O problema é que essas autorizações são feitas no super-esquema (seção V.1.2) e não na base de dados (BDTRABA). Assim, tornou-se imperativo que esses arquivos também fossem envolvidos na reconstrucão.

Este problema se localiza somente no programa CRIABD, pois só ele atualiza o super-esquema antes do final de uma transação. Com isso, tínhamos duas opções: a primeira seria também gravar no arquivo BI os registros anteriores às atualizações do TUB e TBD, o que implicaria em controlar mais tipos de registros, consumir mais área em disco, e mais código para gravação e recuperação. A segunda opção, a escolhida, seria regerar parcialmente estes dois arquivos (TUB e TBD) a partir do arquivo SUPDADOS, tal qual é feito quando da inicialização da COPPEREL-PC pelo administrador do sistema. Assim, pegamos o programa inicializador do supera-esquema (SUPESQ) e o enxugamos de maneira a só regerar os registros do tipo dois (2), que são os referentes a autorizações (MATTOSO 1985b). Esse programa (o SUQESQenxuto) virou a sub-rotina "SUPES1", complementando a reconstrução de falha de sistema.

VI.2.4 - O Problema do Espaço em Disco

Após a implementação de reconstrucão para Calhas de transação e de sistema, verificamos nos testes que o arquivo BI crescia rapidamente, pois para cada página modificada na base de dados, uma pagina anterior a modificação era gravada no BI, ou seja, se uma mesma página da base de dados fosse alterada "n" vezes, "n" cópias de imagens anteriores seriam gravadas no BI.

Visando melhorar o algoritmo, criamos um vetor na memória denominado "IUSE" de tantas posições quantas forem as paginas reservadas para a base de dados. O raciocinio é simples. No início da transação (CRIABD e ABREBD) esse vetor e inicializado com zeros. A partir dai, para cada página alterada na base de dados, antes de gravar sua imagem anterior no BI e feita uma consulta no vetor IUSE na posição referente a pagina, com isso temos duas possibilidades:

a) se seu conteúdo for zero significa que e a primeira vez que a pagina está sendo modificada, então, atualiza-se no vrtor IUSE a posição correspondente à pagina com o próprio número da pagina e grava-se a página anterior no BI.

b) se seu conteúdo for diferente de zero, ou seja, a pagina já foi modificada anteriormente, não se arava nada no BI. Assim, no final da transação, só gravaremos no BI uma imagem anterior para cada página modificada, mesmo que ela seja modificada "n" veres, o que é muito provável de acontecer numa transação pelo conceito de "localidade".

Com essa modificação no algoritmo verificamos nos testes que o espaço necessário por uma transação para o BI diminuiu substancialmente.

Esse mesmo raciocinio foi mais tarde empregado para o AI com uma diferença: em ver de consultar o vetor IUSE para evitar gravar a mesma pagina, o propósito e saber se a página já foi modificada e consequentemente gravada no AI dentro da mesma transação. Em caso positiva grava-se a página modificada em cima da página já existente no AI, aproveitando novamente o conceito de "localidade" e economizando espaço.

UI.3 - Implementando Reconstrução para Falhas de Meio

Essa segunda parte da implementação, apesar de tão importante quanto a primeira, foi mais simples de ser implementada. São dois os motivos para essa afirmação:

a) na implementação da parte de reconstrução para falhas de transação/sistema (seção VI.2) já havíamos determinado os pontos onde as gravações deveriam ocorrer no diário (BI), o que quase na sua totalidade são os mesmos pontos de gravação do AI.

b) o algorítmo de reconstrução de falhas de meio não faz parte do código do COPPEREL-PC (CPREL.EXE), sendo um programa à parte que só será chamado no caso de ocorrer um evento desta natureza.

Este segundo motivo facilitou em muito a implementação, pois uma vez de posse do arquivo AI com as informações necessárias gravadas e da Última descarga, isolamos o problema e consequentemente facilitamos os testes.

O programa para recuperação de falha de meio foi denominado "FAMEIO" e deve ser executado logo após o procedimento de restaurar o banco de dados (Última descarga) atraves de um utilitário.

O "lay-out" definido para o AI foi o seguinte:

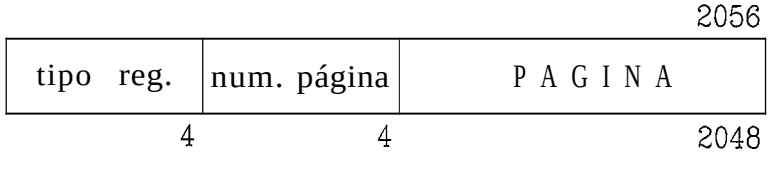


Figura VI.2 - O "lar-out" do arquivo AI.

Apesar do "lay-out" do AI ser igual ao do BI, o conteúdo do primeiro campo é diferente. No BI o primeiro campo é utilizado para guardar o número da transação (seção VI.2),

enquanto que no arquivo **AI** o primeiro campo serve para determinar o tipo de registro gravado.

Essa diferença se deve ao fato de que no **BI** só necessitamos guardar as páginas da base de dados (BDTRABA), pois a reconstrução do super-esquema ficou a cargo da rotina "SUPES1" (seção VI.2.3). Já no arquivo AI, necessitamos guardar diferentes tipos de registros com diferentes propósitos como vemos a seguir:

1- primeiro registro do **BDTRABA.** Contem informações sobre a base de dados;

2- é a própria pagina modificada do **BDTRABA;**

3- primeiro resistro do arquivo **TUB.** Contém informações sobre autorizações na base de dados;

4- registro normal do TUB contendo informações sobre os usuários e suas autorizações;

5- registro do arquivo **TBD.** Contém informacões sobre as base de dados;

6- registro de comprometimento. Indica para o **AI** que a transação terminou normalmente e deve ser recuperada no caso de uma falha de meio.

Além destes registros, o arquivo **AI** possui um primeiro registro que guarda o número do próximo registro a ser gravado nele. O propósito e saber qual o próximo registro a ser gravado no AI no inicio de uma transação, o que nos da duas possibilidades. A primeira e guardar essa informação num mio não volátil; a segunda possibilidade é ler sequencialmente o arquivo **AI** contando os registros, o sue se tornaria cada vez mais dispendioso a medida sue o arquivo fosse enchendo.

Na implementação foram usados os dois métodos. No caso de processamento normal do sistema, o próximo registro a ser gravado no AI e encontrado no primeiro registro do próprio AI. Porém, no caso de acontecer uma falha de sistema, essa informação estaria desatualirada, uma vez que a atualização deste registro só é feita no final da transação. Assim, quando o sistema se recupera de uma falha de sistema, ele lê sequencialmente o arquivo AI para conseguir esta informação, o que e probabilisticamente difícil de acontecer.

É importante enfatizar que os algoritmos implementados no sub-sistema de reconstrução do COPPEREL-PC para falhas de sistema e meio são "idempotentes" (seção II.2.3).

A seguir, explicaremos o algoritmo do programa "FAMEIO":

a) uma leitura sequencial é realizada no arquivo AI. Quando um registro do tipo i (um) for encontrado e nele o "flag" INICOK possuir o valor 1 (um) indicando que uma transação foi iniciada, guarda-se o número deste registro prevendo um eventual retorno a ele para efetuar a reconstrução.

b) quando um registro do tipo 6 (seis) for encontrado, significa que a transação terminou normalmente e deve ser recuperada. Neste caso inicia-se uma nova leitura a partir do número do registro salvo (que indica o início da transação) e regravamos os arquivas (BDTRABA, TBD e TUB) conforme os tipos de registros encontrados nesta transação.

c) no caso de encontrarmos um outro registro tipo 1 (um) com "flag" INICOK=1, indicando o início de outra transação **antes** de encontrarmos um registro **tipo** 6 (seis), significa que a transação anterior não terminou normalmente (houve uma falha) e deve ser desprezada.

Assim, quando chegarmos no final do arquivo AI, todas os arquivos (BDTRABA, TBD e TUB) estarão com seus conteúdos recuperados.

V1.4 - A Flexibilidade do Sub-Sistema de Reconstrução

Objetivando tornar o sub-sistema de reconstrução o mais flexível possível, decidimos fazer com que todas aç informações referentes à ele estivessem localizadas num arquivo (LOGDADOS) e pudessem ser alteradas pelo administrador do sistema sem que fosse necessário alterar o código do **COPPEREL-PC.** A idéia seguiu **a** rotina já existente para o super-esquema, onde o administrador do sistema através de um arquivo (SUPDADOS) monta o super-esquema do COPPEREL-PC. Assim, quando da implantação do sistema, o administrador fica inteiramente a vontade para decidir:

a) não usar o sub-sistema de reconstrução;

b) usar somente a parte do sub-sistema de recontrução referente à falhas de transação/sistema, e qual será o espaço utilizado para arquivo BI;

c) usar somente a parte do sub-sistema de recontrução referente a falhas de meio, e qual será o espaço utilizado rara arquivo AI;

d) usar o sub-sistema de recontrução completo, o que significa recuperação para falhas de transação/sistema e de meio, bem como os espaços utilizados para os arquivos BI e AI;

e) sobre o dispositivo onde se localizara os arquivos BI e AI se forem utilizados;

f) sobre o percentual de utilização que os arquivos BI e AI deverão ter se forem utilizados.

A seguir, falaremos sobre o "lay-out" do arquivo "LOGDADOS" e explicaremos cada decisão que o administrador pode tomar referente ao sub-sistema de reconstrução.

O "lay-out" do arquivo LOGDADOS é o seguinte:

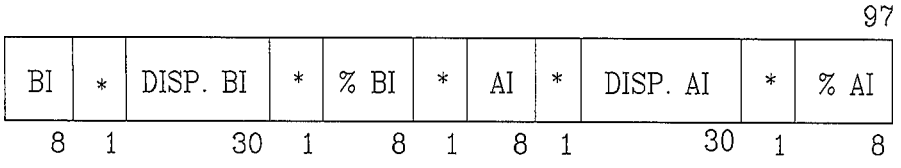


Figura VI.3 - O "lay-out" do arquivo LOGDADOS.

onde:

BI - número de páginas de 2056 bytes reservadas para o BI

disp BI- dispositivo onde o BI residira

% BI - percentual de utilização do BI

AI - número de páginas de 2056 bytes reservadas para o AI

disp AI- dispositivo onde o AI residira

% AI - percentual de utilização do AI

* - delimitador ("filler")

Para cada decisão do administrador do banco de dados com relação ao sub-sistema de reconstrução existe um procedimento adequado como vemos a seguir:

a) não usar o sub-sistema de recontrução. Deve-se modificar no arquivo "LOGDADOS" os campos "BI" e "AI" com o valor

0 (zero). Os outros campos podem continuar tendo informações.

b) usar somente a parte do sub-sistema de recontrução referente à falhas de transação/sistema. Deve-se modificar no arquivo "LOGDADOS" o campo "BI" com o número de páginas reservadas para o arquivo BI, o campo "disp BI" com a localização do arquivo BI (ex: C:\MICSOFT\BI), o campo "% BI" com o percentual de utilização do arquivo e o campo "AI" com o valor 0 (zero). Os outros campos podem continuar tendo informacões.

c) usar somente a parte do sub-sistema de recontrução referente à falhas de meio. Deve-se modificar no arquivo "LOGDADOS" o campo "AI" com o número de paginas reservadas para o arquivo AI, o campo "disp AI" com a localização do arquivo AI (ex: B:\MICSOFT\AI), o campo "% AI" com o percentual de utilização do arquivo e o campo "BI" com o valor 0 (zero). Os outros campos podem continuar tendo informações.

d) usar o sub-sistema de recontrução completo, o que significa recuperação para falhas de transação/sistema e de meio. Deve-se modificar no arquivo "LOGDADOS" o campo "BI" com o número de páginas reservadas para o arquivo BI, o campo "disp BI" com a localização do arquivo BI (ex: C:\MICSOFT\BI), o campo "% BI" com o percentual de utilização do arquivo, o campa "AI" c o número de páginas reservadas para o arquivo AI, o campo "disp AI" com a localização do arquivo AI (ex: B:\MICSOFT\AI) e o campo "% AI" com o percentual de utilização do arquivo.

A seguir, explicaremos como é feita a monitoração dos espaços reservados para os arquivos BI e AI, uma ver que espaço em disco é uma preocupação constante do sub-sistema de reconstrução.

Para permitir esse monitoramento, criamos um programa chamado de "INILOG" que inicializa os arquivos BI e **AI** com o tamanho informado pelo administrador do sistema no arquivo LOGDADOS. Esta inicialização e semelhante a já existente para a própria base de dados (prograna INIBDT), e sua principal função e limpar e reservar as áreas que serão usadas pelos arquivos BI e **AI**.

Durante o processamento normal do sistema, depois de cada gravação feita nos arquivo BI e AI, um teste é realizado para verificar se o numero de paginas alocadas para os arquivos BI e **AI** já atingiu o seu limite de utilização. Esse limite de utilização é determinado pela multiplicação do número de páginas alocadas rara o BI e para o AI com seus respectivos percentuais de utilização. Por exemplo, se o administrador do sistema decidir que o BI terá 200 páginas e seu percentual de utilização sera de 90%, isto significa que seu limite de utilização é de 180 páginas e que ele tem uma margem de segurança de 20 páginas para tentar finalizar seu trabalho.

Se em algum momento os arquivos atingirem seus limites de utilização, o sistema utiliza as seguintes telas (figura VI.4 e VI.5) para informar o usuário do ocorrido, qual o arquivo com problema de espaço, quantas paginas restam (para que o usuário avalie se há condições de prosseguir), e quais seriam suas opções.

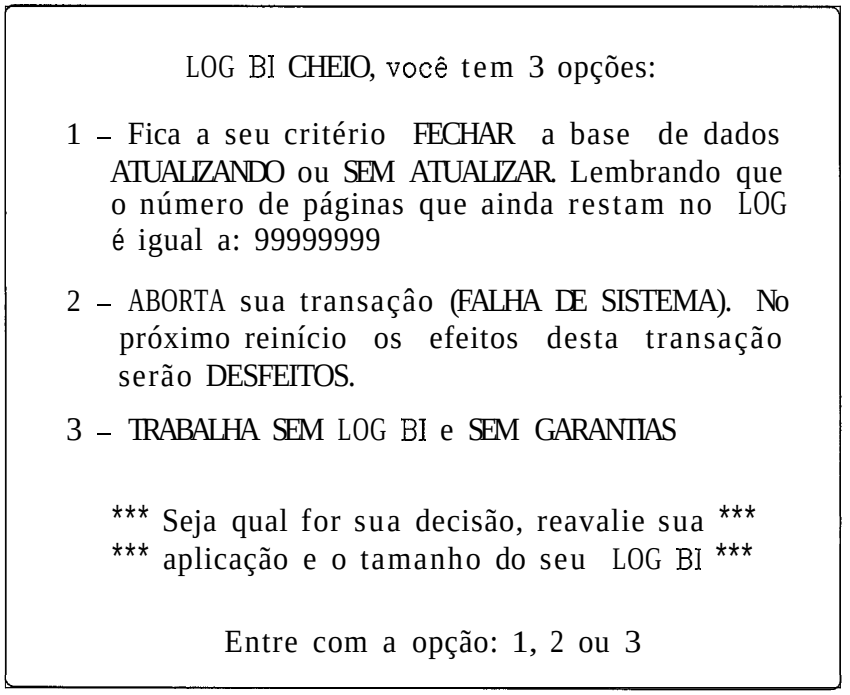


Figura VI.9 – Tela de opções do arquivo BI

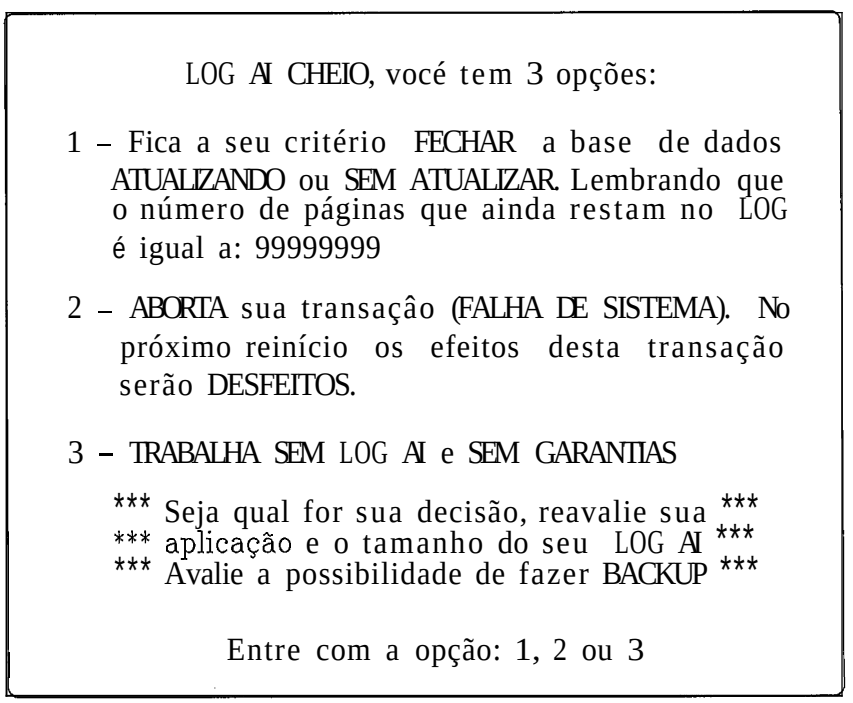


Figura VI.5 - Tela de opções do arquivo BI.

Caso o usuário decida continuar sua transação (opção 1), o sistema continuará dando a mensagem a cada gravação no BI

ou AI ate que o número de paginas restantes seja igual a zero, quando então o sistema não permitirá mais que o usuário use a primeira opção e o informará disso.

A opção dois permite o usuário abortar sua transação, reavaliar sua aplicação e/ou o tamanho dos seus arquivos BI e AI, e reinicializar o sistema com a garantia do mesmo "desfazer" os efeitos da transação que ele abortou.

A terceira opção, é mais um aspecto da Clexibilidade do sub-sistema de reconstrução que deve ser ressaltado. Com ela, o usuario pode decidir por sua própria conta e risco, passar a não trabalhar mais com o(s) arquivo(s) em questão no meio de uma transação que exceda os limites determinados pelo administrador para o sub-sistema de reconstrução.

CAPÍTULO VII

CONCLUSÃO

No primeiro capítulo deste trabalho procuramos mostrar a utilidade dos sistemas gerenciadores de banco de dados, enfatizando a importância da informação no mundo atual. Evidenciamos também a necessidade da segurança, da integridade e da disponibilidade da informação para que ela seja realmente útil. Com isso, podemos concluir que um sub-sistema de reconstrução deve ser parte integrante de um ÇGBD.

VII.1 - Avaliação do Trabalho

A proposta de um sub-sistema de reconstrução para o COPPEREL-PC descrita no capítulo V, foi integralmente implementada, o que significa que atualmente o COPPEREL-PC possui o conceito de transação e também um sub-sistema de reconstrução capaz de recuperar falhas de transação e de sistema de maneira totalmente transparente ao usuário e também de falhas de meio. E importante salientar que a característica de "idempotência" (seção II.1.2) também está presente no sub-sistema de reconstrução implementado.

O objetivo de aumentar o menos possível o código do COPPEREL-PC (seção V.3) foi alcançado. O çub-sistema de reconstrução implementado tem aproximadamente 30 K, ou seja, aumentou em 5% o código inicial do COPPEREL-PC que era de 600 K.

O tempo de execução do CQPPEREL-PC com o sub-sistema de reconstrucão não teve uma variação significativa. Entretanto, devemos ressaltar que se um dos arquivos, BI e

AI, forem alocados num dispositivo de disco flexível, o tempo de execução aumenta drasticamente.

Finalmente, a implementação do sub-sistema de reconstrução no COPPEREL-PC provocou um aumento de confiabilidade de 0 para 100%, aumentando consequentemente o interesse na sua utilização.

O apêndice contém resultados de simulações e testes do sub-sistema de reconstrução do COPPEREL-PC.

VII.2 - Contribuição do Trabalho

Dividimos em quatro partes distintãs esse tópico de contribuição, sendo eles: revisão bibliográfica, implementação do sub-sistema de reconstrucão, maneira como foi implementado e experiência transmitida.

a) Quanto a revisão bibliográfica, podemos dizer que apesar do assunto ser bastante conhecido e estudado, houve neste trabalho uma coletânea dos principais artigos sobre o assunto, abordando temas ligados a reconstrução como transação, falhas, controle de concorrência e "bufferização". No capitulo IV apresentamos as técnicas mais conhecidas e empregadas de reconstrução e, além disso, no capitulo 5 procuramos mostrar a parte prática do tema, ou seja, sua aplicabilidade através de exemplos e também a relação de atualidade do tema com as tendências da área de banco de dados para um futuro próximo, que são os sistemas de banco de dados orientados a objrto (SBDOO) e os sistemas de banco de dados distribuídos (SBDDs).

b) Quanto à própria implementação do sub-sistema de reconstrucão no COPPEREL-PC que atendeu um importante requisito para uma melhor utilização do sistema, uma vez que introduziu no mesmo o conceito de transação e

garantiu ao seu usuário a integridade e a disponibilidade necessária para usa-lo.

c) Quanto a maneira que o sub-sistema de reconstrução foi implementado, respeitando o aspecto da portabilidade e incorporando a ele uma enorme flexibilidade de uso, permitindo com isso que o usuário decida quando (se usara ou. não o sub-sistema de reconstrução), quanto (qual vai ser a area necessária), onde (em qual dispositivo os arquivos irão residir) e qual (pode decidir usar só reconstrução de falhas de transação/sistema, só a de falhas de meio, os dois tipos de reconstrução ou nenhum) recurso do sub-sistema de reconstrução utilizar sem modificar uma linha de código do COPPEREL-PC.

d) Quanto a experiência a transmitir, uma ver que o sub-sistema de reconstrução foi totalmente implementado e encontra-se disponível para qualquer pessoa interessada no assunto.

VII.3 - Sugestões para Futuros Trabalhos

Neste tópico, entendemos que devem ser consideradas somente as sugestões que estejam ligadas diretamente a esse trabalho. Assim, registramos três sugestões.

A primeira e mais imediata seria melhorar o próprio sub-sistema de reconstrução no que diz respeito a area de armazenamento necessária. Isto poderia ser conseguido através da mudança de "diário físico a nível de paginas" para um tipo que consuma menos espaço. O mais próximo das características atuais do COPPEREL-PC e do atual sub-sistema de reconstrução, e portanto com menos impacto para implementar seria o "diário de transição física a nível de página" (seção III.6.3.1 e **V.4.2).** Uma segunda alternativa, embora aparentemente mais trabalhosa, seria

implementar um "diário lógico a nível de registro", o que implicaria em todas as considerações feitas nas seções III.6.3.1 e U.4.2.

A segunda sugestão é revisar o código do COPPEREL-PC e avaliar a possibilidade de reescrevê-lo numa linguagem mais poderosa r atual.

A terceira e Última sugestão e melhorar e atualizar a documentação do sistema como um todo. Esta sugestão, apesar de aparentemente menos importante deve ter prioridade, uma ver que varias modificações já foram feitas no sistema (parte interativa do sistema, interface para linguagens hospedeiras, incorporação de campos longos, assistência interativa e agora a parte de reconstrução) e sua documentação se encontra dispersa nos respectivos trabalhos r teses que trataram desses assuntos.

REFERÊNCIAS BIBLIOGRÁFICAS

ADIBA 1980 - Adiba, M. E. e Lindsay, B. G., "Databases Snapshots", IBM Research Report: RJ2772, 1980.

AGHILI 1982 - Aghili, H., Severance, D., "A Pratical Guide to the Design of Differential Files for Recovery of On-Line Databases", ACM Transactions on Database Systems, Vol. 7, No. 4, december 1982.

AGRAWAL 1985 - Agrawal, R., Dewitt, D., "Integrated Concurrency Control and Recovery Mechanisms: Design and Performance Evaluation", ACM Transactions on Database Systems, Vol. 10, No. 4, december 1985.

ANDERSON 1978 - Anderson, T., Lee, P. A. e Shrivastava, S.K.,"A Model of Recoverability in Muitilevel Systems", IEEE Transactions on Software Engeneering, SE-4, No. 6, nov 1978, pag. 486 - 494.

ANDERSON 1986 - Anderson, T. & Ecklund, E. F. & Maier, D., "PROTEUS: Objectifying the MS User Interface", em OODS 1986.

ASTRAHAN 1976 - Astrahan, M. M., Blasgen, M. W., Chamberlin, D. D., Eswaran, K. P., Gray, J. N., Griffiths, P. P., King, W. F., Lorie, R. A., McJones, P. R., Mehl, J. W., Putzolu, G. R., Traiger, I. L., Wade, B. W., e Watson, V., "System R: Relational Approach to Database Management", ACM Transactions on Database Systems, Vol. 1, No. 2, June 1976, Pages 97-137.

BERNSTEIN 1987 - Bernstein, P.A., Hadzilacos, V., Goodman, N., "Concurrency Control and Recoverr in Database Systems", Addison-Wesley, 9987.

BJORK 1975 - Bjork, L. A., "Generalized Audit Trail Requirements and Concept for Data Base Applications", IBM System Journal, 14, 3, 1975, pag. 229 - 245.

BLUM 1988 - Blum, H., Gonçalves, L. A., Rossato, M. A., "Bancos de Dados Orientados a Objetos", Relatório Técnico do Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, ES-172/88.

BLUM 1989 - Blum, H., Gonçalves, L. A., Rossato, M. A., "Proposta de Desenvolvimento de um Protótipo de Sistema de Gerência de Banco de Dados Orientado a Objetos", Relatório Técnico do Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, ES-218/89.

BLUM 1989 - Blum, H., Gonçalves, L. A., Rossato, M. A., "Implementação de um Protótipo de Sistema de Gerência de Banco de Dados Orientado a Objetos", Relatório Técnico do Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, ES-219/89.

CASANOVA 1985 - Casanova, M.A., Moura, A.V., "Princípios de Sistemas de Gerência de Bancos de Dados Distribuídos", Campus, 1985.

CCRI 1984 - Ceri, S., Pelagatti, G., "Distributed Databases Principles and Systems", McGraw-Hill, Inc., 1984.

CHANDY 1975 - Chandy, K. M., Brown, J. C., Dissley, C. W. e Uhrig, W. R.,"Analytic Model for Rollback and Recovery Strategies in Data Base Systems", IEEE Transactions on Software Engeneering, SE-1, No. 1, march, 1975, pag. 100 - 110.

COPELAND 1984 - Copeland, E. & Maier, D., "Making SmallTalk a Database System", em SIGMOD 1984.

CRUS 1984 - Crus, R. A., "Data Recovery in IBM Database 2", IBM Srstem Journal, Vol. 23, No. 2, 1984.

DATE 1986 - Date, C.J., "An Introduction to Database Systems", Vol I, Fourth Edition, Addison-Wesley, 1986.

DATE 1987 - Date, C.J., "A Guide to INGRES", Addison-Wesley, 1987.

DATE 1988 - Date, C.J., "Banco de Dados: Tópicos Avançados", Rio de Janeiro, Campus, 1988. Tradução de: An introduction to database systems - Vol II, Addison-Wesley, 1983.

DITTRICH 1986 - Dittrich, K. R., "Object-Oriented Database Systems - A Workshop Report", Proceedings of the 5th E-R Conference 1986, Dijon, North Holland Publisher Group.

EFFELSBERG 1984 - Effelsberg, W., Haerder, T., "Principles of Database Buffer Management", ACM Transactions on Database Systems, Vol. 9, No. 4, december 1984.

GONÇALVES 1988 - Goncalves, L. A. e Mattoso, L. Q. M., "Análise de Sistemas de Gerência de Objetos (SGO) e a Proposta da Arquitetura SGO do TABA", Relatório Técnico do Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, ES-181/88.

GRAY 1978 - Gray, J.,"Notes on Data Base Operating Systems", IBM Research Report: RJ2188, IBM Research Laboratory, San Jose, California, 1978.

GRAY 1980 - Gray, J.,"A Transaction Model", IBM Research Rerort: RJ2895, IBM Research Laboratory, San Jose, California, 1980.

GRAY 1981a - Gray, J.,"The Transaction Concept: Virtues and Limitations", in Proceedings of the 7th International Conference on VLDB (Cannes, France, Sept. 9 - 11). ACM New York, pag. 144 - 154.

GRAY 1981b - Gray, J., McJones, P., Blasgen, M., Lindsay, B., Lorie, R., Price, T., Putzolu, F., Traiger, I., "The

Recovery Manager of the System R Database Manager", Computing Survers, Vol. 13, No. 2, june 1981.

HAEDER 1983 - Haeder, T., Reuter, A., "Principles of Transaction-Oriented Database Recovery", Computing Surveys, Vol. 15, No. 4, december 1983.

WCILCR 1987 - Heiler, S., Dayal, U., Orenstein, J., Radke-Sproull, S., "An Object-Oriented Approach to Data Management. Why design Databases need it?", 24th ACM/IEEE Design Automation Conference Proceedings 1987, Miami Beach.

HUDSON 1986 - Hudson, S. E. & King, R., "CACTIS: A Database System for Specifying Functionally-Defined Data", em OODS 1986.

HUDSON 1987 - Hudson, S. E. & King, R., "Object-Oriented Database Support for Software Enviranments", em SIGMOD 1987.

HUDSON 1988 - Hudson, S. E. 8 King, R., "The Cactis Project: Database Support for Software Environments", IEEE Transactions on Software Engineering. V. 14, N. 6, June 1988.

IBM 1388 - IBM Corporation, "Introduction to Distributed Relational Data", International Technical Support Center, Santa Teresa, GG24-3200-00, 1988.

JONEÇ 1987 - Jones, J.A., Databases in Theory and Practice", TAB Books Inc., 1987.

KIM 1979 - Kim, W., "Relational Database Systems", Computing Surveys, Vol. 11, No. 3, september 1979.

KING 1987 - King, R. & Novak, M., "Freefrom: A user-adaptable form management system", em VLDB 1987.

KOHLER 1981 - Kohlrr, W. H., "A Survey of Techniques for Synchronization and Recovery in Decentralired Computer Systems", Computing Survers, Vol. 13, No. 2, june 1981

KORTH 1989 - Korth, H.F., Silberschatr, A., "Sistemas de Bancos de Dados", McGraw-Hill, 1989.

LANDES 1980 - Landes, O., Menascé, D., "Um Estudo Sobre Ténicas de Recuperação de Erros em Bancos de Dados", Departamento de Informática - PUC, monografia em ciência e computação No. 5/80, março 1980.

LOHMAN 1977 - Lohman, G. M. e Muckstadt, J. A., "Optimal Policy for Batch Operations: Backup, checkpointing, reorganization, and updating", ACM Transactions on Database Systems, Vol. 2, No. 3, september 1977, pag. 209 - 222.

LORIE 1977 - Lorie, R., "Phisical Integrity in a Large Segmented Database", ACM Transactions on Database Systems, Vol. 2, No. I, march 1977.

MAIER 1985 - Maier, David & Otis, allen & Purdy, Allan, "Object-Oriented Database Development at Servio Logic", IEEE Transactions on Software Engineering, V. 18, N. 4, December 1985.

MAIER 1986a - Maier, David & Stein, Jacob & Otis, Allen & Purdy,Alan, "Development of an Object-Oriented DBHS", em OOPSLA 1986.

MAIER 1986b - Maier, David & Stein, Jacob, "Indexing in an Object-Oriented DBMS", em OODS 1986.

MATTOSO 1985a - Mattoso, M. L. Q., Zakimi, B. M., "Manual de Instalação do SGBD COPPEREL", relatório técnico ES-63/85, COPPE/UFRJ, 1985.

MATTOSO 1985b - Mattoso, M. L. Q., Zakimi, B. M., "Manual do Administrador das Bases de Dados do SGBD COPPEREL", relatório técnico ES-68/85, COPPE/UFRJ, 1985.

MATTOSO 1987 - Mattoso, M.L.Q., "A Incorporação de Ferramentas de Apoio aos Usuários do SGBD COPPEREL", dissertação de Mestrado, Prog. de Sistemas - COPPE/UFRJ, maio 1987.

MICROSOFT 1987a - Microsoft FORTRAN Optimizing Compiler for ihe MS-DOS Operating System, "User's Guide", Microsoft Corporation, 1987.

MLCHOSOFT 1987b - Microsoft FORTRAN Optimizing Compiler for the MS-DOS Operating System, "Language Reference", Microsoft Corporation, 1987.

OODS 1986 - Proceedings of 1986 International Workshop on Object-Oriented Database Systems, September 23 -26, 1986, Asilomar, California.

OOPSLA 1986 - Object Oriented Programming Systems, languagens and appíications Conference Proceedings, September 29 - October 2, 1986, Portland, Oregon, SIGPLAN Notices V 21, N.11, November 1986.

OOPSLA 1987a - Object Oriented Programming Systems, languagens and applications Conference Proceedings, October 4 - 8, 1987, Orlando, Florida, SIGPLAN Notices V 22, N.12, December 2987.

PALERMO 1985 - Paleumo, L.I., "Proposta de um Controlador de Concorrência para um SGBD Relaciona1 Centralizado", dissertação de Mestrado, Prog. de Sistemas - COPPE/UFRJ, outubro 1985.

PURDY 1987 - Purdy, A. & Schchardt, B. & Maier, D., "Integrating an object server with other worlds", ACM

Transactions on Office Information Systems, V 5, N. 1, January 1987.

REUTCR 1980 - Reuter, A., "A Fast Transaction-Oriented Logging Scheme for UNDO Recovery", IEEE Transactions on Software Engineering, Vol. SE-6, No. 4, july 1980.

REUTER 1984 - Reuter, A., "Performance Analysis of Recovery Techniques", ACM Transactions on Database Systems, Uol. 9, No. 4, december 1984.

ROCHA 1988 - Rocha, A.R.C. e Souza, J.M., "TABA: Uma Estação de Trabalho para o Engenheiro de Software", relatório técnico COPPE/UFRJ ES-145/88.

ROWE 1986 - Rowe, L. A., "A Shared Object Hierarchy", em OODS 9986.

ROWE 1987 - Rowe, L. A. & Stonebraker, M., "The Postgres Data Model", em ULDB 1987.

SEVERANCE 1976 - Severance, D., Lohman, G., "Differential Files: Their Application to the Maintenance of Large Databases", ACM Transactions on Database Systems, Vol. 1, No. 3, september 1976.

SIGMOD 1984 - Proceedings of Annual Meeting Database Week, Boston, USA, June 18 - 21, 1984, SIGMOD Record V 14, N.2.

SIGMOD 1986 - Proceedings of ACM-SIGMOD 1986 International Conference on Management of Data, Washington, D. C., May 1986, SIGMOD Record.

SIGMOD 1987 - Proceedings of Association for Computing Machinery Special Interest Group on Management of Data, 1987 Annual Conference, San Francisco, May 27-29, 1987, SIGMOD Record V 16, N. 3, December 1987.

SOUZA **1987** - Souza, J. M., Mattoso, M. L. Q., "O Projeto Engenharia de Dados", Anais do II Simpósio Brasileiro de banco de Dados, Porto Alegre, março, 1987.

STANKOVIC **1984** - Stankovic, **J.** A., "A Perspective on Distributed Computer Systems", IEEE Transactions on Computers, Vol. C-33, No. **12,** december **9984.**

STEIN **1987** - Stein, Jacob & Penney, D. Jason, "Class Modification in the GemStone Object-Oriented DBMS", em OOPSLA **1987.**

STONEBRAKER 1976 - Stonebraker, M., Wong, E., Kreps, P., "The Design and Implementation of INGRES", ACM Transactions on Database Systems, Vol. 1, No. 3, september 1976.

STONEBRAKER **i980** - Stonebraker, M., "Retrospection on a Database System", ACM Transactions on Database Systems, Vol. 5, No. 2, June 1980.

STONEBRAKER **1984** - Stonebraker, .M. et al, "Quel a5 a data type", em SIGMOD **1984.**

STONEBRAKER 1986a - Stonebraker, M. & Rowe, L.A., "The Design of Postgres", em SIGMOD 1986.

SIONCBRAKER 1986b - Stonebraker, M., " Object Management in Postgres Using Procedures", em OODS 1986.

STONEBRAKER **1997** - Stonebraker, M., "The Design of the Postgres Storage System", em VLDB **1987.**

TROTTA **1989** - Trotta, C.N.F., "Extensões no SGBD COPPEREL para Aplicações não Convencionais", dissertação de Mestrado, Prog. de Sistemas - COPPE/UFRJ, março **1989.**

UERHOFSTAD **1977** - Verhofstad, J.S.M., "Recovery and Crash Resistance in a Filing System", in Proceedings of **1977**

ACM SIGMOD, International Conference on Management of Data, ACM, New York, pp. 158 - 167.

VERHOFSTAD 1978 - Verhofstad, J.S.M., "Recovery Technisues For Database Systems", Computing Survers, Vol. 10, No. 2, june 1978.

VLDB 1987 - Proceedings of the Thirteenth International Conference on Very Large Data Bases, Brighton, England, September 1 - 4, 1987.

ZAKIMI 1982a - Zakimi, B. M. et all., "COPPEREL: Sistema de gerência de base de dados da COPPE baseado em álgebra relacional", Anais do IX SEMISH, julho 1982.

ZAKIMI 1982b - Zakimi, B. M., Mattoso, M. L. Q., Prazeres, M. J., "Manual do usuário da LOSEREL (linguagem de Operação do SGBD COPPEREL)", relatório técnico ES-62/85, COPPE/UFRJ, 1985.

APÊNDICE

SIMULAÇÃO E TESTES DO SUB-SISTEMA DE RECONSTRUÇÃO DO COPPEREL-PC

Este apêndice contém as simulações de falha de transação ("rollback"), falha de sistema e falha de meio de armazenamento. Com o objetivo de evitar a repetição dos procedimentos e agilizar os testes, criamos alguns arquivos auxiliares descritos a seguír.

1 - Arquivas Auxiliares

a) CRIATESE

```
abrir sessao para us aluno/coppe;
criar base de dados BDTESE em F1 e F2 para 50 regs por arq;
autorizar usuario aluno/coppe para ler tab_entidades;
autorizar usuario aluno/coppe para remover
tab_autorizacoes;
autorizar usuario aluno/coppe para ler tab_atributos;
autorizar usuario aluno/coppe para ler tab_derivacoes;
criar arquivo tesl com 28 regs de 2 atributos:
codigo:i(2),
nome:a(10)
com chave: codigo;
most tab_autorizacoes;
fechar base de dados;
fechar sessao;
enc
```

b) CRIASEM

```
abrir sessao para us aluno/coppe;
abrir base de dados BDTESE;
```

```
criar arquivo tes2 com 20 registros de 2 atributos:
CODP   : I(2),
NONEP  : A(10)
com chave : codp e COM indice sobre : nomep;
criar arquivo tes3 COM 20 registros de 2 atributos:
CODP   : I(2),
NOMEP  : A(10)
com chave : codp e com indice sobre : nomep;
criar arquivo tes4 com 20 registros de 2 atributos:
COBP   : I(2),
NOMEP  : A(10)
com chave : codp e com indice sobre : nomep;
criar arquivo tes5 com 20 registros de 2 atributos:
CODP   : I(2),
NOMEP  : A(10)
com chave : codp e COM indice sobre : nomep;
criar arquivo tes6 com 20 registros de 2 atributos:
CODP   : I(2),
NOMEP  : A(10)
com chave : codp e com indice sobre : nomep;
criar arquivo tes7 com 20 registros de 2 atributos:
CODP   : I(2),
NOMEP  : A(10)
com chave : codp e COM indice sobre : nomep;
criar arquivo tes8 com 20 resistros de 2 atributos:
CODP   : I(2),
NOMEP  : A(10)
com chave : codp e COM indice sobre : nomep;
criar arquivo tes? com 28 registros de 2 atributos:
CODP   : I(2),
NOMEP  : A(10)
com chave : codp e com indice sobre : namep;
criar arquivo tes10 com 20 registros de 2 atributos:
CODP   : I(2),
NOMEP  : A(10)
com chave : codp e com indice sobre : nomep;
most tab_autorizacoes;
copiar tes1 em tes2;
copiar tesl em tes3;
```

```
copiar tes1 em tes4;
copiar tesl em tes5;
copiar tes1 em tes6;
copiar Cesl em tes7;
copiar tesl em tes8;
copiar tesl em tes9;
copiar tesl em tes10;
modificar tesb tal que: codp = 5 (nomep para "teste-5");
modificar tes7 tal que: codp = 5 (nomep para "teste-5");
modificar tes8 tal que: codp = 5 (nomep para "teste-5");
modificar tes9 tal que: codp = 5 (nomep para "teste-5");
modificar tes10 tal que: codp = 5 (nomep para "teste-5");
most tes1;
most tes2;
most tes3;
most tes4;
most tes5;
most tes6;
most tes7;
most tes8;
most tes9;
most tes10;
ret regs de tes2 tal que codp = 4;
ret regs de tes3 tal que codp = 4;
ret regs de tes4 tal que codp = 4;
ret regs de tes5 tal que codp = 4j
ret regs de tes6 tal que codp = 4;
ret regs de tes7 tal que codp = 4;
ret regs de tes8 tal sue codp = 4;
ret regs de tes? tal que codp = 4;
ret regs de tes10 tal que codp = 4;
most tesl;
most tes2;
most tes3;
most tes4;
most tes5;
most tes6;
most tes7;
most tes8;
```

```
most tes9;
most tes10;
most tab_autorizacoes;
fechar base de dados sem atualizar;
fechar sessao;
enc
```

c) CRIACOM

```
abrir sessao para us aluno/coppe;
abrir base de dados BDTESE;
criar arquivo tes2 com 28 registros de 2 atributos:
CODP  : I(2),
NOMEP : A(10)
com chave : codp e com indice sobre : nomep;
criar arquivo tes3 com 20 registros de 2 atributos:
CODP  : I(2),
NQMEP : A(10)
com chave : codp e com indice sobre : nomep;
criar arquivo tes4 com 20 registros de 2 atributos:
CODP  : I(2),
NOMEP : A(10)
com chave : codp e com indice sobre : nomep;
criar arquivo tes5 com 20 registros de 2 atributos:
CODP  : I(2),
NOMEP : A(10)
com chave : codp e com indice sobre : nomep;
criar arquivo tes6 com 20 registros de 2 atributos:
CODP  : I(2),
NOMEP : A(10)
com chave : codp e com indice sobre : nomep;
criar arquivo tes7 com 20 registros de 2 atributos:
CODP  : I(2),
NOMEP : A(10)
com chave : codp e com indice sobre : nomep;
criar arquivo tes8 com 20 registros de 2 atributos:
CODP  : I(2),
NOMEP : A(10)
```

```
com chave : codp e com indice sobre : nomep;
criar arquivo tes9 com 20 resistros de 2 atributos:
CODP  : I(2),
NOMEP : A(10)
com chave : codp e com indice sobre : nomep;
criar arquivo tes10 com 20 registros de 2 atributos:
CODP  : I(2),
NOMEP : A(10)
com chave : codp e com indice sobre : nomep;
most tab_autorizacoes;
copiar tesl em tes2;
copiar tesl em tes3;
copiar tes1 ein tes4;
copiar tesl em tes5;
copiar tesl em tes6;
copiar tesl em tes7;
copiar tes1 em tes8;
copiar tesl em tes9;
copiar tes1 em tes10;
modificar tes6 tal que: codp = 5 (nomep para "teste-5");
modificar tes7 tal que: codp = 5 (nomep para "teste-5");
modificar tes8 tal que: codp = 5 (nomep para "teste-5");
modificar tes9 tal que: codp = 5 (nomep para "teste-5");
modificar tes10 tal que: cadp = 5 (nomep para "teste-5");
most tesl;
most tes2;
most tes3;
most tes4;
most tes5;
most tes6;
most tes7;
most tes8;
most tes9;
most tes10;
ret regs de tes2 tal que codp = 4;
ret regs de tes3 tal que codp = 4;
ret regs de tes4 tal sue codp = 4;
ret regs de tes5 tal que codp = 4;
ret regs de tes6 tal que codp = 4;
```

```
ret regs de tes7 tal que codp = 4;
ret regs de tes8 tal que codp = 4;
ret regs de tes9 tal que codp = 4;
ret regs de tes10 tal que codp = 4;
most tes1;
most tes2;
most tes3;
most tes4;
most tes5;
most tes6;
most tes7;
most tes8;
most tes9;
most tes10;
most tab_autorizacoes;
fechar base de dados;
fechar sessao;
enc
```

2 - Simulação de Falha de Transação/Sistema

a) Chamada do programa **INIBDT** (inicializa a base de dados)

```
inibdt
Stop - Program terminated
```

b) Chamada do programa **INILOG** (inicializa os diários **BI** e **AI**)

```
inilog
   ******* INICIALIZACAO DO BI *******

   Localizacao do arquivo 81          - C:\CPREL\BI
   No. de paginas alocadas para o BI -          50

   ************************************

   ******* INICIALIZACAO DO AI *******

   Localizacao do arquivo AI          - C:\CPREL\AI
   No. de paginas alocadas para o AI -         170

   ************************************
Stop - Program terminated.
```

c) Chamada do programa SUPESQ (inicializa a super-esquema)

```
supesq
Stop - Program terminated.
```

d) Chamada do programa **CRLATESE** (cria uma base de dados)

```
cprel criatese
#?
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          - C:\CPREL\BI
No. de paginas alocadas para o BI -          50
```

```
Percentual de paginas   para o BI -    9.000000E-01
Localizacao do arquivo AI          - C:\CPREL\AI
No. de paginas alocadas para o AI -          170
Percentual de paginas   para o AI -    9.000000E-01
#?
 criar base de dados BDTESE em F1 r F2 para 50 regs por
 arq;
ABREBD - abriu AI
CRIABD - posicionou AI, icontai              2
FIMAI - testa espaco do AI          170            2
ABREBD - abriu BI
Leu BI, tipo, pagina          0             0
Recupera o super-esquema - SUPES1
CRIABD - grava primeiro registro na BASE
grava 1o reg. do AI, proximo registro            2
FIMAI - testa espaco do AI          170            3
grava 1o reg da BASE no AI            3            1          1
FIMAI - testa espaco do AI          170            4
CRIABD,grava reg. TBD no I            4            5          1
FIMAI - testa eçpaco do AI          170            5
CRIABD,grava reg. TUB no AI           5            4          2
#?
 autorizar usuario aluno/coppe para ler tab_entidades;
#?
 autorizar usuario aluno/coppe para remover
 tab_autorinacoes;
#?
 autorizar usuario aluno/coppe para ler tab_atributos;
#?
 autorizar usuario aluno/coppe para ler tab_derivacoes;
#?
 criar arquiva tes1 com 20 regs de 2 atributos:
#?
 codigo:i(2),
#?
 nome:a(10)
#?
 cam chave: codigo;
          2           2
```

```
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES  5 02/06/70
ALUNO            TAB_ENTIDADES     4 02/06/90
ALUNO            TAB_ATRIBUTOS     4 02/06/90
ALUNO            TAB_DERIVACOES    4 02/06/90
ALUNO            TES1              3 02/06/90
#?
 fechar base de dados;
FECHBD, testa flag de rollback          0
FECHBD, testa ultima transacao          1
FIMBI- Testa espaco do BI          50          0
FIMAI - testa espaco do AI         170         5
FIMBI- Testa espaco do BI          50          1
Grava reg. na BI          1          2
Grava do buffer para a BASE
FIMAP - testa espaco do AI         170         6
Grava nova pagina no AI          2          2
FIMBI- Testa espaco do BI          50          2
Grava reg. no BI          1          17
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         7
Grava nova pagina no AI          2          17
FIMBI- Testa espaco do BI          50          3
Grava reg. no BI          1          15
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         8
Grava nova pagina no AI          2          15
FIMBI- Testa espaco do BI          50          4
Grava reg. no BI          1          5
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         9
Grava nova pagina no AI          2          5
FIMBI- Testa espaco do BI          50          5
Grava reg. no BI          1          19
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         10
Grava nova pagina no AI          2          19
FIMBI- Testa espaco do BI          50          6
```

```
Grava reg. no BI          1          4
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          11
Grava nova pagina no AI          2           4
FIMBI- Testa espaco do BI         50           7
Grava reg. no BI          1         16
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          12
Grava nova pagina no AI          2          16
FIMBI- Testa espaco do BI         50           8
Grava reg. no BI          1         14
Grava do buffer para a BAÇE
FIMAI - testa espaco do AI        170          13
Grava nova pagina no AI          2          14
FIMBI- Testa espaco do BP         50           9
Grava reg. no BI          i        171
Grava do buffer para a BAÇE
FIMAI - testa espaco do AI        170          14
Grava nova pagina no AI          2         171
FIMBI- Testa espaco do BI         50          10
Grava reg. no BI          1         51
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          15
Grava nova pagina no AI          2          51
FIMBI- Testa espaco do BI         50          11
Grava reg. no BI          1         46
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          16
Grava nova pagina no AI          2          46
FIMBI- Testa espaco do BI         50          12
Grava reg. no BI          1         47
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          17
Grava nova pagina no AI          2          47
FIMBI- Testa espaco do BI         50          13
Grava reg. no BI          ▮         45
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          18
Grava nova pagina no AI          2          45
```

FIMBI- Testa espaco do BI 50 14
Grava reg. no **BI** 1 **6**
Grava do buffer para a **BASE**
FIMAI - testa espaco do **AI** 170 19
Grava nova pagina no **AI** 2 6
FIMBI- Testa espaco do BI 50 15
Grava reg. no BI 1 44
Grava do buffer rara a **BASE**
FIMAI - testa espaco do **AI** 170 20
Grava nova pagina no **AI** 2 44
FIMBI- Testa espaco do **BI** 50 16
Grava reg. no BI 1 21
Grava do buffer para a **BASE**
FIMAI - testa espaco do **AI** 170 21
Grava nova pagina no **AI** 2 21
FIMBI- Testa espaco do BI 50 17
Grava reg. no **BI** 1 52
Grava do buffer para a **BASE**
FIMAI - testa espaco do **AI** 170 22
Grava nova pagina no **AI** 2 52
FIMBI- Testa espaco do **BI** 50 18
Grava reg. no **BI** 1 20
Grava do buffer para a **BASE**
FIMAI - testa espaco do **AI** 170 23
Grava nova pagina no **AI** 2 20
FIMBI- Testa espaco **do** BI 50 19
Grava reg. no BI i 7
Grava do buffer para a BASE
FIMAI - testa espaco do **AI** 170 24
Grava nova pagina no **AI** 2 7
FIMBI- Testa espaco do BI 50 19
FIMAI - testa espaco do **AI** 170 24
FIMAI - testa espaco do **AI** 170 25
FECHBD grava **reg.** no **AI** 25 1 1
FECHBD atualiza o primeiro registro na **BASE**
FIMAI - testa espaco do **AI** 170 26
FECHBD,grava reg. TBD no **AI** 26 5 1
FIMAI - testa espaco do AI 170 27
FECHBD,grava reg. TUB no AI 27 **3** 1

```
FIMAI - testa espaco do AI          170          28
FECHBD, grava COMMIT no AI           28           6          0
FECHBD - fechou BI
FECHBD - fechou AI
#?
 fechar sessao;
#?
 enc
Çtop - Program terminated.
```

e) Inclusao de registros no arquivo TES1

```
cprel con
#?
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          _ C:\CPREL\BI
No. de paginas alocadas para o BI -          50
Percentual de paginas  para o BI -    9.000000E-01
Localizacao do arquivo AI          _ C:\CPREL\AI
No. de paginas alocadas para o AI -         170
Percentual de paginas  para o AI -    9.000000E-01
#?
 abrir base de dadas BDTEÇE;
ABREBD - testa flag de inicio            0
ABREBD - testa ultima transacao             1
ABREBD - abriu AI
contador = tipo do reg i do AI           28
ABREBD - pasicionou AI, icontai             28
FEMAI - testa espaco do AI          170          28
Leu BI, tipo, pagina              1           2
FIMBI - testa espaco no BI           50           1
ABREBD grava primeiro registro na BI
ABREBD grava primeiro registro na BASE
grava contador no Iro. registro de AI          28
FIMAI - testa espaco do AI          470          29
ABREBD grava reg. no AI          29           1          1
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES   5 02/06/90
```

```
ALUNO              TAB_ENTIDADES       4 02/06/90
ALUNO              TAB_ATRIBUTOS       4 02/06/90
ALUNO              TAB_DERIVACOES      4 02/06/90
ALUNO              TES1                3 02/06/90
#?
 most tes1;

 ins segs em tesl;
 ENTRE COM OS REGISTROS A INSERIR
??
1/"tes1"//2/"tes2"//3/"tes3"//4/"tes4"//5/"tes5"///
 FORAM INSERIDOS        5 REGISTROS
#?
 most tesl;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 fechar base de dados;
FECHBD, testa flag de rollback              0
FECHBD, testa ultima transacao              2
FIMBI - testa espaco do BI          50              1
FIMAI - testa espaco do AI          170             29
FIMBI - testa espaco do BI          50              2
Grava reg. no BI          2             2
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170             30
Grava nova pagina no AI           2               2
FIMBI - testa espaco do BI          50              3
Grava reg. na BI          2             52
Grava do buffer para a BASE
FLMAI - testa espaco do AI          170             31
Grava nova pagina no AI           2             52
FIMBI - testa espaco do BI          50              4
Grava reg. no BI          2             6
Grava do buffer para a BASE
FIMAE - testa espaco da AI          170             32
```

```
Grava nova pagina no AI              2              6
FIMBI - testa espaco do BI           50             5
Grava reg. no BI          2          3.71
Grava do buffer para a BASE
FIMAI - testa espaco do AI           170            33
Grava nova pagina no AI              2              171
FIMBI - testa espaco do BI           50             6
Grava reg. no BI          2          20
Grava do buffer para a BASE
FIMAI - testa espaco do AI           170            34
Grava nova pagina no AI              2              20
FIMBI - testa espaco do BI           50             6
FIMAI - testa espaco do AI           170            34
FIMAI - testa espaco do AI           170            35
FECHBD grava reg. no AI         35             1              1
FECHBD atualiza o primeiro registro na BASE
FIMAI - testa espaco do AI           170            36
FECHBD,grava reg. TBD no AI          36             5          1
FIMAI - testa espaco do AI           170            37
FECHBD,grava reg. TUB no AI          37             3          1
FIMAI - testa espaco do AI           170            38
FECHBD, grava COMMIT no AI           38             6          0
FECHBD - fechou BI
FECHBD - fechou AI
#?
 fechar sessao;
#?
 enc
Stop - Program terminated.
```

f) Chamada do programa **CRIASEM** (realiza falha de transação)

```
cprel criasem
#?
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          - C:\CPREL\BI
No. de paginas alocadas para o BI -          59
Percentual de paginas  para o BI -    9.000000E-01
Localizacao do arquivo AI          - C:\CPREL\AI
```

```
No. de paginas alocadas para o AI -          170
Percentual de paginas  rara o AI -     9.000000E-01
#?
 abrir base de dados BDTESE;
ABREBD - testa flag de inicio            0
ABRCBD - testa ultima transacao            2
ABREBD - abriu AI
contador = tipo do reg 1 do AI           38
ABREBD - posicionou AI, icontai            38
FIMAI - testa espaco do AI          170          38
Leu BI, tipo, pagina          2           1
FIMBI- Testa espaco do BI          50           1
ABREBD grava primeiro registro no BI
ABREBD grava primeiro registro na BASE
grava contador no 1ro. registro de AI          38
FIMAI - testa espaco do AI          170          39
ABREBD grava reg. no AI          39           1           1
#?
 criar arquivo tes2 com 20 registros de 2 atributos:
#?
 ÇODP  : ICE!),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2          2
#?
 criar arquivo tes3 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2          2
#?
 criar arquiva tes4 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
```

```
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes5 com 28 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           E
#?
 criar arquivo tes6 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#-?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes7 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes8 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
```

```
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes9 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes10 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 most tab_autorizacoes;
ALUNO             TAB_AUTORIZACOES   5 02/06/90
ALUNO             TAB_ENTIDADES      4 02/06/90
ALUNO             TAB_ATRIBUTOS      4 02/06/90
ALUNO             TAB_DERIVACOES     4 02/06/90
ALUNO             TES1               3 02/06/90
ALUNO             TES2               3 02/06/90
ALUNO             TES3               3 02/06/90
ALUNO             TES4               3 02/06/90
ALUNO             TES5               3 02/06/90
ALUNO             TES6               3 02/06/90
ALUNO             TES7               3 02/06/90
ALUNO             TES8               3 02/06/90
ALUNO             TES9               3 02/06/90
ALUNO             TES10              3 02/06/90
#?
 copiar tes1 em tes2;
#?
```

```
 copiar tes1 em tes3;
#?
 copiar tesl em tes4;
#?
 copiar tes1 em tes5;
#?
 copiar tes1 em tes6;
#?
 copiar tes1 em tes7;
#?
 copiar tes1 em tes8;
#?
 copiar tes1 em tes9;
#?
 copiar tes1 em tes10;
#?
 modificar tes6 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI          50          1
FIMAI - testa espaco do AI         170         39
FIMBI- Testa espaco do BI          50          2
Grava reg. no BI          3          55
Grava do buffer para a BASE
FHMAI - testa espaco do AI         170         40
Grava nova pagina no AI          2          55
FZMBI- Testa espaco do BI          50          3
Grava reg. no BI          3          6
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         41
Grava nova pagina no AI          2          6
FIMBI- Testa espaco do BI          50          4
Grava reg. no BI          3          2
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         42
Grava nova pagina no AI          2          2
FIMBI- Testa espaco do BI          50          5
Grava reg. no BI          3          171
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         43
Grava nova pagina no AI          2          171
```

FIMBI- Testa espaco do BI 50 6
Grava reg. no BI 3 15
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 44
Grava nova pagina no AI 2 15
FIMBI- Testa espaco do BI 50 7
Grava reg. no BI 3 47
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 45
Grava nova pagina no AI 2 47
FIMBI- Testa espaco do BI 50 8
Grava reg. no BI 3 54
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 46
Grava nova pagina no AI 2 54
FIMBI- Testa espaco do BI 50 9
Grava reg. nu BI 3 52
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 47
Grava nova pagina no AI 2 52
FIMBI- Testa espaco do BI 50 10
Grava reg. no BI 3 4
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 48
Grava nova pagina no AI 2 4
FIMBI- Testa espaco do BI 50 11
Grava reg. no BI 3 14
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 49
Grava nova pagina no AI 2 14
FIMBI- Testa espaco do BI 50 12
Grava reg. no BE 3 56
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 50
Grava nova pagina no AI 2 56
FIMBI- Testa espaco do BI 50 13
Grava reg. no BI 3 53
Grava do buffer para a BASE
FIMAE - testa espaco do AI 170 51

```
Grava nova pagina no AI          2          53
FIMBI- Testa espaco do BI        50         14
Grava reg. nu BI          3      17
Grava do buffer rara a BASE
FIMAI - testa espaco do AI       170        52
Grava nova pagina nu AI          2          17
FIMBI- Testa espaco do BI        50         15
Grava reg. no BI          3      19
Grava do buffer para a BASE
FIMAI -- testa espaco do AI      170        53
Grava nova pagina no AI          2          19
FIRBI- Testa espaco do BI        50         16
Grava reg. no BI          3      16
Grava do buffer para a BASE
FIMAI - testa espaco do AI       170        54
Grava nova pagina no AI          2          16
FIMBI- Testa espaco do BI        50         17
Grava reg. no BI          3      20
Grava do buffer para a BASE
FIMAI - testa espaco do AI       170        55
Grava nova pagina nu AI          2          20
FIMBI- Testa espaco do BI        50         18
Grava reg. no BI          3      57
Grava do buffer para a BAÇE
FIMAI - testa espaco do AI       170        56
Grava nova pagina no AI          2          57
#?
 modificar tes7 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco da BI        50         18
FIMAI - testa espaco do AI       170        56
Grava do buffer para a BASE
Grava mesma pagina no AI         2          55
Grava do buffer para a BASE
Grava mesma pagina na AI         2          6
Grava do buffer para a BASE
Grava mesma pagina no AI         2          2
Grava do buffer para a BASE
Grava mesma pagina no AI         2          171
Grava do buffer para a BASE
```

```
Grava mesma pagina no AI          2          15
Grava do buffer para a BASE
Grava mesma pagina no AI          2          47
Grava do buffer para a BAÇE
Grava mesma pagina no AI          2          52
Grava do buffer para a BASE
Grava mesma pagina no AI          2           4
Grava do buffer para a BASE
Grava mesma pagina no AI          2          14
Grava do buffer para a BASE
Grava mesma pagina no AI          2          54
Grava do buffer para a BASE
Grava mesma pagina no AI          2          56
Grava do buffer para a BASE
Grava mesma pagina no AI          2          53
Grava do buffer para a BASE
Grava mesma pagina no AI          2          17
Grava do buffer para a BASE
Grava mesma pagina no AI          2          19
Grava do buffer para a BAÇE
Grava mesma pagina no AI          2          16
Grava do buffer para a BASE
Grava mesma pagina no AI          2          20
Grava do buffer para a BASE
Grava mesma pagina no AI          2          57
#?
 modificar tes8 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco da BI         50          18
FIMAI - testa espaco do AI        170         56
Grava do buffer para a BASE
Grava mesma pagina no AI          2          56
Grava do buffer para a BAÇE
Grava mesma pagina no AI          2           6
Grava do buffer para a BASE
Grava mesma pagina no AI          2           2
Grava do buffer para a BASE
Grava mesma pagina no AI          2          55
Grava do buffer para a BASE
Grava mesma pagina no AI          2         171
```

```
Grava do buffer para a BASE
Grava mesma pagina no AI          2          15
Grava do buffer para a BASE
Grava mesma pagina no AI          2          47
Grava do buffer para a BASE
Grava mesma pagina no AI          2          52
Grava do buffar para a BASE
Grava mesma pagina no AI          2           4
Grava do buffer para a BASE
Grava mesma pagina no AI          2          14
Grava do buffer para a BASE
Grava mesma pagina no AI          2          54
Grava do buffer para a BAÇE
Grava mesma pagina no AI          2          53
Grava do buffer para a BASE
Grava mesma pagina no AI          2          17
Grava do buffer para a BASE
Grava mesma pagina no AI          2          19
Grava do buffer para a BASE
Grava mesma pagina no AI          2          16
Grava do buffer para a BASE
Grava mesma pagina na AI          2          20
Grava do buffer para a BASE
Grava mesma pagina no AI          2          57
#?
 modificar tes9 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI         50          18
FIMAI - testa espaco do AI        170          56
Grava do buffer para a BASE
Grava mesma pagina no AI          2          56
Grava do buffer para a BASE
Grava mesma pagina no AI          2           2
Grava do buffer para a BASE
Grava mesma pagina no AI          2          47
Grava do buffer para a BASE
Grava mesma pagina no AI          2           6
Grava do buffer para a BASE
Grava mesma pagina no AI          2         171
Grava do buffer para a BASE
```

```
Grava mesma pagina no AI                2          4
Grava do buffer para a BASE
Grava mesma pagina no AI                2          15
Grava do buffer para a BASE
Grava mesma pasina no AI                2          52
Grava do buffer para a BASE
Grava mesma pagina no AI                2          14
Grava do buffer para a BASE
Grava mesma pagina no AI                2          55
Grava do buffer para a BASE
Grava mesma pagina no AI                2          54
Grava do buffer para a BASE
Grava mesma pagina no AI                2          53
Grava do buffer para a BASE
Grava mesma pagina no AI                2          17
Grava do buffer para a BASE
Grava mesma pagina no AI                2          19
Grava do buffer para a BASE
Grava mesma pagina no AI                2          16
Grava do buffer para a BASE
Grava mesma pagina no AI                2          20
Grava do buffer para a BASE
Grava mesma pagina no AI                2          57
H?
 modificar tes10 tal que: codp = 5 (nomep para "teste-3");
FIMBI- Testa espaco do BI               50          18
FIMAI - testa espaco do AI              170          56
Grava do buffer para a BASE
Grava mesma pagina no AI                2          56
Grava do buffer para a BASE
Grava mesma pagina no AI                2           6
Grava do buffer para a BASE
Grava mesma pagina no AI                2           2
Grava do buffer para a BASE
Grava mesma pagina no AI                2          171
Grava do buffer para a BASE
Grava mesma pagina no AI                2          47
Grava do buffer para a BASE
Grava mesma pagina no AI                2          45
```

```
Grava do buffer para a BASE
Grava mesma pagina no AI          2          52
Grava do buffer para a BASE
Grava mesma pagina no AI          2          14
Grava do buffer para a BASE
Grava mesma pagina no AI          2           4
Grava do buffer para a BASE
Grava mesma pagina no AI          2          55
Grava do buffer para a BASE
Grava mesma pagina no AI          2          54
Grava do buffer para a BASE
Grava mesma pagina no AI          2          53
Grava do buffer para a BASE
Grava mesma pagina no AI          2          17
Grava do buffer para a BASE
Grava mesma pagina no AI          2          19
Grava do buffer para a BASE
Grava mesma pagina no AI          2          16
Grava do buffer para a BASE
Grava mesma pagina no AI          2          20
Grava do buffer para a BASE
Grava mesma pagina no AI          2          57
#?
 most tes1;
 81 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 most tes2;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 most tes3;
 01 TES1
```

```
 02 TES2
 03 TES3
 84 TES4
 05 TES5
#?
 most tes4;
 01 TES1
 02 TES2
 03 TES3
 84 TES4
 05 TES5
#?
 inost tes5;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESS
#?
 most tes6;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESTE-5
#?
 most tes7;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESTE-5
#?
 most tes8;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 83 TESTE-5
```

```
#?
 most tes9;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESTE-5
#?
 most tes10;
 01 TESP
 0% TES2
 03 TES3
 04 TES4
 05 TESTE-5
#?
 ret regs de tes2 tal que codp = 4;
#?
 ret regs de tes3 tal que codp = 4;
#?
 ret regs de tes4 tal que codp = 4;
#?
 ret regs de tes5 tal sue codp = 4;
#?
 ret regs de tes6 tal que codp = 4;
#?
 ret regs de tes7 tal que codp = 4;
#?
 ret regs de tes8 tal que codp = 4;
#?
 ret regs de tes9 tal que codp = 4;
#?
 ret regs de tes10 tal que codp = 4;
#?
 most tes1;
 01 TESI
 02 TES2
 03 TES3
 84 TES4
 05 TES5
```

```
#?
 most tes2;
 81 TES1
 O2 TES2
 O3 TES3
 05 TESS
#?
 most tes3;
 01 TESL
 O2 TES2
 O3 TES3
 O5 TES5
#?
 most tes4;
 81 TES1
 O2 TES2
 O3 TES3
 05 TCS5
#?
 most tes5;
 01 TES1
 82 TES2
 O3 TES3
 05 TCS5
#?
 most tesá;
 01 TES1
 02 TCS2
 03 TES3
 05 TESTE-5
#?
 most tes7;
 01 TES1
 O2 TES2
 O3 TES3
 05 TESTE-5
#?
 most tes8;
 01 TES1
```

```
 02 TES2
 03 TCS3
 05 TESTE-5
#?
 most tes9;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes10;
 01. TES1
 82 TES2
 03 TEÇ3
 05 TESTE-L)
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES  5 02/06/90
ALUNO            TAB_ENTIDADES     4 02/06/90
ALUNO            TAB_ATRIBUTOS     4 02/06/90
ALUNO            TAB_DERIVACOES    4 02/06/90
ALUNO            TES1              3 02/06/90
ALUNO            TES2              3 02/06/90
ALUNO            TES3              3 02/06/90
ALUNO            TES4              3 02/06/90
ALUNO            TES5              3 02/06/90
ALUNO            TES6              3 02/06/90
ALUNO            TES7              3 02/06/90
ALUNO            TES8              3 02/06/90
ALUNO            TES9              3 02/06/90
ALUNO            TES10             3 02/06/90
#?
 Cechar base de dados sem atualizar;
FECHBD, testa flag de rollback          1
FECHBD, testa ultima transacao          3
ROLLBACK DA TRANSACAO
testa se iniciou por CRIABD/ABREBD          0
Recupera BDTRABA, tipo, pagina          3          1
Recupera BUTRABA, tipo, pagina          3         55
```

```
Recupera BDTRABA, tipo, pagina          3          6
Recupera BDTRABA, tipo, pagina          3          2
Recupera BDTRABA, tipo, pagina          3        171
Recupera BDTRABA, tipo, pagina          3         15
Recupera BDTRABA, tipo, pagina          3         47
Recupera BDTRABA, tipo, pagina          3         54
Recupera BDTRABA, tipo, pagina          3         52
Recupera BDTRABA, tipo, pagina          3          4
Recupera BDTRABA, tipo, pagina          3         14
Recupera BDTRABA, tipo, pagina          3         56
Recupera BDTRABA, tipo, pagina          3         53
Recupera BDTRABA, tipo, pagina          3         17
Recupera BDTRABA, tipo, pagina          3         19
Recupera BDTRABA, tipo, pagina          3         16
Recupera BDTRABA, tipo, pagina          3         20
Recupera BDTRABA, tipo, pagina          3         57
FECHBD - fechou BI
FECHBD - fechou AI
#?
 fechar sessao;
#?
 enc
Stop - Program terminated.
```

g) Chamada do programa CRIACOM (simulação de falha de sistema)

```
cprel criacom
#?
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          - C:\CPREL\BI
No. de paginas alocadas para o BI -          50
Percentual de paginas   para o BI -    9.000000E-01
Localizacao do arquivo AI          - C:\CPREL\AI
No. de paginas alocadas para o AI -         170
Percentual de paginas   para o AI -    9.000000E-01
#?
 abrir base de dados BDTESE;
ABREBD - testa flag de inicio           0
```

```
RBREBD - testa ultima transacao            6
RBREBD - abriu AI
contador = tipo do reg 1 do AI          112
ABREBD - posicionou AI, icontai           112
FIMAI - testa espaco do AI          170          112
Leu BI, tipo, pagina            6            1
FIMBI- Testa espaco do BI           50           1
ABREBD grava primeiro registro no BI
ABREBD grava primeiro registro na BASE
grava contador no 1ro. registro de AI          112
FIMAI - testa espaco do AI          170          113
ABREBD grava reg. no AI          113           1            1
#?
 criar arquivo tes2 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMCP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2          2
#?
 criar arquivo tes3 com 28 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2          2
#?
 criar arquivo te54 com 28 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2          2
```

```
#?
 criar arquivo tes5 com 28 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes6 com 28 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes7 com 28 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOHEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquiva tes8 com 28 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes9 com 20 registros de 2 atributos:
#?
```

```
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2            2
#?
 criar arquivo tes10 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2            2
#?
 most tab_autorizacoes;
ALUNO              TAB_AUTORIZACOES   5 02/06/90
ALUNO              TAB_ENTIDADES      4 02/06/90
ALUNO              TAB_ATRIBUTOS      4 02/06/90
ALUNO              TAB_DERIVACOES     4 02/06/90
ALUNO              TES1               3 02/06/90
ALUNO              TES2               3 02/06/90
ALUNO              TES3               3 02/06/90
ALUNO              TES4               3 02/06/90
ALUNO              TES5               3 02/06/90
ALUNO              TES6               3 02/06/90
ALUNO              TES7               3 02/06/90
ALUNO              TES8               3 02/06/90
ALUNO              TES9               3 02/06/90
ALUNO              TES10              3 02/06/90
#?
 copiar tes1 em tes2;
#?
 copiar tes1 em tes3;
#?
 copiar tes1 em tes4;
#?
 copiar tes1 em tes5;
```

```
#?
 copiar tes1 em tes6;
#?
 copiar tes1 em tes7;
#?
 copiar tes1 em tes8;
#?
 coriar tes1 em tes9;
#?
 copiar tesl em tes10;
#?
 modificar tes6 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BP          50          1
FIMAI - testa espaco do AI         170         75
FIMBI- Testa espaco do BI          50          2
Grava reg. no BI          5          55
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         76
Grava nova pagina no AI          2          55
FIMBI- Testa espaco do BI          50          3
Grava reg. no BI          5          6
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         77
Grava nova pagina no AI          2          6
FIMBI- Testa espaco do BI          50          4
Grava reg. no BI          5          2
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         78
Grava nova pagina no AI          2          2
FIMBI- Testa espaco do BI          50          5
Grava reg. no BI          5          171
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         79
Grava nova pagina no AI          2          171
FLMBI- Testa espaco do BI          50          6
Grava reg. no BI          5          15
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         80
Grava nova pagina no AI          2          15
```

FIMBI- Testa espaco do BI 50 7
Grava reg. no BI 5 47
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 81
Grava nova pagina no AI 2 47
FIMBI- Testa espaco do BI 50 8
Grava reg. no BI 5 54
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 82
Grava nova pagina no AI 2 54
FIMBI- Testa espaco do BI 50 9
Grava reg. no BI 5 52
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 83
Grava nova pagina no AI 2 52
FIMBI- Testa espaco do BI 50 10
Grava reg. no BI 5 4
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 84
Grava nova pagina no AI 2 4
FIMBI- Testa espaco do BI 50 11
Grava reg. no BI 5 14
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 85
Grava nova pagina no AI 2 14
FIMBI- Testa esparo do BI 50 12
Grava reg. no BI 5 56
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 86
Grava nova pagina no AI 2 56
FIMBI- Testa espaco do BI 50 13
Grava reg. no BI 5 53
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 87
Grava nova pagina no AI 2 53
FIMBI- Testa espaco do BI 50 14
Grava reg. no BI 5 17
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 88

```
Grava nova pagina no AI          2          17
FIMBI- Testa espaco do BI        50         15
Grava reg. no BI          5          19
Grava do buffer para a BASE
FIMAI - testa espaco do AI       170        89
Grava nova pagina no AI          2          19
FIMBI- Testa espaco do BI        50         16
Grava reg. no BI          5          16
Grava do buffer para a BASE
FIMAI - testa espaco do AI       170        90
Grava nova pagina no AI          2          16
FIMBI- Testa espaco do BI        50         17
Grava reg. no BI          5          20
Grava do buffer para a BASE
FIMAI - testa espaco do AI       170        91
Grava nova pagina no AI          2          20
FIMBI- Testa espaco do BI        50         18
Grava reg. no BI          5          57
Grava do buffer para a BASE
FIMAI - testa espaco do AI       170        92
Grava nova pagina no AI          2          57
#?
 modificar tes7 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI        50         18
FIMAI - testa espaco do AI       170        92
Grava do buffer para a BASE
Grava mesma pagina no AI         2          55
Grava do buffer para a BASE
Grava mesma pagina no AI         2          6
Grava do buffer para a BASE
Grava mesma pagina no AI         2          2
Grava do buffer para a BASE
Grava mesma pagina no AI         2          171
Grava do buffer para a BASE
Grava mesma pagina no AI         2          15
Grava do buffer para a BASE
Grava mesma pagina no AT         2          47
Grava do buffer para a BASE
Grava mesma pagina no AI         2          52
```

```
Grava do buffer para a BASE
Grava mesma pagina no AI          2          4
Grava do buffer para a BASE
Grava mesma pagina no AI          2          14
Grava do buffer para a BASE
Grava mesma pagina no AI          2          54
Grava do buffer para a BASE
Grava mesma pagina no AI          2          56
Grava do buffer para a BAÇE
Grava mesma pagina no AI          2          53
Grava do buffer para a BASE
Grava mesma pagina no AI          2          17
Grava do buffer para a BASE
Grava mesma pagina no AI          2          19
Grava do buffer para a BASE
Grava mesma pagina no AI          2          16
Grava do buffer para a BASE
Grava mesma pagina no AI          2          20
Grava do buffer para a BAÇE
Grava mesma pagina no AI          2          57
#?
 modificar tes8 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI         50          18
FIMAI - testa espaco do AI        170          92
Grava do buffer para a BASE
Grava mesma pagina no AI          2          56
Grava do buffer para a BASE
Grava mesma pagina no AI          2          6
Grava do buffer para a BAÇE
Grava mesma pagina no AI          2          2
***  silmulação de falha de sistema (Control+Alt+Del) ***
```

h) Chamada do Programa CRIACOM (recuperação da simulação de falha de sistema e execução normal do CRIACOM com "commit")

```
cprel criacom
#?
```

```
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          C:\CPREL\BI
No. de paginas alocadas para a BI -          50
Percentual de paginas  para o BI -    9.000000E-01
Localizacao do arquivo AI          C:\CPREL\AI
Na. de paginas alocadas para o AI -         170
Percentual de paginas  para o AI -    9.000000E-01
#?
 abrir base de dados BDTESE;
ABREBD - testa flag de inicio            1
ABREBD - testa ultima transacao            4
RECUPERACAO DE FALHA DE SISTEMA
Recupera BDTRABA, tipo, pagina             4           1
Recupera BDTRABA, tipo, pagina             4          55
Recupera BDTRABA, tipo, pagina             4           6
Recupera BDTRABA, tipo, pagina             4           2
Recupera BDTRABA, tipo, pagina             4         171
Recupera BDTRABA, tipo, pagina             4          15
Recupera BDTRABA, tipo, pagina             4          47
Recupera BDTRABA, tipo, pagina             4          54
Recupera BDTRABA, tipo, pagina             4          52
Recupera BDTRABA, tipo, pagina             4           4
Recupera BDTRABA, tipo, pagina             4          14
Recupera BDTRABA, tipo, pagina             4          56
Recupera BDTRABA, tipo, pagina             4          53
Recupera BDTRABA, tipo, pagina             4          17
Recupera BDTRABA, tipo, pagina             4          19
Recupera BDTRABA, tipo, pagina             4          16
Recupera BDTRABA, tipo, pagina             4          20
Recupera BDTRABA, tipo, pagina             4          57
ABREBD - abriu AI
posiciona AI a partir de FALHA DE SISTEMA
ABREBD - posicionou AI, icontai            74
FIMAI - testa espaco da AI          170          74
Leu BI, tipo, pagina          4           1
FIMBI- Testa espaco do BI           50           1
ABREBD grava primeiro registro no BI
ABREBD grava primeiro registro na BASE
grava contador no 1ro. registro de AI          74
```

```
FIMAI - testa espaco do AI          170          75
ABREBD grava reg. no AI          75          2          1
#?
 criar arquivo tes2 com 28 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2          2
#?
 criar arquivo tes3 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2          2
#?
 criar arquivo tes4 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NQMCP : A(10)
#?
 com chave : codp e com indice sobre : namep;
         2          2
#?
 criar arquivo tes5 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e cam indice sobre : nomep;
         2          2
#?
```

```
 criar arquivo tes6 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 MOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes7 com 28 registros de 2 atributos
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquive tes8 com 28 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 MOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes9 com 28 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
ti?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes10 com 28 registros de 2 atributos:
#?
 CODP   : I(2),
```

```
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 most tab_autorizacoes;
ALUNO              TAB_AUTORIZACOES  5 02/06/90
ALUNO              TAB_ENTIDADES     4 02/06/90
ALUNO              TAB_ATRIBUTOS     4 02/06/90
ALUNO              TAB_DERIVACOES    4 02/06/90
ALUNO              TES1              3 02/06/90
ALUNO              TES2              3 02/06/90
ALUNO              TES3              3 02/06/90
ALUNO              TES4              3 02/06/90
ALUNO              TESS              3 02/06/90
ALUNO              TES6              3 02/06/90
ALUNO              TES7              3 02/06/90
ALUNO              TES8              3 02/06/90
ALUNO              TES9              3 02/06/90
ALUNO              TES10             3 02/06/90
#?
 copiar tes1 em tes2;
#?
 copiar tes1 em tes3;
#?
 copiar tes1 em tes4;
#?
 copiar tes1 em tes5;
#?
 copiar tes1 em tes6;
#?
 copiar tes1 em tes7;
#?
 copiar tes1 em tes8;
#?
 copiar tes1 em tes9;
#?
 copiar tes1 em tes10;
```

```
#?
 modificar tes6 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI          50          1
FIMAI - testa espaco do AI         170         75
FIMBI- Testa espaco do BI          50          2
Grava reg. no BI          5          55
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         76
Grava nova pagina no AI           2          55
FIMBI- Testa espaco do BI          50          3
Grava reg. no BI          5          6
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         77
Grava nova pagina no AI           2          6
FIMBI- Testa espaco do BI          50          4
Grava reg. no BI          5          2
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         78
Grava nova pagina no AI           2          2
FIMBI- Testa espaco do BI          50          5
Grava reg. no BI          5          171
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         79
Grava nova pagina no AI           2          171
FIMBI- Testa espaco do BI          50          6
Grava reg. no BI          5          15
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         80
Grava nova pagina no AI           2          15
FIMBI- Testa espaco do BI          50          7
Grava reg. no BI          5          47
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         81
Grava nova pagina no AI           2          47
FIMBI- Testa espaco do BI          50          8
Grava reg. no BI          5          54
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         82
Grava nova pagina no AI           2          54
```

FIMBI- Testa espaco do BI 50 9
Grava reg. no BI 5 52
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 83
Grava nova pagina no AI 2 52
FIMBI- Testa espaco do BI 50 10
Grava reg. no BI 5 4
Grava da buffer para a BASE
FIMAI - testa espaco do AI 170 84
Grava nova pagina no AI 2 4
FIMBI- Testa espaco do BI 50 11
Grava reg. no BI 5 14
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 85
Grava nova pagina no AI 2 14
FIMBI- Testa espaco do BI 50 12
Grava reg. no BI 5 56
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 86
Grava nova pagina no AI 2 56
FIMBI- Testa espaco do BI 50 13
Grava reg. no BI 5 53
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 87
Grava nova pagina no AI 2 53
FIMBI- Testa espaco do BI 50 14
Grava reg. no BI 5 17
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 88
Grava nova pagina no AI 2 17
FIMBI- Testa espaco do BI 50 15
Grava reg. no BI 5 19
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 89
Grava nova pagina no AI 2 19
FLMBI- Testa espaco do BI 50 16
Grava reg. no BI 5 16
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 90

```
Grava nova pagina no AI          2          16
FPMBI- Testa espaco do BI        50         17
Grava reg. no BI          5          20
Grava do buffer para a BASE
FIMAI - testa espaco do AI       170        91
Grava nova pagina no AI          2          20
FIMBI- Testa espaco do BI        50         18
Grava reg. no BI          5          57
Grava do buffer para a BASE
FIMAI - testa espaco do AI       170        92
Grava nova pagina no AI          2          57
#?
 modificar tes7 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI        50         18
FIMAI - testa espaco do AI       170        92
Grava do buffer para a BASE
Grava mesma pagina no AI         2          55
Grava do buffer para a BASE
Grava mesma pagina no AI         2          6
Grava do buffer para a BASE
Grava mesma pagina no AI         2          2
Grava do buffer para a BASE
Grava mesma pagina no AI         2          171
Grava do buffer para a BASE
Grava mesma pagina no AI         2          15
Grava do buffer para a BASE
Grava mesma pagina no AI         2          47
Grava do buffer para a BASE
Grava mesma pagina no AI         2          52
Grava do buffer para a BASE
Grava mesma pagina no AI         2          4
Grava do buffer para a BASE
Grava mesma pagina no AI         2          14
Grava do buffer para a BASE
Grava mesma pagina no AI         2          54
Grava do buffer para a BASE
Grava mesma pagina no AI         2          56
Grava do buffer para a BASE
Grava mesma pagina no AI         2          53
```

```
Grava do buffer para a BASE
Grava mesma pagina no AI          2          17
Grava do buffer para a BASE
Grava mesma pagina no AI          2          19
Grava do buffer para a BASE
Grava mesma pagina no AI          2          16
Grava do buffer para a BASE
Grava mesma pagina no AI          2          20
Grava do buffer para a BASE
Grava mesma pagina no AI          2          57
#?
 modificar tes8 tal que: codp = 5 (nomep para "teste-5");
FIMBL- Testa espaco do BI         50          18
FIMAI - testa espaco do AI        170          92
Grava do buffer para a BASE
Grava mesma pagina no AI          2          56
Grava do buffer para a BASE
Grava mesma pagina no AI          2           6
Grava do buffer para a BASE
Grava mesma Pagina no AI          2           2
Grava do buffer para a BASE
Grava mesma pagina no AI          2          55
Grava do buffer para a BASE
Grava mesma pagina no AI          2         171
Grava do buffer para a BASE
Grava mesma pagina no AI          2          15
Grava do buffer para a BASE
Grava mesma pagina no AI          2          47
Grava do buffer para a BASE
Grava mesma pagina no AI          2          52
Grava do buffer para a BASE
Grava mesma pagina no AI          2           4
Grava do buffer para a BASE
Grava mesma pagina no AI          2          14
Grava do buffer para a BASE
Grava mesma pagina no AI          2          54
Grava do buffer para a BASE
Grava mesma pagina no AI          2          53
Grava do buffer para a BASE
```

```
Grava mesma pagina no AI          2          17
Grava do buffer para a BASE
Grava mesma pagina no AI          2          19
Grava do buffer para a BAÇE
Grava mesma pagina no AI          2          16
Grava do buffer para a BASE
Grava mesma pagina no AI          2          20
Grava do buffer para a BASE
Grava mesma Pagina no AI          2          57
#?
 modificar tes9 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI         50          18
FIMAI - testa espaco do AI        170          92
Grava do buffer para a BAÇE
Grava mesma pagina no AI          2          56
Grava do buffer para a BASE
Grava mesma pagina no AI          2           2
Grava do buffer para a BASE
Grava mesma pagina no AI          2          47
Grava do buffer para a BASE
Grava mesma pagina no AI          2           6
Grava do buffer para a BASE
Grava mesma pagina no AI          2         171
Grava do buffer para a BAÇE
Grava mesma pagina no AI          2           4
Grava do buffer para a BASE
Grava mesma pagina no AI          2          15
Grava do buffer para a BASE
Grava mesma pagina no AI          2          52
Grava do buffer para a BASE
Grava mesma pagina no AI          2          14
Grava do buffer para a BASE
Grava mesma pagina no AI          2          55
Grava do buffer para a BASE
Grava mesma pagina no AI          2          54
Grava do buffer para a BASE
Grava mesma pagina no AI          2          53
Grava do buffer para a BASE
Grava mesma pagina no AI          2          17
```

```
Grava do buffer para a BASE
Grava mesma pagina no AI          2          19
Grava do buffer para a BASE
Grava mesma pagina no AI          2          16
Grava do buffer para a BASE
Grava mesma pagina no AI          2          20
Grava do buffer para a BASE
Grava mesma pagina no AI          2          57
#?
 modificar tes10 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI         50          18
FIMAI - testa espaco do AI        170         92
Grava do buffer Para a BASE
Grava mesma pagina no AI          2          56
Grava do buffer para a BASE
Grava mesma pagina no AI          2           6
Grava do buffer para a BASE
Grava mesma pagina no AI          2           2
Grava do buffer para a BASE
Grava mesma pagina no AI          2         171
Grava do buffer  para a BASE
Grava mesma pagina no AI          2          47
Grava do buffer Para a BASE
Grava mesma pagina no AI          2          15
Grava do buffer Para a BASE
Grava mesma pagina no AI          2          52
Grava do buffer Para a BASE
Grava mesma pagina no AI          2          14
Grava do buffer para a BASE
Grava mesma pagina no AI          2           4
Grava do buffer para a BASE
Grava mesma pagina no AI          2          55
Grava do buffer para a BASE
Grava mesma pagina no AI          2          54
Grava do buffer Para a BASE
Grava mesma pagina no AI          2          53
Grava do buffer para a BASE
Grava mesma pagina no AI          2          17
Grava do buffer para a BASE
```

```
Grava mesma pagina no AI          2          19
Grava do buffer para a BASE
Grava mesma pagina no AI          2          16
Grava do buffer para a BASE
Grava mesma pagina no AI          2          20
Grava do buffer para a BASE
Grava mesma pagina no AI          2          57
#?
 most tesi;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 most tes2;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 most tes3;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 most tes4;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 tnost tes5;
 01 TES1
 02 TES2
```

```
 03 TES3
 04 TES4
 03 TES5
#?
 most tes6;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESTE-5
#?
 most tes7;
 81 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESTE-5
#?
 most tes8;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESTE-5
#?
 most tes9;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESTE-5
#?
 most tes10;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESTE-5
H?
```

```
 ret ress de tes2 ta1 que codp = 4;
#?
 ret regs de tes3 tal que codp = 4;
#?
 ret regs de tes4 tal que codp = 4;
#?
 ret regs de tes5 tal que codp = 4;
#?
 ret regs de tes6 tal que codp = 4;
#?
 ret regs de tes7 tal que codp = 4;
#?
 ret ress de tes8 tal que codp = 4;
#?
 ret regs de tes9 tal que codp = 4;
#?
 ret regs de tes10 tal que codp = 4;
#?
 most tes1;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESS
#?
 most tes2;
 01 TES1
 02 TES2
 03 TES3
 05 TESS
#?
 most tes3;
 01 TES1
 02 TES2
 03 TES3
 05 TESS
#?
 most tes4;
 01 TES1
```

```
 02 TES2
 03 TES3
 05 TES5
#?
 most tes5;
 01 TES1
 02 TESE!
 03 TES3
 05 TES5
#?
 most tes6;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes7;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes8;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes9;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes10;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
```

```
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES  5 02/06/90
ALUNO            TAB_ENTIDADES     4 02/06/90
ALUNO            TAB_ATRIBUTOS     4 02/06/90
ALUNO            TAB_DERIVACOES    4 02/06/90
ALUNO            TES1              3 02/06/90
ALUNO            TES2              3 02/06/90
ALUNO            TES3              3 02/06/90
ALUNO            TES4              3 02/06/90
ALUNO            TES5              3 02/06/90
ALUNO            TES6              3 02/06/90
ALUNO            TES7              3 02/06/90
ALUNO            TES8              3 02/06/90
ALUNO            TES9              3 02/06/90
ALUNO            TES10             3 02/06/90
#?
 fechar base de dados;
FECHBD, testa flag de rollback          0
FECHBD, testa ultima transacao          5
FIMBI- Testa espaco do BI          50          18
FIMAI - testa espaco do AI         170          92
Grava do buffer para a BASE
Grava mesma pagina no AI           2            2
Grava do buffer para a BASE
Grava mesma pagina no AI           2           17
Grava do buffer para a BASE
Grava mesma pagina no AI           2           15
Grava do buffer para a BASE
Grava mesma pagina no AI           2            4
Grava do buffer para a BAÇE
Grava mesma pagina no AI           2            6
Grava do buffer para a BASE
Grava mesma pagina no AI           2          171
Grava do buffer para a BASE
Grava mesma pagina no AI           2           56
Grava do buffer para a BASE
Grava mesma pagina no AI           2           14
Grava do buffer para a BASE
```

```
Grava mesma pagina no AI              2          55
Grava do buffer para a BASE
Grava mesma pagina no AI              2          54
Grava do buffer para a BASE
Grava mesma pagina no AI              2          53
Grava do buffer para a BASE
Grava mesma pagina no AI              2          52
Grava do buffer para a BASE
Grava mesma pagina no AI              2          47
Grava do buffer para a BASE
Grava mesma pagina no AI              2          19
Grava do buffer para a BASE
Grava mesma pagina no AI              2          16
Grava do buffer para a BASE
Grava mesma pagina no AI              2          20
Grava do buffer para a BASE
Grava mesma pagina no AI              2          57
FIMBI- Testa espaco do BI             50          18
FIMAI - testa espaco do AI            170         92
FIMAI - testa espaco do AI            170         93
FECHBD grava reg. no AI          93               1          1
FECHBD atualiza o primeiro registro na BASE
FIMAI - testa espaco do AI            170         94
FECHBD,grava reg. TBD no AI           94          5          1
FIMAI - testa espaco do AI            170         95
FECHBD,grava reg. TUB no R1           95          3          1
FIMAI - testa espaco do AI            170         96
FECHBD, grava COMMIT no AI            96          6          0
FECHBD - fechou BI
FECHBD - fechou AI
#?
 fechar sessao;
#?
 enc
Stop - Program terminated.
```

3 - Simulação de Falha de Meio de Armazenamento

a) Chamada do programa INIBDT (inicializa a base de dados)

```
inibdt
Stop - Program terminated.
```

b) Chamada do programa INILOG (inicializa os diários BI e AI)

```
inilog
   ******* INICIALIZACAO DO BI *******

   Localiracao do arquivo BI          - C:\CPREL\BI
   No. de paginas alocadas para o BI -          50

   ************************************

   ******* INICIALIZACAO DO AI *******

   Localizacao do arquivo AI          - C:\CPREL\AI
   No. de paginas alocadas para o AI -         170

   ************************************
Stop - Program terminated.
```

c) Chamada do programa SUPESQ (inicializa o super-esquema)

```
supesq
Stop - Program terminated
```

d) Chamada do programa CRIATESE (cria uma base de dados)

```
cprel criatese
#?
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          - C:\CPREL\BI
No. de paginas alocadas para o BI -          50
```

```
Percentual de paginas   para o BI -    9.000000E-01
Localizacao do arquivo AI         - C:\CPREL\AI
No. de paginas alocadas para o AI -         170
Percentual de paginas   para o AI -    9.000000E-01
#?
 criar base de dados BDTESE em F1 e F2 para 50 regs por
arq;
ABREBD - abriu AI
CRIABD - posicionou AI, icontai            2
FIMAI - testa espaco do AI         170           2
ABREBD - abriu BI
Leu BI, tipo, pagina           0           0
Recupera o super-esquema - SUPES1
CRIABD - grava primeiro registro na BASE
grava 1o reg. do AI, proximo registro           2
FIMAI - testa espaco do AI         170           3
grava 1o reg da BASE no AI           3           1        1
FIMAI - testa espaco do AI         170           4
CRIABD,grava reg. TBD no I           4           5        1
FIMAI - testa espaco do AI         170           5
CRIABD,grava reg. TUB no AI          5           4        2
#?
 autorizar usuario aluno/coppe para ler tab_entidades;
#?
 autorizar usuario aluno/coppe para remover
tab_autorizacoes;
#?
 autorizar usuario aluno/coppe para ler tab_atributos;
#?
 autorizar usuario aluno/coppe para ler tab_derivacoes;
#?
 criar arquivo tes1 com 20 regs de 2 atributos:
#?
 codigo:i(2),
#?
 nome:a(10)
#?
 com chave: codigo;
         2          2
```

```
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES  5 02/06/90
ALUNO            TAB_ENTIDADES     4 02/06/90
ALUNO            TAB_ATRIBUTOS     4 02/06/90
ALUNO            TAB_DERIVACOES    4 02/06/90
ALUNO            TES1              3 02/06/90
#?
 fechar base de dados;
FECHBD, testa flag de rollback          0
FECHBD, testa ultima transacao          1
FIMBI- Testa espaco do BI          50          0
FIMAI - testa espaco do AI         170         5
FIMBI- Testa espaco do BI          50          1
Grava reg. no BI          1          2
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         6
Grava nova pagina no AI           2          2
FIMBI- Testa espaco do BI          50          2
Grava reg. no BI          1         17
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         7
Grava nova pagina no AI           2         17
FIMBI- Testa espaco do BI          50          3
Grava reg. no BI          1         15
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         8
Grava nova pagina no AI           2         15
FIMBI- Testa espaco do BI          50          4
Grava reg. no BI          1          5
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         9
Grava nova pagina no AI           2          5
FIMBI- Testa espaco do BI          50          5
Grava reg. no BE          1         19
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         10
Grava nova pagina no AI           2         19
FIMBI- Testa espaco do BI          50          6
```

```
Grava reg. no BI          1          4
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          11
Brava nova pagina no AI           2           4
FIMBI- Testa espaco do BI         50           7
Grava reg. no BI          1         16
Grava do buffer para a BAÇE
FINAI - testa espaco do AI        170          12
Grava nova pagina no AI           2           16
FIMRI- Testa espaco do BI         50           8
Grava reg. no BI          1         14
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          13
Grava nova pagina no AI           2           14
FIMBI- Testa espaco do BI         50           9
Grava reg. no BI          1        171
Grava do buffer para a BAÇE
FIMAI - testa espaco do AI        170          14
Grava nova pagina no AI           2          171
FIMBI- Testa espaco do BI         50          10
Grava reg. no BI          1         51
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          15
Grava nova pagina no AI           2           51
FIMBI- Testa espaco do BI         50          11
Grava reg. no BI          1         46
Grava do buifer para a BAÇE
FIMAI - testa espaco do AI        170          16
Grava nova pagina no AI           2           46
FIMBI- Testa espaco do BI         50          12
Grava reg. no BI          9         47
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          17
Grava nova pagina no AI           2           47
FIMBI- Testa espaco do BI         50          13
Grava reg. no BI          1         45
Grava do buffer para a BASE
FIMAI - testa espaco do AI        170          18
Grava nova pagina no AI           2           45
```

```
FIMBI- Testa espaco do BI          50          14
Grava reg. no BI          1          6
Grava do buffer para a BASE
FIMAI -- testa espaco do AI          170          19
Grava nova pagina no AI          2          6
FIMBI- Testa espaco do BI          50          15
Grava reg. no BI          1          44
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          20
Grava nova pagina no AI          2          44
FIMBI- Testa espaco do BI          50          16
Grava reg. no BI          1          21
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          21
Grava nova pagina no AI          2          21
FIMBI- Testa espaco do BI          50          17
Grava reg. no BI          1          52
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          22
Grava nova pagina no AI          2          52
FIMBI- Testa espaco do BI          50          18
Grava reg. no BI          1          20
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          23
Grava nova pagina no AI          2          20
FIMBI- Testa espaco do BI          50          19
Grava reg. no BI          1          7
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          24
Grava nova pagina no AI          2          7
FIMBI- Testa espaco do BI          50          19
FIMAI - testa espaco do AI          170          24
FIMAI - testa espaco do AI          170          25
FECHBD grava reg. no AI          25          1          1
FECHBD atualiza o primeiro registro na BASE
FIMAI - testa espaco do AI          170          26
FECHBD,grava reg. TBD no AI          26          5
FIMAI - testa espaco do AI          170          27
FECHBD,grava reg. TUB no AI          27          3
```

```
FIMAI - testa espaco do AI          170          28
FECHBD, grava COMMIT no AI           28           6          0
FECHBD - fechou BI
FECHBD - fechou AI
#?
 fechar sessao;
#?
 enc
Stop - Program terminated.
```

e) Inclusao de registros no arquivo TES1

```
cprel con
#?
 abrir base de dados bdtese;
ABREBD - testa flag de inicio              0
ABREBD - testa ultima transacao               1
ABREBD - abriu AI
contador = tipo do reg 1 do AI             28
ABREBD - posicionou AI, icontai               28
FIMAI - testa espaco do AI          170          28
Leu BI, tipo, pagina                1           2
FIMBI - testa rspaco no BI           50           1
ABREBD grava primeiro registro no BI
ABREBD grava primeiro registro na BASE
grava contador no Iro. registro de AI          28
FIMAI - testa espaco do AI          170          29
ABREBD grava reg. no AI             29           1          1
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES   5 02/06/90
ALUNO            TAB_ENTIDADES      4 02/06/90
ALUNO            TAB_ATRIBUTOS      4 02/06/90
ALUNO            TAB_DERIVACOES     4 02/06/90
ALUNO            TES1               3 02/06/90
H?
 most tes1;

 ins segs em tes1;
```

```
 ENTRE COM OS REGISTROS A INSERIR
??
1/"tes1"//2/"tes2"//3/"tes3"//4/"tes4"//5/"tes5"///
 FORAM INSERIDOS        5 REGISTROS
H?
 most tes1;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 fechar base de dados;
FECHBD, testa flag de rollback              0
FECHBD, testa ultima transacao              2
FIMBI - testa espaco do BI          50              1
FIMAI - testa espaco do AI          170             29
FIMBI - testa espaco do BI          50              2
Grava reg. no BI          2           2
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170             30
Grava nova pagina no AI           2             2
FIMBI - testa espaco do BI          50              3
Grava reg. no BI          2           52
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170             31
Grava nova pagina no AI           2             52
FIMBI - testa espaco do BI          50              4
Grava reg. no BI          2           6
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170             32
Grava nova pagina no AI           2             6
FIMBI - testa espaco do BI          50              5
Grava reg. no BI          2           171
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170             33
Grava nova pagina no AI           2             171
FIMBI - testa espaco do BI          50              6
Grava reg. no BI          2           20
```

```
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          34
Grava nova pagina no AI             2            20
FIMBI - testa espaco do BI          50           6
FIMAI - testa espaco do AI          170          34
FIMAI - testa espaco do AI          170          35
FECHBD grava reg. no AI             35           1          1
FCCHBD atualiza o primeiro registro na BASE
FIMAL - testa espaco do AI          170          36
FECHBD,grava reg. TBD no AI         36           5          1
FIMAI - testa espaco do AI          170          37
FECHBD,grava reg. TUB no AI         37           3          1
FIMAI - testa espaco do AI          170          38
FECHBD, grava COMMIT no AI          38           6          0
FECHBD - fechou BI
FECHBD - fechou AI
#?
 fechar sessao;
#?
 enc
Stop - Program terminated.
```

f) DSBACKUP (opção de "backup")

OBS: Realização de descarga "dump" dos arquivas: BDTRABA, TUB, TBD, BSTRAP, TRACE, SOCORRO, ERROS, SUPDADOS e LOGDADQS. A base de dados salva (BDTRABA) tem somente um arquivo (TES1) com 5 (cinco) registros.

g) Chamada do programa CRIACOM (execução normal do CRIACOM com comprometimento - "commit")

```
cprel criacom
#?
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          - C:\CPREL\BI
No. de paginas alocadas para o BI -          50
```

```
Percentual de paginas  para o BI -    9.000000E-01
Localizacao do arquivo AI         - C:\CPREL\AI
No. de paginas alocadas para o AI -         170
Percentual de paginas  para o AI -    9.000000E-01
#?
 abrir base de dados BDTESE;
ABREBD - testa flag de inicio           0
ABREBD - testa ultima transacao           6
ABREBD - abriu AI
contador = tipo do reg 1 do AI          112
ABREBD - posicionou AI, icontai           112
FIMAI - testa espaco do AI          170          112
Leu BI, tipo, pagina           6            1
FIMBI- Testa espaco do BI           50            1
ABREBD grava primeiro registro no BI
ABREBD grava primeiro registro na BASE
grava contador no Iro. registro de AI          112
FIMAI - testa espaco do AI          170          113
ABREBD grava reg. no AI         113           1            1
#?
 criar arquivo tes2 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2           2
#?
 criar arquivo tes3 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
H?
 com chave : codp e com indice sobre : nomep;
         2           2
#?
 criar arquivo tes4 com 28 registros de 2 atributos:
```

```
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes5 com 28 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes6 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes7 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes8 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
```

```
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes9 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes10 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES  5 02/06/90
ALUNO            TAB_ENTIDADES     4 02/06/90
ALUNO            TAB_ATRIBUTOS     4 02/06/90
ALUNO            TAB_DERIVACOES    4 02/06/90
ALUNO            TES1              3 02/06/90
ALUNO            TES2              3 02/06/90
ALUNO            TES3              3 02/06/90
ALUNO            TES4              3 02/06/90
ALUNO            TES5              3 02/06/90
ALUNO            TES6              3 02/06/90
ALUNO            TES7              3 02/06/90
ALUNO            TES8              3 02/06/90
ALUNO            TES9              3 02/06/90
ALUNO            TES10             3 02/06/90
#?
```

```
 copiar tes1 em tes2;
#?
 copiar tes1 em tes3;
#?
 copiar tes1 em tes4;
#?
 copiar tes1 em tes5;
#?
 copiar tes1 em tes6;
#?
 copiar tes1 em tes7;
#?
 copiar tes1 em tes8;
#?
 copiar tes1 em tes9;
#?
 copiar tes1 em tes10;
#?
 modificar tesá tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI          50          1
FIMAI - testa espaco do AI         170         75
FIMBI- Testa espaço do BI          50          2
Grava reg. no BI          5          55
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         76
Grava nova pagina no AI           2          55
FIMBI- Testa espaco do BI          50          3
Grava reg. no BI          5          6
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         77
Grava nova pagina no AI           2          6
FIMBI- Testa espaco do BI          50          4
Grava reg. no BI          5          2
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         78
Grava nova pagina no Al           2          2
FIMBI- Testa espaco do BI          50          5
Grava reg. no BI          5          171
Grava do buffer para a BASE
```

FIMAI - testa espaco do AI 170 79
Grava nova pagina no AI 2 171
FIMBI- Testa espaco do BI 50 6
Grava reg. no BI 5 15
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 80
Grava nova pagina no AI 2 15
FIMBI- Testa espaco do BI 50 7
Grava reg. no BI 5 47
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 81
Grava nova pagina no AI 2 47
FIMBI- Testa espaco do BI 50 8
Grava reg. no BI 5 54
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 82
Grava nova pagina no AI 2 54
FIMBI- Testa espaco do BI 50 9
Grava reg. no BI 5 52
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 83
Grava nova pagina no AI 2 52
FIMBI- Testa espaco do BI 50 10
Grava reg. no BI 5 4
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 84
Grava nova pagina no AI 2 4
FIMBI- Testa espaco do BI 50 11
Grava reg. no BI 5 14
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 85
Grava nova pagina no AI 2 14
FIMBI- Testa espaco do BI 50 12
Grava reg. no BI 5 56
Grava do buffer para a BASE
FIMAI - testa espaco do AI 170 86
Grava nova pagina no AI 2 56
FIMBI- Testa espaco do BI 50 13
Grava reg. no BI 5 53

```
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          87
Grava nova pagina no AI             2            53
FIMBI- Testa espaco do BI           50           14
Grava reg. no BI          5         17
Erava do buffer para a BASE
FPMAI - testa espaco do AI          170          88
Grava nova pagina no AI             2            17
FIMBI- Testa espaco do BI           50           15
Grava reg. no BI          5         19
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          89
Grava nova pagina no AI             2            19
FIMBI- Testa espaco do BI           50           16
Grava reg. no BI          5         16
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          90
Grava nova pagina no AI             2            16
FIMBI- Testa espaco do BI           50           17
Grava reg. no BI          5         20
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          91
Grava nova pagina no AI             2            20
FIMBI- Testa espaco do BI           50           18
Grava reg. no BI          5         57
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          92
Grava nova pagina no AI             2            57
#?
 modificar tes7 tal que: codp = 5 (nomep para "teste-5");
FPMBI- Testa espaco do BI           50           18
FIMAI - testa espaco do AI          170          92
Grava do buffer para a BASE
Grava mesma pagina no AI            2            55
Grava do buffer para a BASE
Grava mesma pagina no AI            2            6
Grava do buffer para a BASE
Erava mesma pagina no AI            2            2
Grava do buffer para a BASE
```

```
Grava mesma pagina no AI          2          171
Grava do buffer para a BASE
Grava mesma pagina no AI          2           15
Grava do buffer para a BASE
Grava mesma pagina no AI          2           47
Grava do buffer para a BASE
Grava mesma pagina no AI          2           52
Grava do buffer para a BASE
Grava mesma pagina no AI          2            4
Grava do buffer para a BASE
Grava mesma pagina no AI          2           14
Grava do buffer para a BASE
Grava mesma pagina no AI          2           54
Grava do buffer para a BASE
Grava mesma pagina no AI          2           56
Grava da buffer para a BASE
Grava mesma pagina no AI          2           53
Grava do buffer para a BASE
Grava mesma pagina no AI          2           17
Grava do buffer para a BASE
Grava mesma pagina no AI          2           19
Grava do buffer para a BASE
Grava mesma pagina no AI          2           16
Grava do buffer para a BASE
Grava mesma pagina no AI          2           20
Grava do buffer para a BASE
Grava mesma pagina no AI          2           57
#?
 modificar tes8 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI         50           18
FIMAI .. testa espaco do AI       170          92
Grava do buffer para a BASE
Grava mesma pagina no AI          2           56
Grava do buffer para a BASE
Grava mesma pagina no AI          2            6
Grava do buffer para a BASE
Grava mesma pagina no AI          2            2
Grava do buffer para a BASE
Grava mesma pagina no AI          2           55
```

```
Grava do buffer para a BASE
Grava mesma pagina no AI              2          171
Grava do buffer para a BASE
Grava mesma pagina no AI              2           15
Grava do buffer para a BASE
Grava mesma pagina no AI              2           47
Grava do buffer para a BASE
Grava mesma pagina no AI              2           52
Grava do buffer para a BASE
Grava mesma pagina no AI              2            4
Grava do buffer para a BASE
Grava mesma pagina no AI              2           14
Grava do buffer para a BASE
Grava mesma pagina no AI              2           54
Grava do buffer para a BASE
Grava mesma pagina no AI              2           53
Grava do buffer para a BASE
Grava mesma pagina no AI              2           17
Grava do buffer para a BASE
Grava mesma pagina no AI              2           19
Grava do buffer para a BASE
Grava mesma pagina no AI              2           16
Grava do buffer para a BASE
Grava mesma pagina no AI              2           20
Grava do buffer para a BASE
Grava mesma pagina no AI              2           57
#?
 modificar tes9 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI             50          18
FIMAI - testa espaco do AI            170          92
Grava do buffer para a BASE
Grava mesma pagina no AI              2           56
Grava do buffer para a BASE
Grava mesma pagina no AI              2            2
Grava do buffer para a BASE
Grava mesma pagina no AI              2           47
Grava do buffer para a BASE
Grava mesma pagina no AI              2            6
Grava do buffer para a BASE
```

```
Grava mesma pagina no AI          2          171
Grava do buffer para a BASE
Grava mesma pagina no AI          2            4
Grava do buffer para a BASE
Grava mesma Pagina no AI          2           15
Grava do buffer para a BASE
Grava mesma Pagina no AI          2           52
Grava do buffer para a BASE
Grava mesma pagina no AI          2           14
Grava do buffer para a BASE
Grava mesma pagina no AI          2           55
Grava do buffer para a BASE
Grava mesma pagina no AI          2           54
Grava do buffer para a BASE
Grava mesma pagina no AI          2           53
Grava do buffer para a BASE
Grava mesma pagina no AI          2           17
Grava do buffer para a BASE
Grava mesma pagina no AI          2           19
Grava do buffer para a BASE
Grava mesma pagina no AI          2           16
Grava do buffer para a BASE
Grava mesma pagina no AI          2           28
Grava do buffer para a BASE
Grava mesma pagina no AI          2           57
#?
 modificar tes10 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI         50           18
FIMAI - testa espaco do AI        170          92
Grava do buffer para a BASE
Grava mesma pagina no AI          2           56
Grava do buffer para a BASE
Grava mesma pagina no AI          2            6
Grava do buffer para a BASE
Grava mesma pagina no AI          2            2
Grava do buffer para a BASE
Grava mesma pagina no AI          2          171
Grava do buffer para a BASE
Grava mesma pagina no AI          2           47
```

```
Grava do buffer para a BASE
Grava mesma pagina no AI          2          15
Grava do buffer para a BASE
Grava mesma pagina no AI          2          52
Grava do buffer para a BASE
Grava mesma pagina no AI          2          14
Grava do buffer para a BASE
Grava mesma pagina no AI          2           4
Grava do buffer para a BASE
Grava mesma pagina no AI          2          55
Grava do buffer para a BASE
Grava mesma pagina no AI          2          54
Grava do buffer para a BASE
Grava mesma pagina no AI          2          53
Grava do buffer para a BASE
Grava mesma pagina no AI          2          17
Grava do buffer para a BASE
Grava mesma pagina no AI          2          19
Grava do buffer para a BASE
Grava mesma pagina no AI          2          16
Grava do buffer para a BASE
Grava mesma pagina no AI          2          20
Grava do buffer para a BASE
Grava mesma pagina no AI          2          57
#?
 most tes1;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESS
#?
 most tes2;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
```

```
most tes3;
01 TES1
02 TES2
03 TES3
04 TES4
05 TES5
#?
most te54;
01 TES1
02 TES2
03 TES3
04 TES4
05 TES5
#?
most tes5;
01 TES1
82 TES2
03 TES3
04 TES4
05 TES5
#?
most tes6;
01 TES1
02 TES2
03 TES3
04 TES4
05 TESTE-5
#?
most tes7;
01 TES1
02 TES2
03 TES3
04 TES4
05 TESTE-5
#?
most tes8;
01 TES1
02 TES2
03 TES3
```

```
 04 TES4
 05 TESTE-5
H?
 most tes9;
 01 TES1
 02 TES2
 03 TES3
 84 TES4
 05 TESTE-5
H?
 most tes10;
 01 TES1
 82 TES2
 03 TES3
 04 TES4
 05 TESTE-5
#?
 ret regs de tes2 tal que codp = 4;
#?
 ret regs de tes3 tal que codp = 4;
#?
 ret regs de tes4 tal que codp = 4;
#?
 ret regs de tes5 tal que codp = 4;
#?
 ret regs de tes6 tal que codp = 4;
#?
 ret regs de tes7 tal que codp = 4j
#?
 ret regs de tes8 tal que codp = 4;
#?
 ret regs de tes9 tal que codp = 4;
#?
 ret regs de tes10 tal que codp = 4;
#?
 most tes1;
 01 TES1
 02 TES2
 03 TES3
```

```
 04 TES4
 05 TES5
#?
 rnost tes2;
 01 TES1
 02 TES2
 03 TES3
 05 TESS
#?
 most tes3;
 01 TES1
 02 TES2
 03 TES3
 05 TES5
#?
 most tes4;
 01 TES1
 02 TES2
 03 TES3
 05 TCSS
#?
 most tes5;
 01 TES1
 02 TES2
 03 TES3
 05 TCS5
#?
 most tes6;
 01 TES1
 02 TCS2
 03 TES3
 05 TESTE-5
#?
 most tes7;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
```

```
 most tes8;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes9;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes10;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES  5 02/06/90
ALUNO            TAB_ENTIDADES     4 02/06/90
ALUNO            TAB_ATRIBUTOS     4 02/06/90
ALUNO            TAB_DERIVACOES    4 02/06/90
ALUNO            TES1              3 02/06/90
ALUNO            TES2              3 02/06/90
ALUNO            TES3              3 02/06/90
ALUNO            TES4              3 02/06/90
ALUNO            TES5              3 02/06/90
ALUNO            TES6              3 02/06/90
ALUNO            TES7              3 02/06/90
ALUNO            TES8              3 02/06/90
ALUNO            TES9              3 02/06/90
ALUNO            TES18             3 02/06/90
#?
 fechar base de dados;
FECHBD, testa flag de rollback          0
FECHBD,,testa ultima transacao          5
FIMBI- Testa espaco do BI          50          18
FIMAI - testa espaco do AI         170          92
```

Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 2
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 17
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 15
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 4
Grava do buffer para a **BASE**
Grava mesma Pagina no **AI** 2 6
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 171
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 56
Grava do buffer para a **BASE**
Grava mesma pagina no AI 2 14
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 55
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 54
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 53
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 52
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 47
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 19
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 16
Grava do buffer para a **BASE**
Grava mesma pagina no AI 2 20
Grava do buffer para a **BASE**
Grava mesma pagina no **AI** 2 57
FIMBI- Testa espaco do **BI** 50 18
FIMAI - testa espaco do **AI** 170 92
FIMAI - testa espaco do AI 170 93
FECHBD grava reg. no **AI** 93 1 1
FECHBD atualiza o primeiro registro na **BASE**

```
FIMAI - testa espaco do AI          170          94
FECHBD,grava reg. TBD no AI          94           5          1
FIMAI - testa espaco do AI          170          95
FECHBD,grava reg. TUB no AI          95           3          1
FIMAI - testa espaco do AI          170          96
FECHBD, grava COMMIT no AI           96           6          0
FECHBD - fechou BI
FECHBD - fechou AI
#?
 fechar sessao;
#?
 enc
Stop - Program terminated.
```

h) Verifica a criação e modificação das nove tabelas (tes2-tes10) e simula uma falha de meio de armarenamento

```
cprel con
#?
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          - C:\CPREL\BI
No. de paginas alocadas para o BI -          50
Percentual de paginas  para o BI -    9.000000E-01
Localizacao do arquivo AI          - C:\CPREL\AI
No. de paginas alocadas para o AI -         170
Percentual de paginas  para o AI -    9.000000E-01
#?
 abrir base de dados bdteçe;
ABREBD - testa flag de inicio            0
ABREBD - testa ultima transacao          3
ABREBD - abriu AI
contador = tiro do reg 1 do AI          60
ABREBD - posicionou AI, icontai          60
FIMAI - testa espaco do AI          170          60
Leu BI, tipo, pagina          3           1
FIMBI- Testa espaco do BI           50           1
ABREBD grava primeiro registro no BI
ABREBD grava primeiro registro na BASE
grava contador no Iro. registro de AI          60
```

```
FIMAI - testa espaco do AI          170          61
ABREBD grava reg. no AI          61            1            1
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES  5 03/06/90
ALUNO            TAB_ENTIDADES     4 03/06/90
ALUNO            TAB_ATRIBUTOS     4 03/06/90
ALUNO            TAB_DERIVACOES    4 03/06/90
ALUNO            TES1              3 03/06/90
ALUNO            TES2              3 03/06/90
ALUNO            TES3              3 03/06/90
ALUNO            TES4              3 03/06/90
ALUNO            TES5              3 03/06/90
ALUNO            TES6              3 03/06/90
ALUNO            TES7              3 03/06/90
ALUNO            TES8              3 03/06/90
ALUNO            TES9              3 03/06/90
ALUNO            TES10             3 03/06/90
#?
 most tes1;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESS
#?
 most tes2;
 01 TES1
 02 TES2
 03 TES3
 05 TES5
#?
 most tes3;
 01 TES1
 02 TES2
 03 TES3
 05 TES5
#?
 most tes4;
```

```
 01 TES1
 02 TES2
 03 TES3
 05 TES5
#?
 most tes5;
 01 TES1
 02 TCS2
 03 TES3
 05 TES5
#?
 most tes6;
 01 TES1
 02 TES2
 03 TES3
 05 TCSTE-5
#?
 most tes7;
 01 TES1
 02 TCS2
 03 TES3
 05 TESTE-5
#?
 most tes8;
 01 TES1
 82 TES2
 03 TES3
 05 TESTE-5
#?
 most tes9;
 01 TES1
 02 TCS2
 03 TES3
 05 TESTE-5
#?
 most tes10;
 01 TES1
 02 TES2
 03 TES3
```

```
 05 TESTE-5
#?
 fechar base de dados;
FECHBD, testa flag de rollback          0
FECHBD, testa ultima transacao          4
FIMBI- Testa espaco do BI          50          1
FIMAI - testa espaco do AI         170         61
FIMBE- Testa espaco do BI          50          2
Grava reg. no BI          4          2
Grava do buffer para a BASE
```

*** silmulação de falha de meio de armazenamento (Control+Alt+Del, seguido da deleção dos arquivos que foram salvos na Última descarga.

i) DSBACKUP (opção de "restare")

OBS: Aplicação da Última descarga dos arquivos: BDTRABA, TUB, TBD, BSTRAP, TRACE, SOCORRO, ERROS, SUPDADOS e LOGDADOS. A base de dados salva (BDTRABA) tem somente um arquivo (TES1) com 5 (cinco) registros.

j) Chamada do programa FAMEIO (faz a recuperação de falha de meio após a aplicação da última descarga)

```
fameio

Localizacao do arquivo AI          _ C:\CPREL\AI
No. de paginas alocadas para o AI -          170
Percentual de paginas  para o AI -    9.000000E-01
FAMEIO - abriu AI
FAMEIO - leu AI          1          1
Entrou no if tipo=1, inicok=1          1          1
FAMEIO - leu AI          5          1
FAMEIO - leu AI          4          2
FAMEIO - leu AI          2          2
FAMEIO - leu AI          2         17
FAMEIO - leu AI          2         15
FAMEIO - leu AI          2          5
```

FAMEPO - leu AI 2 19
FAMEIO - leu AI 2 4
FAMEIO - leu AI 2 16
FAMEIO - leu AI 2 14
FAMEIO - leu AI 2 171
FAMEIO - leu AI 2 51
FAMEIO - leu AI 2 46
FAMEIO - leu AI 2 47
FAMEIO - leu AI 2 45
FAMEIO - leu AI 2 6
FAMEIO - leu AI 2 44
FAMEIO - leu AI 2 21
FAMEIO - leu AI 2 52
FAMEIO - leu AI 2 20
FAMEIO - leu AI 2 7
FAMEIO - leu AI 1 1
FAMEIO - leu AI 5 1
FAMEIO - leu AI 3 1
FAMEIO - leu AI 6 0
Rotina 100 - leu AI 1ro reg do BDTRABA
Rotina 100 - recuperou Iro reg do BDTRABA
550 - testa se quebrou registro 27 3
Rotina 500 - leu AI registro do TBD
Rotina 500 - recuperou reg do TBD
550 - testa çe quebrou registro 27 4
Rotina 400 - leu AI reg normal do TUB
Rotina 400 - recuperou reg normal do TUB
550 - testa se quebrou registro 27 5
Rotina 208 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 6
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 7
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 8
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA

550 - testa se quebrou registro 27
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 10
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 11
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 12
Rotina 280 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 13
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 208 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 14
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 15
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 16
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 17
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 18
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 19
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg narmal do BDTRABA
550 - testa se quebrou resistro 27 20
Rotina 200 - leu AI reg normal do BDTRABA
Rolina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 27 21
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA

```
550 - testa se quebrou registro          27          22
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          27          23
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
530 - testa se quebrou registro          27          24
Rotina 100 - leu AI Iro reg do BDTRABA
Rotina 100 - recuperou 1ro reg do BDTRABA
550 - testa se quebrou registro          27          25
Rotina 500 - leu AI registro do TBD
Rotina 500 - recuperou reg do TBD
550 - testa çe quebrou registro          27          26
Rotina 300 - leu AI 1ro reg do TUB
Rotina 300 - recuperou lro reg do TUB
550 - testa se quebrou registro          27          27
550 - nao houve quebra
FAMEIO - leu AI          1          1
Entrou no if tipo=1, inicok=1          1          1
FAMEIO - leu AI          2          2
FAMEIO - leu AI          2         52
FAMEIO - leu AI          2          6
FAMEIO - leu AI          2        171
FAMEIO - leu AI          2         20
FAMEIO - leu AI          1          1
FAMCIO - leu AI          5          i
FAMEIO - leu AI          3          1
FAMEIO - leu AI          6          0
Rotina 100 - leu AI iro reg do BDTRABA
Rotina 100 - recuperou 1ro reg do BDTRABA
550 - testa se quebrou registro          37          29
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          37          30
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          37          31
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
```

```
530 - testa se quebrou registro                37          32
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro                37          33
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro                37          34
Rotina 100 - leu AI 1ro reg do BDTRABA
Rotina 100 - recuperou 1ro reg do BDTRABA
550 - testa se quebrou registro                37          35
Rotina 500 - leu AI registro do TBD
Rotina 500 - recuperou reg do TBD
550 - testa se quebrou registro                37          36
Rotina 300 - leu AI 1ro reg do TUB
Rotina 3@0 - recuperou 1ro reg do TUB
550 - testa se quebrou registro                37          37
550 - nao houve quebra
FAMEIO - leu AI                 1              1
Entrou no if tipo=1, inicok=1                  1           1
FAMEIO - leu AI                 2             55
FAMEIO - leu AI                 2              6
FAMEIO - leu AI                 2              2
FAMEIO - leu AI                 2            171
FAMEIO - leu AI                 2             15
FAMEIO - leu AI                 2             47
FAMEIO - leu AI                 2             54
FAMEIO - leu AI                 2             52
FAMEIO - leu AI                 2              4
FAMEIO - leu AI                 2             14
FAMEIO - leu AI                 2             56
FAMEIO - leu AI                 2             53
FAMEIO - leu AI                 2             17
FAMEIO - leu AI                 2             19
FAMEIO - leu AI                 2             16
FAMEIO - leu AI                 2             20
FAMEIO - leu AI                 2             57
FAMEIO - leu AI                 1              1
FAMEIO - leu AI                 5              1
FAMEIO - leu AI                 3              1
```

FAMEIO - leu AI 6 0
Rotina 100 - leu AI Iro reg do BDTRABA
Rotina 100 - recuperou 1ro reg do BDTRABA
550 - testa se quebrou registro 59 39
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 59 40
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 59 41
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 59 42
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 59 43
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou resistro 59 44
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 59 45
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 59 46
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 59 47
Rutina 200 - leu AI reg normal do BDTRABA
Rutrna 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 59 48
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 59 49
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro 59 50
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA

```
550 - testa se quebrou registro          59          51
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          59          52
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          59          53
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          59          54
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          59          55
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          59          56
Rotina 100 - leu AI 1ro reg do BDTRABA
Rotina 100 - recuperou lro reg do BDTRABA
550 - testa se quebrou registro          59          57
Rotina 500 - leu AI registro do TBD
Rotina 500 - recuperou reg do TBD
550 - testa se quebrou registro          59          58
Rotina 300 - leu AI 1ro reg do TUB
Rotina 300 - recuperou Iro reg do TUB
550 - testa se quebrou registro          59          59
550 - nao houve quebra
FAMEIO - leu AI          1          1
Entrou no if tipo=1, inicok=1          1          1
FAMEIO - leu AI          2          2
FAMEIO - leu AI          2        171
FAMEIO - leu AI          2         20
FAMEIO - leu AI          1          1
FAMEIO - leu AI          5          1
FAMEIO - leu AI          3          1
FAMEIO - leu AI          6          0
Rotina 100 - leu AI 1ro reg do BDTRABA
Rotina 100 - recuperou lro reg do BDTRABA
550 - testa se quebrou registro          67          61
Rotrna 200 - leu AI reg normal do BDTRABA
```

```
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          67          62
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          67          63
Rotina 200 - leu AI reg normal do BDTRABA
Rotina 200 - recuperou reg normal do BDTRABA
550 - testa se quebrou registro          67          64
Rotina 100 - leu AI 1ro reg do BDTRABA
Rotina 100 - recuperou 1ro reg do BDTRABA
550 - testa se quebrou registro          67          65
Rotina 500 - leu AI registro do TBD
Rotina 500 - recuperou reg do TBD
330 - testa se quebrou registro          67          66
Rotina 300 - leu AI 1ro reg do TUB
Rotina 300 - recuperou 1ro reg do TUB
550 - testa se quebrou registro          67          67
550 - nao houve quebra
FANEIO - leu AI          0          0
FIM DA RECUPERACAO DE FALHA DE MEIO
INICIALIZACAO DO BI
FIM DA INICIALIZACAO DO BI
Stop - Program terminated.
```

k) Verifica se a base de dados foi devidamente recuperada

```
cprel con
#?
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          - C:\CPREL\BI
No. de paginas alocadas para o BI -          50
Percentual de paginas  para o BI -    9.000000E-01
Localizacao do arquivo AI          - C:\CPREL\AI
No. de paginas alocadas para o AI -         170
Percentual de paginas  para o AI -    9.000000E-01
#?
 abrir base de dados bdtese;
ABREBD - testa flag de inicio           0
```

```
ABREBD - testa ultima transacao            0
ABREBD -- abriu AI
contador = tipo do reg 1 do AI           68
ABREBD - posicionou AI, icontai            68
FIMAI - testa espaco do AI          170          68
Leu BI, tipo, pagina          0          0
FIMBI- Testa espaco do BI           50          1
ABREBD grava primeiro registro no BI
ABREBD grava primeiro registro na BASE
grava contador no 1ro. registro de AI          68
FIMAI - testa espaco do AI          170          69
ABREBD grava reg. no AI          69          1          1
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES  5 03/06/90
ALUNO            TAB_ENTIDADES     4 03/06/90
ALUNO            TAB_ATRIBUTOS     4 03/06/90
ALUNO            TAB_DERIVACOES    4 03/06/90
ALUNO            TES1              3 03/06/90
ALUNO            TES2              3 03/06/90
ALUNO            TES3              3 03/06/90
ALUNO            TES4              3 03/06/90
ALUNO            TESS              3 03/06/90
ALUNO            TES6              3 03/06/90
ALUNO            TES7              3 03/06/90
ALUNO            TES8              3 03/06/90
ALUNO            TES?              3 03/06/90
ALUNO            TES10             3 03/06/90
#?
 most tes1;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 most tes2;
 01 TES1
 02 TES2
```

```
 03 TES3
 05 TESS
#?
 most tes3;
 01 TES1
 02 TES2
 03 TES3
 05 TES5
#?
 most tes4;
 01 TES1
 02 TES2
 03 TES3
 05 TESS
#?
 most tes5;
 01 TES1
 02 TES2
 03 TES3
 05 TCS5
#?
 most tes6;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes7;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes8;
 01 TES1
 02 TES2
 03 TES3
 05 TCSTE-5
#?
```

```
 most tes9;
 01 TES1
 02 TESE
 03 TES3
 05 TESTE-5
#?
 most tes10;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 fechar base de dados;
FECHBD, testa flag de rollback          0
FECHBD, testa ultima transacao          1
FIMBI- Testa espaco do BI          50          1
FIMAI - testa esraco do AI         170         69
FIMBT- Testa espaco do BI          50          2
Grava reg. no BI          1          2
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         70
Grava nova pagina no AI          2          2
FIMBI- Testa espaco do BI          50          3
Grava reg. no BI          1        171
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         71
Grava nova pagina no AI          2        171
FIMBI- Testa espaco do BI          50          4
Grava reg. no BI          1         20
Grava do buffer para a BASE
FIMAI - testa espaco do AI         170         72
Grava nova pagina no AI          2         20
FIMBI- Testa espaco do BI          50          4
FIMAI - testa espaco do AI         170         72
FIMAI - testa espaco do AI         170         73
FECHBD grava reg. no AI         73          1          1
FECHBD atualiza o primeiro registro na BASE
FIMAI - testa espaco do AI         170         74
FECHBD,grava reg. TBD no AI         74          5          1
```

```
FIMAI - testa espaco do AI          170          75
FECHBD,grava reg. TUB no AI          75           3          1
FIMAI - testa espaco do AI          170          76
FECHBD, grava COMMIT no AI           76           6          0
FECHBD - fechou BI
FECHBD - fechou AI
#?
 fechar sessao;
#?
 enc
Stop - Program terminated.
```

4 - Simulação de Falta de Espaço nos Arquivos BI e AI, e Alocação do AI em Outro Dispositivo

Para tal simulacão modificou-se no arquivo LOGDADOS os valores referentes aos tamanhos dos arquivos BI e AI de forma que o espaço alocado se esgote rapidamente (BI=15 e AI=50). Para mostrar a flexibilidade quanto a localização dos arquivos BI e AI, modificou-se o dispositivo do arquivo AI (B:\AI).

a) Chamada do programa INIBDT (inicializa a base de dados)

```
inibdt
Stop - Program terminated.
```

b) Chamada do programa INILOG (inicializa os diários BI e AI)

```
inilog
   ******* INICIALIZACAO DO BI *******

   Localizacao do arquivo BI         - C:\CPREL\BI
   No. de paginas alocadas para o BI -          15

   ************************************

   ******* INICIALIZACAO DO AI *******

   Localizacao do arquivo AI         - B:\AI
   No. de paginas alocadas para o AI -          50

   ************************************
   FIM DO INILOG
Stop - Program terminated.
```

c) Chamada do programa SUPESQ (inicializa o super-esquema)

```
supesq
Stop - Program terminated.
```

d) Chamada do programa **CRIATESE** (cria uma base de dados)

```
cprel criatese
#?
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          - C:\CPREL\BI
No. de paginas alocadas para o BI -          15
Percentual de paginas   para o BI -    9.000000E-01
Localizacao do arquivo AI          - B:\AI
No. de paginas alocadas para o AI -          50
Percentual de paginas   para o AI -    9.000000E-01
#?
 criar base de dados BDTESE em F1 e F2 para 50 regs por
 arq;
ABREBD - abriu AI
CRIABD - posicionou AI, icontai            2
FIMAI - testa espaco do AI          50           2
ABREBD - abriu BI
Leu BI, tipo, pagina           0           0
Recupera o super-esquema - SUPES1
CRIABD - grava primeiro registro na BASE
grava 1o reg. do AI, proximo registro           2
FIMAI - testa espaco do AI          50           3
grava 1o reg da BASE no AI           3           1         1
FIMAI - testa espaco do AI          50           4
CRIABD,grava reg. TBD no AI          4           5         1
FIMAI - testa espaco do AI'         50           5
CRIABD,grava reg. TUB no AI          5           4         2
H?
 autorizar usuario aluno/coppe para ler tab_entidades;
#?
 autorizar usuario aluno/coppe para remover
tab_autorizacoes;
#?
 autorizar usuario aluno/coppe para ler tab_atributos;
```

```
#?
 autorizar usuario aluno/coppe para ler tab_derivacoes;
#?
 criar arquivo tes1 com 20 regs de 2 atributos:
#?
 codigo:i(2),
#?
 nome:a(10)
#?
 com chave: codigo;
         2          2
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES  5 03/06/90
ALUNO            TAB_ENTIDADES     4 03/06/90
ALUNO            TAB_ATRIBUTOS     4 03/06/90
ALUNO            TAB_DERIVACOES    4 03/06/90
ALUNO            TES1              3 03/06/90
#?
 fechar base de dados;
FECHBD, testa flag de rollback          0
FECHBD, testa ultima transacao          1
FIMBI- Testa espaco do BI          15          0
FIMAI - testa espaco do AI          50          5
FIMBI- Testa espaco do BI          15          1
Grava reg. no BI          1          2
Grava do buffer para a BASE
FIMAI - testa espaco do AI          50          6
Grava nova pagina no AI          2          2
FIMBI- Testa espaco do BI          15          2
Grava reg. no BI          1          17
Grava do buffer rara a BASE
FIMAI - testa espaco do AI          50          7
Grava nova pagina no AI          2          17
FIMBI- Testa espaco do BI          15          3
Grava reg. no BI          1          15
Grava do buffer para a BASE
FIMAI - testa espaco do AI          50          8
Grava nova pagina no AI          2          15
```

FIMBI- Testa espaco do BI 15 4
Grava reg. no BI 1 5
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 9
Grava nova pagina no AI 2 5
FIMBI- Testa espaco do BI 15 5
Grava reg. no BI 1 19
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 10
Grava nova pagina no AI 2 19
FIMBI- Testa espaco do BI 15 6
Grava reg. no BI 1 4
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 11
Grava nova pagina no AI 2 4
FIMBI- Testa espaco do BI 15 7
Grava reg. no BI 1 16
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 12
Grava nova pagina no AI 2 16
FIMBI- Testa espaco do BI 15 8
Grava reg. no BI 1 14
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 13
Grava nova pagina no AI 2 14
FIMBI- Testa espaco do BI 15 9
Grava reg. no BI 1 171
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 14
Grava nova pagina no AI 2 171
FIMBI- Testa espaco do BI 15 10
Grava reg. no BI 1 51
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 15
Grava nova pagina no AI 2 51
FIMBI- Testa espaco do BI 15 11
Grava reg. no BI 1 46
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 16

Grava nova pagina no AI 2 46
FIMBI- Testa espaco do BI 15 12
Grava reg. no BI 1 47
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 17
Grava nova pagina no AI 2 47
FIMBI- Testa espaco do BI 15 13

LOG BI CHEIO, voce tem 3 OPCOES:

1 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando que o numero de paginas que ainda restam no LOG eh igual a: 2

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG BI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG BI ***

Entre com a opcao: 1, 2 ou 3 - 1

Grava reg. no BI 1 45
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 18
Grava nova pagina no AI 2 45
FIMBI- Testa espaco do BI 15 14

LOG BI CHEIO, voce tem 3 OPCOES:

1 - Fica a seu criterio FECHAR a base de dados

ATUALIZANDO ou **SEM ATUALIZAR.** Lembrando que o numero de paginas que ainda restam no **LOG** eh igual a: 1

2 - ABORTA sua transacao **(FALHA DE SISTEMA).** No proximo reinicio os efeitos desta transacao serao **DESFEITOS.**

3 - TRABALHA SEM LOG BI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu **LOG BI** ***

Entre com a opcao: 1, 2 ou 3 - 1

Grava reg. no **BI**	1	6
Grava do buffer para a **BASE**		
FIMAI - testa espaco do **AI**	50	19
Grava nova pagina no **AI**	2	6
FIMBI- Testa espaco do **BI**	15	15

LOG BI CHEIO, voce tem **3 OPCOES:**

1 - Fica a seu criterio **FECHAR** a base de dados **ATUALIZANDO** ou **SEM ATUALIZAR.** Lembrando que o numero de paginas que ainda restam no LOG eh igual a: 0

2 - ABORTA sua transacao **(FALHA DE SISTEMA).** No proximo reinicio os efeitos desta transacao serao **DESFEITOS.**

3 - TRABALHA SEM LOG BI e SEM GARANTIAS

*** **Seja** qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu **LOG BI** ***

Entre com a opcao: 1, 2 ou 3 - 1

Opcao ▮ invalida, nao existem mais paginas

LOG BI CHEIO, voce tem 3 OPCOES:

1 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando que o numero de paginas que ainda restam no LOE eh igual a: 0

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG BI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG BI ***

Entre com a opcao: 1, 2 ou 3 - 3

Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 20
Grava nova pagina no AI 2 44
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 21
Grava nova pagina no AI 2 21
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 22
Grava nova pagina no AI 2 52
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 23
Grava nova pagina no AI 2 20
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 24

```
Grava nova pagina no AI                2              7
FIMAI -- testa espaco do AI           50             24
FIMAI - testa espaco do AI            50             25
FECHBD grava reg. no AI               25              1              1
FECHBD atualiza o primeira registro na BASE
FIMAI - testa espaco do AI            50             26
FECHBD,grava reg. TBD no AI           26              5              1
FIMAI - testa espaco do AI            50             27
FECHBD,grava reg. TUB no AI           27              3              1
FIMAI - testa espaco do AI            50             28
FECHBD, grava COMMIT no AI            28              6              0
FECHBD -- .fechou AI
#?
 fechar sessao;
#?
 enc
Stop - Program terminated.
```

e) Inclusao de registros no arquivo TES1

```
cprel con
#?
 abrir sessao para us aluno/coppe;
Localizacao do arquivo BI          - C:\CPREL\BI
Mo. de paginas alocadas para o BI -          50
Percentual de paginas  para o BI -    9.000000E-01
Localizacao do arquivo AI          - C:\CPREL\AI
No. de paginas alocadas para o AI -         170
Percentual de paginas  para o AI -    9.000000E-01
#?
 abrir base de dados BDTESE;
ABREBD - testa flag de inicio              0
ABRERD - testa ultima transacao               1
ABREBD - abriu AI
contador = tipo do reg 1 do AI            28
ABREBD - posicionou AI, icontai              28
FIMAI - testa espaco do AI           170            28
Leu BI, tipo, pagina                 1             2
```

```
FIMBI - testa espaco no BI          50          1
ABREBD grava primeiro registro no BI
ABREBD grava primeiro registro na BASE
grava contador no Iro. registro de AI          28
FIMAI - testa espaco do AI          170          29
ABREBD grava reg. no AI          29          1          1
#?
 most tab_autorizacoes;
ALUNO          TAB_AUTORIZACOES    5 02/06/90
ALUNO          TAB_ENTIDADES       4 02/06/90
ALUNO          TAB_ATRIBUTOS       4 02/06/90
ALUNO          TAB_DERIVACOES      4 02/06/90
ALUNO          TES1                3 02/06/90
H?
 most tes1;

 ins segs em tes1;
 ENTRE COM OS REGISTROS A INSERIR
??
1/"tes1"//2/"tes2"//3/"tes3"//4/"tes4"//5/"tes5"///
 FORAM INSERIDOS        5 REGISTROS
#?
 most tes1;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 fechar base de dados;
FECHBD, testa flag de rollback          0
FECHBD, testa ultima transacao          2
FIMBI - testa espaco do BI          50          1
FIMAI - testa espaco do AI          170          29
FTMBI - testa espaco do BI          50          2
Grava reg. no BI          2          2
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170          30
Crava nova pagina no AI          2          2
```

```
FIMBI - testa espaco do BI          50              3
Grava reg. no BI          2           52
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170             31
Grava nova pagina no AI          2             52
FIMBI - testa espaco do BI          50              4
Grava reg. no BI          2           6
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170             32
Grava nova pagina no AI          2             6
FIMBI - testa espaco do BI          50              5
Grava reg. no BI          2           171
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170             33
Grava nova pagina no AI          2             171
FIMBI - testa espaco do BI          50              6
Grava reg. no BI          2           20
Grava do buffer para a BASE
FIMAI - testa espaco do AI          170             34
Grava nova pagina no AI          2             20
FIMBI - testa espaco do BI          50              6
FIMAI - testa espaco do AI          170             34
FIMAI - testa espaco do AI          170             35
FECHBD grava reg. no AI          35             1          1
FECHBD atualiza o primeiro registro na BASE
FIMAI - testa espaco do AI          170             36
FECHBD,grava reg. TBD no AI          36              5      1
FIMAI - testa espaco do AI          170             37
FECHBD,grava reg. TUB no AI          37              3      1
FIMAI - testa espaco do A1          170             38
FECHBD, grava COMMIT no AI          38              6      0
FECHBD - fechou BI
FECHBD - fechou AI
#?
 fechar sessao;
#?
 enc
Stop - Program terminated.
```

f) Chamada do programa **CRIACOM** (execução normal do CRIACOM com comprometimento - "commit")

```
cprel criacom
#?
 abrir sessao para us aluno/coppe;
Localiracao do arquivo BI          - C:\CPREL\BI
No. de paginas alocadas para o BI -          15
Percentual de paginas   para o BI -    9.000000E-01
Localizacao do arquivo AI          - B:\AI
No. de paginas alocadas para o AI -          50
Percentual de paginas   para o AI -    9.000000E-01
#?
 abrir base de dados BDTESE;
ABREBD - testa flag de inicio            0
ABREBD - testa ultima transacao            2
ABREBD - abriu AI
contador = tipo do reg 1 do AI           38
ABREBD - posicionou AI, icontai            38
FIMAI - testa espaco do AI          50           38
Leu BI, tipo, pagina          2           1
FIMBI- Testa espaco do BI          15           1
ABREBD grava primeiro registro no BI
ABREBD grava primeiro registro na BASE
grava contador no 1ro. registro de AI          38
FIMAI - testa espaco do AI          50          39
ABREBD grava reg. no AI          39           1           1
#?
 criar arquivo tes2 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2          2
#?
 criar arquivo tes3 com 20 registros de 2 atributos:
```

```
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquiva tes4 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 MOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes5 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes6 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
 NOMEP  : A(10)
#?
 com chave : codp e com indice sobre : nomep;
          2           2
#?
 criar arquivo tes7 com 20 registros de 2 atributos:
#?
 CODP   : I(2),
#?
```

```
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2           2
#?
 criar arquivo tes8 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2           2
#?
 criar arquivo tes9 com 20 resistros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMEP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2           2
#?
 criar arquivo tes10 com 20 registros de 2 atributos:
#?
 CODP  : I(2),
#?
 NOMCP : A(10)
#?
 com chave : codp e com indice sobre : nomep;
         2           2
#?
 most tab_autorizacoes;
ALUNO            TAB_AUTORIZACOES   5 03/06/90
ALUNO            TAB_ENTIDADES      4 03/06/90
ALUNO            TAB_ATRIBUTOS      4 03/06/90
ALUNO            TAB_DERIVACOES     4 03/06/90
ALUNO            TES1               3 03/06/90
ALUNO            TES2               3 03/06/90
```

```
ALUNO                TES3                3 03/06/90
ALUNO                TES4                3 03/06/90
ALUNO                TES5                3 03/06/90
ALUNO                TES6                3 03/06/90
ALUNO                TES7                3 03/06/90
ALUNO                TES8                3 03/06/90
ALUNO                TES9                3 03/06/90
ALUNO                TES10               3 03/06/90
#?
 copiar tesl em tes2;
#?
 copiar tesl em tes3;
#?
 copiar tesl em tes4;
#?
 copiar tesl em tes5;
#?
 copiar tesl em tes6;
#?
 copiar tesl em tes7;
#?
 copiar tesl em tes8;
#?
 copiar tesl em tes9;
#?
 copiar tesl em tes10;
#?
 modificar tes6 tal que: codp = 5 (nomep para "teste-5");
FIMBI- Testa espaco do BI          15            1
FIMAI - testa espaco do AI          50           39
FIMBI- Testa espaco do BI          15            2
Grava reg. no BI            3          55
Grava do buffer para a BASE
FIMAI - testa espaco do AI          50           40
Grava nova pagina no AI           2           55
FIMBI- Testa espaco do BI          15            3
Grava reg. no BI            3           6
Grava do buffer para a BASE
FIMAI - testa espaco do AI          50           41
```

```
Grava nova pagina no AI          2          6
FIMBI- Testa espaco do BI       15          4
Grava reg. no BI          3          2
Grava do buffer para a BASE
FIMAI - testa espaco do AI      50         42
Grava nova pagina no AI          2          2
FIMBI- Testa espaco do BI       15          5
Grava reg. no BI          3        171
Grava do buffer para a BASE
FIMAI - testa espaco do AI      50         43
Grava nova pagina no AI          2        171
FIMBI- Testa espaco do BI       15          6
Grava reg. no BI          3         15
Grava do buffer para a BASE
FIMAI - testa espaco do AI      50         44
```

LOG AI CHEIO, voce tem 3 OPCOES:

1 - Fica a seu criterio FECHAR a' base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando que o numero de paginas que ainda restam no LOG eh igual a: 6

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG AI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG AI ***
*** Avalie a possibilidade de fazer BACKUP ***

Entre com a opcao: 1, 2 ou 3 - 1

```
Grava nova pagina no AI          2         15
FIMBI- Testa espaco do BI       15          7
```

Grava reg. no BI 3 47
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 45

LOG AI CHEIO, voce tem 3 OPCOES

1 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando que o numero de paginas que ainda restam no LOG eh igual a: 5

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta trançacao serao DESFEITOS.

3 - TRABALHA SEM LOG AI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG AI ***
*** Avalie a possibilidade de fazer BACKUP ***

Entre com a opcao: 1, 2 ou 3 - 1

Grava nova pagina no AI 2 47
FIMBI- Testa espaco do BI 15 8
Grava reg. no BI 3 54
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 46

LOG AI CHEIO, voce tem 3 OPCOES:

1 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando que o numero de paginas que ainda restam no LOG eh igual a: 4

2 - ABORTA sua transacao (FALHA DE SISTEMA). No

proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG AI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG AI ***
*** Avalie a possibilidade de fazer BACKUP ***

Entre com a opcao: 1, 2 ou 3 - 1

Grava nova pagina no AI 2 54
FIMBI- Testa espaco do BI 15 9
Grava reg. no BI 3 52
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 47

LOG AI CHEIO, voce tem 3 OPCOES:

2 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando que o numero de paginas que ainda restam no LOG eh igual a: 3

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG AI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG AI ***
*** Avalie a possibilidade de fazer BACKUP ***

Entre com a opcao: 1, 2 ou 3 - 1

Grava nova pagina no AI 2 52
FIMBI- Testa espaco do BI 15 10
Grava reg. no BI 3 4
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 48

LOG AI CHEIO, voce tem 3 OPCOES

1 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando que o numero de paginas que ainda restam no LOG eh igual a: 2

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG AI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG AI ***
*** Avalie a possibilidade de fazer BACKUP ***

Entre com a opcao: 1, 2 ou 3 - 1

Grava nova pagina no AI 2 4
FINBI- Testa espaco do BI 15 11
Grava reg. no BI 3 14
Grava do buffer para a BASE
FIMAI - testa espaco do AI 50 49

LOG AI CHEIO, voce tem 3 OPCOES:

1 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando sue o numero de paginas sue ainda restam no LOG eh igual a: 1

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG Al e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG Al ***
*** Avalie a possibilidade de fazer BACKUP ***

Entre com a opcao: 1, 2 ou 3 - 1

Grava nova pagina no Al 2 14
FIMBI- Testa espaco do Bl 15 12
Grava reg. no Bl 3 56
Grava do buffer para a BASE
FIMAI - testa espaco do Al 50 50

LOG Al CHEIO, voce tem 3 OPCOES:

1 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando que o numero de paginas que ainda restam no LOG eh igual a: 0

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG Al e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG AI ***
*** Avalie a possibilidade de fazer BACKUP ***

Cntre com a opcao: 1, 2 ou 3 - 1

Opcao 1 invalida, nao existem mais paginas

LOG AI CHEIO, voce tem 3 OPCOES:

1 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando sue o numero de paginas que ainda restam no LOG eh igual a: 0

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG AI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG AI ***
*** Avalie a possibilidade de farer BACKUP ***

Entre com a opcao: 1, 2 ou 3 - 3

FIMBI- Testa espaco do BI 15 13

LOG BI CHEIO, voce tem 3 OPCOES:

1 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM ATUALIZAR. Lembrando que

o numero de paginas que ainda restam no LOG eh igual a: 2

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG BI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG BI ***

Entre com a opcao: 1, 2 ou 3 - 1

Grava reg. no BI 3 53
Grava do buffer para a BASE
FIMBI- Testa espaco do BI 15 14

LOG BI CHEIO, voce tem 3 OPCOES:

1 - Fica a seu criterio FECHAR a base de dados ATUALIZANDO ou SEM RTUALIZAR. Lembrando que o numero de paginas que ainda restam no LOG eh igual a: 1

2 - ABORTA sua transacao (FALHA DE SISTEMA). No proximo reinicio os efeitos desta transacao serao DESFEITOS.

3 - TRABALHA SEM LOG BI e SEM GARANTIAS

*** Seja qual for sua decisao, reavalie sua ***
*** aplicacao ou o tamanho do seu LOG BI ***

Entre com a opcao: 1, 2 ou 3 - 3

```
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
#?
 modificar te57 tal sue: codp = 5 (nomep para "teste-5");
Grava do bufSer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
#?
 modificar tes8 tal que: codp = 5 (nomep para "teste-5");
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
```

```
Grava do buffer para a BASE
Grava da buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
#?
 modificar tes9 tal que: codp = 5 (nomep para "teste-5");
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
#?
 modificar tes10 tal que: çodp = 5 (nomep para "teste-5");
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Gravado buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
```

```
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
#?
 most tes1;
 01 TESA
 02 TCS2
 03 TES3
 04 TES4
 05 TES5
#?
 most tes2;
 01 TES1
 02 TES2
 03 TES3
 e34 TES4
 05 TES5
H?
 most tes3;
 01 TES1
 02 TCS2
 03 TES3
 04 TES4
 05 TES5
#?
 most tes4;
 03. TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 most tes5;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
```

```
#?
 most tes6;
 01 TES1
 02 TES2
 83 TES3
 04 TCS4
 05 TESTE-5
#?
 most tes7;
 01 TES1
 02 TCS2
 03 TES3
 04 TES4
 05 TESTE-S
#?
 inost tes8;
 01 TES1
 02 TES2
 03 TES3
 04 TCS4
 05 TESTE-5
#?
 inost tes9;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TESTE-5
#?
 most tes10;
 01 TES1
 82 TES2
 03 TES3
 04 TCS4
 05 TESTE-5
#?
 ret regs de tes2 tal que codp = 4;
#?
 ret regs de tes3 tal sue codp = 4;
```

```
#?
 ret regs de tes4 tal que codp = 4j
#?
 ret regs de tes5 tal que codp = 4;
#?
 ret regs de tes6 tal que codp = 4;
#?
 ret regs de tes7 tal que codp = 4;
#?
 ret regs de tes8 Tal que codp = 4;
H?
 ret regs de tes9 tal que codp = 4;
#?
 ret regs de tes10 tal que codp = 4;
#?
 most tes1;
 01 TES1
 02 TES2
 03 TES3
 04 TES4
 05 TES5
#?
 most tes2;
 01 TES1
 02 TCS2
 03 TES3
 05 TCS5
#?
 most tes3;
 01 TES1
 02 TCS2
 03 TES3
 05 TES5
#?
 most tes4;
 01 TES1
 02 TCS2
 03 TES3
 05 TES5
```

```
#?
 most tes5;
 01 TES1
 02 TES2
 03 TES3
 05 TES5
#?
 most tes6;
 01 TESI
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes7;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes8;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes9;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tes10;
 01 TES1
 02 TES2
 03 TES3
 05 TESTE-5
#?
 most tab_autorizacoes;
ALUNO              TAB_AUTORIZACOES  5 03/06/90
```

```
ALUNO                TAB_ENTIDADES      4 03/06/90
ALUNO                TAB_ATRIBUTOS      4 03/06/90
ALUNO                TAB_DERIVACOES     4 03/06/90
ALUNO                TES1               3 03/06/90
ALUNO                TES2               3 03/06/90
ALUNO                TES3               3 03/06/90
ALUNO                TES4               3 03/06/90
ALUNO                TES5               3 03/06/90
ALUNO                TES6               3 03/06/90
ALUNO                TES7               3 03/06/90
ALUNO                TES8               3 03/06/90
ALUNO                TES9               3 03/06/90
ALUNO                TES10              3 03/06/90
#?
 fechar base de dados;
FECHBD, testa flag de rollback          0
FECHBD, testa ultima transacao          3
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
Grava do buffer para a BASE
FECHBD atualiza o primeiro registro na BASE
#?
 fechar sessao;
#?
 enc
```