UNIDADE 3 – RESTAURO DE BANCO DE DADOS
MÓDULO 1 – INTRODUÇÃO AO *BACKUP*/RESTORE

01

1 - EVENTOS QUE CAUSAM FALHAS DE SISTEMAS

Olá, seja bem-vindo a mais uma etapa do nosso estudo. Nesta unidade, trataremos de diversos assuntos relacionados a importantes rotinas de manutenção de dados, em especial, rotinas de *backup* e restauro.

Um *backup* é uma cópia fiel do ambiente de produção, que pode ser utilizado para a reconstrução de uma falha de sistema.

O ideal para um sistema de produção é que um *backup* nunca precise ser necessário, mas, sendo necessário, que ele funcione para realizar um restauro perfeito.

Nem todo arquivo precisa ser *backup*eado, arquivos temporários, auxiliares, de *swap* de memória, por exemplo, podem ficar fora das rotinas de *backup*. Às vezes até mesmo a instalação de programas ou o sistema operacional não precisam ser feitos *backup*. A organização deve avaliar o que é importante ser *backup*eado. Normalmente, os arquivos mais importantes são os arquivos de escritório (como planilhas e documentos), arquivos multimídia (como fotos, vídeos e músicas) e arquivos de banco de dados.

Em se tratando de tecnologia de informação, inúmeros são os eventos que podem causar falhas ou erros de sistemas:

- pessoas podem atuar conscientemente ou não de forma a gerar falhas de sistemas,
- *hardware*s podem falhar a qualquer hora,
- *software*s podem gerar erros e falhas capazes de derrubar sistemas.

Enfim, são inúmeros os eventos que causam falhas e erros em sistemas, a seguir, contextualizamos alguns provindos de relatórios de diversas organizações de segurança da informação.

Backupeado

Embora formalmente não exista o verbo “backupear” na língua portuguesa, a exemplo do que ocorre com outros termos da área de tecnologia da informação, seu uso já se consolidou, uma vez que não há vocábulo substitutivo em Português.

02

1.1 - Erros causados por usuários de sistemas

Geralmente, os sistemas são feitos para um grande número de usuários poderem utilizá-los. Sistemas que são pouco testados, durante a fase de desenvolvimento, são mais suscetíveis a erros causados por usuários, que podem tentar operar o sistema de uma forma diferente daquela planejada, tentar inserir informações incompatíveis com os campos da aplicação, operar o sistema em ambientes de *hardware*/*software* não planejado ou desatualizados.

Vários desses motivos podem gerar erros controláveis pela aplicação/ambiente de TI, derrubar o sistema por completo (deixando-o fora do ar), ou mesmo violar a integridade dos dados do sistema ou de um banco de dados, necessitando assim de uma recuperação.

Realizar testes extensos, validar e controlar a forma como as informações são inseridas, controlar perfis de acesso e às funcionalidades do sistema, controlar os mecanismos de acesso aos sistemas (exemplo, versões de navegadores WEB) são meios comuns de se evitar erros causados pelos usuários dos sistemas.

03

1.2 - Erros causados por equipe de TI

Muitas vezes, a manutenção de um sistema exige uma mudança feita diretamente no ambiente de produção. Essas mudanças deveriam ser sempre testadas em ambientes alternativos (de teste, homologação ou desenvolvimento), mas nem sempre isso é possível. Há certas condições que exigem que a mudança seja realizada diretamente em produção, sem a possibilidade de teste prévio.

Certamente, todas as equipes de TI são preocupadas com os eventuais impactos que uma mudança em um ambiente de produção pode gerar, entretanto, nem sempre é possível prever todos os problemas. Não é incomum vermos comandos mal planejados e, consequentemente, mal executados que precisam de “desfazimento” ou restauro. Nessas situações, uma cópia do ambiente pode ajudar.

1.3 - Ataques de *hacker*

Um sem número de organizações são afetadas diariamente por ataques de *hackers*. Essas pessoas visam desde alterar apenas páginas WEB das organizações, ou até mesmo roubar informações, apagar conteúdo ou alterar dados de sistemas.

Fique Atento!

As políticas de segurança das organizações devem prever mecanismos que impeçam ou dificultem esse tipo de ação, entretanto, caso ocorra um ataque *hacker* que apague os dados, será necessário usar informações de *backup* para reconstruir os ambientes.

04

1.4 - Sabotagem

Algumas empresas também são alvos de sabotagens internas. Infelizmente, por motivos diversos, existem vários relatos onde funcionários, usuários comuns, equipe de TI, equipes terceirizadas já sabotaram sistemas de informação nos mais variados tipos de organizações. Para tais ações, é imprescindível possuir *backup*s para recuperação dos sistemas.

1.5 - Pragas digitais

Vírus e outras pragas digitais afetam nossos ambientes de TI continuamente. Existem vários equipamentos e *software*s específicos que protegem os ambientes de TI, mas nada é 100% inviolável, principalmente para ambientes de TI que possuem conexão com a Internet. Vírus e pragas digitais podem ser capazes de corromper bases de dados e repositórios de arquivos, necessitando, novamente, de reconstrução do ambiente.

1.6 - Falhas de *software*

Hoje em dia é muito comum haver a atualização constante de aplicativos (desde sistemas operacionais, drivers, componentes, até aplicativos e componentes dos aplicativos). Qualquer erro de atualização pode gerar parada no sistema. São raros os eventos em que bases de dados foram corrompidas motivadas por falhas de *software*, mas isso não é impossível. Já estudamos, por exemplo, sistemas RAID não tolerantes a falhas, nos quais a falha do gerenciador de discos (*software*) pode gerar uma falha de arquivo.

05

1.7 - Falhas de *hardware*

Falhas de *hardware* são um dos eventos mais comuns que causam falhas de sistemas. Por se tratar de dispositivos mecânicos e eletrônicos, sobrecarga, curtos circuitos, desgaste, falhas eletromecânicas podem causar falhas de *hardware*. A recuperação desse tipo de falha muitas vezes requer a substituição dos equipamentos danificados bem como o restauro do *software*.

1.8 - Eventos da natureza

Embora raros, há vários registros de que enchentes, incêndios, terremotos e outros eventos da natureza destruíram ambientes de TI. Tais eventos são de alta gravidade, devido ao alto impacto financeiro causado. Muitas vezes, não só o ambiente de TI é impactado, mas todo o ambiente físico que comporta o ambiente de TI também é afetado.

A recuperação desse tipo de ocorrência geralmente é a que mais leva tempo, pois exige a remoção de tudo o que foi estragado, limpeza, aquisição de novos equipamentos, reformas estruturais, novas instalações, configuração, teste e só então é possível a recuperação dos sistemas.

06

2 - COMO ARMAZENAR *BACKUPS*

Há um número muito grande de possibilidades de mídias em que podemos armazenar *backup*s. Antigamente, nos anos 1960 e 70, só existiam fitas magnéticas, mas hoje em dia, as opções são diversas.

Muitos tipos de mídia já foram totalmente extintos, por conta da defasagem tecnológica, pouca capacidade de armazenamento e baixa performance de leitura e gravação. Padrões antigos de fitas magnéticas e disquetes são exemplos de mídias que não se usam mais. Entre as possibilidades atuais, há soluções simples e econômicas, e soluções complexas e de alto custo (que geralmente possuem alta qualidade, capacidade e performance).

Hoje em dia, muitas empresas utilizam mais de um tipo de estratégia de *backup*, por exemplo:

- para dados históricos e de pouca necessidade de acesso, uma empresa pode utilizar fitas magnéticas ou DVDs regraváveis,
- para dados aos quais o acesso deve ser rápido, uma empresa pode criar áreas de disco específicas para cópia de *backup* de dados.

A seguir vamos ver alguns tipos de mídias e suas características.

07

2.1 - Pendrive

Pendrives são dispositivos removíveis, baseados em conexão via porta USB, nos quais um chip de memória não volátil (não perde dados após desconexão elétrica) é utilizado para armazenamento de dados.

As primeiras unidades pendrive eram muito pequenas, com capacidade de armazenamento da ordem de Kilo bytes. Atualmente, além de terem obtido uma grande melhoria de performance de leitura e gravação, as capacidades de armazenamento chegam a valores próximos de um giga byte.

Essa capacidade de quase um giga byte deve ser suficiente para armazenar arquivos e bancos de dados de muitas organizações de pequeno e médio porte, mas, certamente é pouco para empresas de grande porte. Além disso, o tempo útil seguro para armazenamento em pendrive é considerado pequeno, normalmente próximo de um ano. Isso significa que não há garantia que você consiga acessar os dados de um pendrive onde os dados foram gravados há mais de um ano.

Dessa forma, apesar de oferecer uma forma barata, altamente portável e simples para cópia de arquivos, essa solução só é indicada para pequenas empresas e para armazenamento de pouco tempo de duração (conforme explicamos, no máximo um ano).

08

2.2 - Disco rígido removível

Discos rígidos removíveis são equipamentos que utilizam discos rígidos comuns instalados dentro de uma gaveta ou caixa que permite a conexão com os computadores.

Esta conexão normalmente é feita por meio de portas USB, mas há discos que suportam conexão nativa de *hardware*, como portas SATA, SCSI e outras. Os discos rígidos apresentam capacidade de armazenamento na ordem de 1 Tera byte (mil vezes maior que um pendrive), taxa de transferência igual ou superior, alta portabilidade e duração de armazenamento na ordem de 10 anos.

Os discos removíveis são uma boa opção para empresas de pequeno e médio porte, visto que pode ser possível que apenas um disco seja o suficiente para armazenar todos os dados de documentos e bases de dados da empresa. Uma outra vantagem de dispositivos que apresentam boa portabilidade é poder armazená-los longe do ambiente de TI, como em um cofre em outro local.

Para grandes corporações, um único disco removível pode ser muito pouco quando falamos em capacidade e necessidade de armazenamento, e dessa forma, não são adequados ao ambiente.

09

2.3 - Disco ótico (CD-ROM, DVD-ROM e discos Blu-Ray)

Os discos óticos, CD-ROM, DVD-ROM e discos Blu-Ray, desde a década de 90 vêm sendo usados por muitas empresas como solução econômica para *backup*s. Devido ao baixo custo das mídias, os discos óticos são largamente utilizados para armazenar dados históricos, ou seja, visões de momentos específicos de um ambiente de TI. Esse tipo de visão é muito interessante quando se precisa saber como determinada informação (seja arquivo ou banco de dados) estava em uma determinada data. Dessa forma, é possível restaurar o sistema para momentos específicos no tempo.

Há basicamente dois tipos de mídias óticas: aquelas que permitem apenas uma gravação e aquelas que permitem várias regravações. A primeira é considerada melhor para sistemas de *backup*, visto que a duração da mídia é superior à segunda. Acredita-se que mídias óticas possuam cerca de 10 a 30 anos de vida útil (garantia de leitura), porém, há dados estatísticos de mídias de baixa qualidade onde os dados não puderam ser lidos após um ou dois anos.

Devido ao fato do baixo custo, aliado à alta portabilidade, boa capacidade de armazenamento, boa performance de leitura e gravação, fácil utilização e alta compatibilidade com diversos tipos de computador, os discos óticos representam uma ótima opção de *backup* para a grande maioria das organizações. As capacidades máximas de armazenamento são: CD-ROM – 700MB, DVD-ROM – 9GB, Blu-Ray – 50 GB.

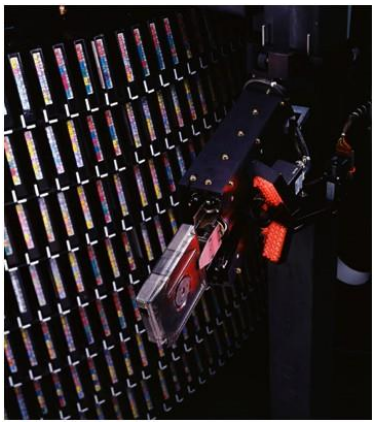
10

2.4 - Fitas magnéticas

As fitas magnéticas lembram aquelas antigas fitas de VHS ou mesmo fitas K7, foram criadas na década de 60 e representam uma solução especialmente dirigida às grandes corporações. Possuem baixa velocidade de leitura e gravação, mas apresentam uma grande vantagem: uma longa vida para o armazenamento.

A vida útil de fitas magnéticas é na ordem de 30 a 50 anos, o que faz com que elas sejam a melhor opção quando se pensa em manter dados por longo período. Entretanto, para essa vida útil são necessárias condições ideais de armazenamento (temperatura, umidade e outros), do contrário, as fitas podem se deteriorar rapidamente.

Hoje em dia há soluções automatizadas de gravação em fitas, denominadas “robôs de gravação”, em que mecanismos eletrônicos são capazes que pegar uma fita de um local de armazenamento, colocá-la no driver de leitura e gravação e depois movê-la novamente para o local de guarda. Um robô de gravação pode guardar as fitas em uma gaveta específica ou em uma “Tape Library” (biblioteca de fitas), que seria um equipamento capaz de guardar diversas (de dezenas a centenas) fitas.



Exemplo de uma tape library (esquerda) e um robô de *backup* (direita)

A **desvantagem** principal das fitas magnéticas é o custo operacional. Os equipamentos geralmente são onerosos e as próprias fitas são muito mais caras quando comparadas a discos óticos.

11

As tecnologias mais comuns para esse tipo de dispositivo são as fitas DLT, LTO e DDS/ (biDAT. Veja uma breve descrição de cada tecnologia:

- DLT (Digital Linear Tape);
- LTO (Linear Tape Open);
- DDS (Digital Data Storage) e DAT (Digital Audio Tape).

A IBM lançou em 2015 uma tecnologia de fita magnética com capacidade de armazenamento de 35 Tera Bytes. Isso representa 700 discos de Blu-ray.

Entretanto, as fitas magnéticas também possuem algumas **desvantagens** (comum a todos os tipos):

- **Fragilidade das mídias:** fitas são frágeis a quedas, campos magnéticos e variações climáticas (temperatura e humidade).
- **Ineficientes em casos de recuperação de grandes volumes de dados:** recuperação de grandes volumes de dados podem requerer dezenas de dias de leituras de fitas.
- **Dados abertos:** muitas fitas não possuem mecanismos de criptografia, o que deixa os dados abertos a qualquer um.

DLT (Digital Linear Tape)

Criado em 1984, cujo padrão mais moderno admite capacidade de armazenamento de até 800 GB com taxa de transferência de 36MB/s. As mais comuns são fitas de 40 a 160 GB. Possui garantia de armazenamento de 30 anos. A maior fabricante de fitas DLT é a Sony, mas também é fabricada pela Fujifilm, Maxell e Imation.

Fita DLT

LTO (Linear Tape Open)

Criada em 2000 pelas empresas Seagate, HP e IBM, possui capacidade de armazenamento de 1.5 TB, ou seja, o dobro das fitas DLT. Velocidade de transferência de 140 MB/s, cerca de quatro vezes mais rápida que as fitas DLT. Apresentam ainda a vantagem de criptografia nativa de dados.

Fita LTO

DDS (Digital Data Storage) e DAT (Digital Audio Tape)

Criada pela Sony e HP em 1989, possui atualmente fitas com 160 GB de armazenamento e 6 MB/s de taxa de transferência de dados. Tem a vantagem do baixo custo quando comparadas com fitas LTO. Essa tecnologia foi muito utilizada por estúdios de gravação de música, utilizando equipamentos digitais de gravação, por isso o nome Digital Audio Tape. Atualmente essa tecnologia encontra-se em desuso.

Fita DAT

12

2.5 - Discos rígidos

Discos rígidos e quaisquer outras soluções que envolvam discos rígidos (SAS, NAS, SAN etc.) são equipamentos interessantes para a realização de *backup*s.

As principais **vantagens** dos discos rígidos são:

- altíssima capacidade de armazenamento,
- altíssima performance (se comparados aos outros dispositivos de *backup*),
- baixo custo por byte armazenado,
- facilidade de uso (não precisa “montar” e “desmontar” unidades, por exemplo, como em fitas) e
- garantia de muitos anos de persistência dos dados magnéticos (de 10 a 30 anos em média).

As principais **desvantagens** dos discos rígidos são:

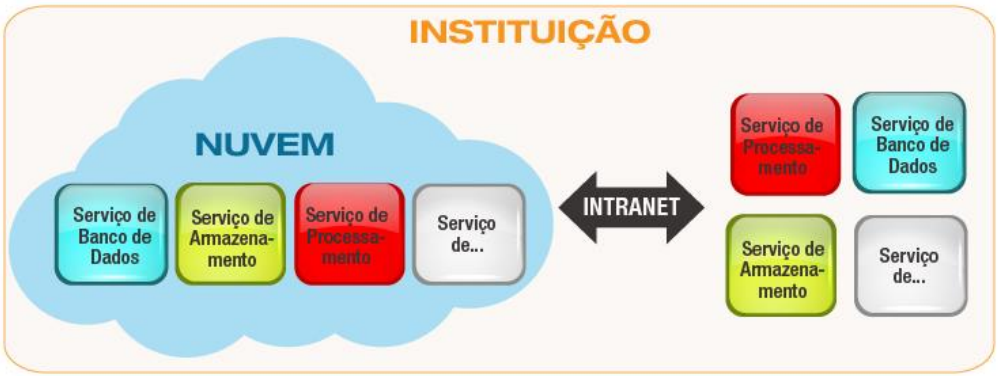
- quando fixos (padrão), os discos não possuem a mobilidade dos outros dispositivos, o que impede a guarda em locais protegidos.
- vulnerabilidade de acesso por usuários não autorizados e
- vulnerabilidade contra incidentes elétricos (como um curto-circuito) e outros.

O uso principal de discos rígidos como *backup* está relacionado aos arquivos de guarda temporária, onde precisamos reavê-los com frequência e alta disponibilidade. Para guarda permanente, discos rígidos não são uma opção adequada.

13

2.6 - Armazenamento na nuvem

O armazenamento na nuvem é uma solução relativamente nova e bastante interessante. Nela, empresas oferecem datacenters (muitas vezes redundantes) onde é possível armazenar dados digitais, utilizando a internet como meio para envio e recebimento de dados.



Uma das grandes **vantagens** desse tipo de solução é que a empresa não precisa adquirir nenhum equipamento, tampouco contratar pessoas para mantê-los funcionando; uma empresa especializada em armazenamento de dados faz todo o serviço e o interessado paga uma espécie de “aluguel de espaço digital”.

14

3 - Tipos de rotinas de *backup*

Basicamente, há dois tipos de rotinas de *backup*:

- *backup* de arquivos e
- *backup* de imagens.

O ***backup* de arquivos** refere-se à cópia dos arquivos de uma pasta para outra. Normalmente, esse tipo de *backup* é adequado apenas para arquivos de escritório (como planilhas e documentos), arquivos multimídia (como fotos, vídeos e músicas) e arquivos de banco de dados. Saiba+

Os ***backups* de imagem** referem-se às cópias fieis de discos rígidos, que inclui todos os arquivos dos discos, incluindo sistema operacional, programas instalados, arquivos temporários e outros. O *backup* de imagem é indicado para o restauro completo de um computador ou para a migração de *hardware*, por exemplo, a migração de todo um conteúdo de um HD para um disco de maior capacidade. Saiba+

O ***backup* de um banco de dados** pode ser feito de três formas:

- por meio de um *backup* de arquivos (copiando os arquivos do banco de dados e de log para outras pastas, discos ou unidades),
- por meio de um *backup* de imagem ou
- por comandos SQL (como o comando *BACKUP*), que geram novos arquivos contendo os dados de banco de dados em formato de *backup*.

Estudaremos no próximo módulo os comandos SQL de *backup* e restauro.

Saiba+

A cópia de arquivos pode ser feita por meio de comandos do sistema operacional (como o copiar os arquivos da pasta de origem e colá-los na pasta de destino) ou por meio de programas específicos que gerenciam melhor a cópia.

Saiba+

A cópia de imagem geralmente é feita por programas específicos, são poucos os sistemas operacionais que possuem nativamente um meio de realizar *backup*s de imagem.

15

4 - O restauro

O termo restauro está relacionado a utilizar as informações de *backup* para colocar o sistema computacional em algum estado operacional.

A realização de um restauro pode envolver o restauro completo do ambiente ou apenas de parte dele. Para a realização de um restauro é necessário não só o conjunto de informações do *backup*, mas sim de um processo completo para o restauro, que inclui, mas não se limita a:

- Procedimentos operacionais para o restauro;
- Suspensão temporária de acesso a sistemas pelos usuários enquanto o restauro é efetuado;
- Janelas de tempo onde os restauros podem ser executados;
- Teste do sistema/ambiente restauro;
- Comunicação e autorização necessárias antes e após o restauro.

Antes de um restauro, a equipe de TI precisa ter certeza de que os arquivos do *backup* são possíveis de serem restaurados. Dessa forma, é importantíssimo que os dados dos *backup*s sejam testados com certa frequência a fim de validar o processo real de restauro. Para realizar os testes, ambientes específicos de teste podem ser montados para verificação. Essa verificação pode ser completa ou parcial, de acordo com a criticidade e necessidade da organização.

Veja como são feitos os restauros:

- restauro de um *backup* de arquivo;
- restauro de um banco de dados;
- restauro de imagens.

restauro de um backup de arquivo
O restauro de um *backup* de arquivo geralmente é feito por comandos simples do sistema operacional (como o copiar e colar).

restauro de um banco de dados
O restauro de um banco de dados normalmente é feito por meio de comandos SQL (como o comando RESTORE), por meio da cópia de arquivos de dados ou por meio de um restauro de imagem.

restauro de imagens
O restauro de imagens é feito por meio de programas específicos e exigem que o computador seja configurado momentaneamente apenas para esse fim (sem acesso de outros sistemas e até mesmo do sistema operacional da máquina). Por isso, os *software*s de *backup* de imagem geralmente possuem um disco de boot específico.

5 - Onde armazenar os *backup*s

Segundo os conceitos de arquivologia, três são os tipos de classificação de informações que possuímos:

- Arquivos correntes;
- Arquivos de guarda temporária;
- Arquivos de guarda permanente.

Podemos perceber que só há duas opções de locais onde os dados podem ser armazenados: ou **dentro** dos computadores (no caso de discos rígidos) ou **fora** deles (nos demais casos). O importante aqui é determinar onde guardar as mídias e fitas de *backup*. Há no mercado, por exemplo, cofres especiais para a guarda de mídias e fitas de *backup*. Esses cofres são à prova de fogo e água, e, portanto, resistentes a qualquer tipo de catástrofe da natureza.

Algumas organizações consideram ainda a guarda dessas mídias em locais separados da organização, como um banco ou uma outra sede da empresa, afim de afastar riscos de uma catástrofe afetar os servidores e os *backup*s (perdendo todos os dados definitivamente).

Arquivos correntes

Todas aquelas informações que precisamos de acesso imediato. Essas informações devem estar disponíveis em tempo real, dessa forma, devem ser armazenadas em bancos de dados e pastas de arquivos (ambos localizados em discos rígidos e/ou soluções similares).

Arquivos de guarda temporária

São arquivos que não são necessários no dia a dia, mas devem estar próximos dos usuários para uma necessidade eventual. Por exemplo, arquivos de vendas não realizadas no ano pode ser interessante para uma empresa avaliar o porquê de as vendas não terem sido concluídas, dessa forma, ao invés de descartar os dados ou armazená-los em um lugar de difícil acesso, é interessante deixá-los próximos aos usuários. Essas informações podem residir em pastas compartilhadas (disco rígido) ou em outros dispositivos de fácil leitura, como mídias óticas.

Arquivos de guarda permanente

São arquivos que precisam ser armazenados por longo período, até que possam ser descartados. Muitas vezes, organizações produzem informações que precisam ser guardadas por 5, 10, 20, 30 anos ou mais. Essas informações geralmente têm valor histórico ou precisam ser guardadas por questões legais apenas, nenhum usuário tem necessidade dessas informações, porém, auditorias ou fiscalização legal podem exigir que tal informação seja identificada. Para esse tipo de questão, dispositivos de longa guarda, como fitas magnéticas ou discos óticos são boas escolhas. Quando possível, discos rígidos de alta qualidade também podem ser utilizados.

17

A política de segurança de uma empresa, quando tratando do assunto de *backup*s, deve incluir os seguintes tópicos:

- Quais são os dispositivos de *backup* a serem utilizados;
- Quais dados precisam e devem ser feitos *backups*;
- Qual a frequência e horários que os *backup*s devem ser gerados;
- Onde guardar os dados de *backup*;
- Por quanto tempo guardar esses dados;
- De quem é a responsabilidade por executar os *backup*s;
- De quem é a responsabilidade de levar as mídias de *backup* das unidades de *backup* até o local de guarda;
- Quem tem acesso físico ao local de guarda dos *backup*s;
- Como destruir mídias que não se tem mais interesse em guardá-las.
- Quem deve ser alertado caso ocorra algum imprevisto ou problema com relação às rotinas de *backup*.
- Quem é responsável por realizar restauros de informações?
- Que sistemas são impactados em eventuais restauros de informações.
- Quem deve ser comunicado no eventual restauro de informações.
- Como são os procedimentos para autorização e realização dos restauros.

- Como testar as mídias de *backup* para saber se estão íntegras.
- Como testar as rotinas de restauro para saber se realmente é possível realizar um restauro.

18

RESUMO

Neste módulo, aprendemos que:

a) Os *backup*s são cópias fieis do ambiente computacional de uma organização.

b) Nem todo dado precisa ser feito *backup*, apenas aquilo que a organização julga ser importante.

c) Há vários eventos que podem causar falhas em sistemas, alguns desses eventos podem requerer o restauro de bases de dados para que os sistemas voltem a operar normalmente. Entre esses tipos de eventos temos: erros causados por usuários de sistemas, erros causados por equipes de TI, ataques de hacker, sabotagem, pragas digitais, falhas de *software*, falhas de *hardware* e eventos da natureza.

d) As mídias mais utilizadas para *backup* são fitas magnéticas (em grandes corporações) e mídias óticas (em pequenas e médias organizações).

e) Mídias óticas são econômicas e de fácil utilização. Fitas magnéticas exigem conhecimentos adicionais para operação e custam bem mais, entretanto, são ideias para grandes volumes de dados.

f) *Backup*s de arquivos geralmente são feitos por meio de cópia de arquivos. *Backup*s de imagem geralmente são feitos por *software*s próprios.

g) Os SGBDs oferecem mecanismos de *backup* e restauro nativo nos quais são criados arquivos para armazenar dados e log.

h) Tanto o *backup* quando o restauro de um sistema deve seguir uma política interna da organização, para controle, segurança e informação aos interessados.

i) É importantíssimo testar os arquivos de *backup* de tempos em tempos a fim de garantir que eles são funcionais e recuperáveis.

j) As informações de uma organização são classificadas em: corrente (aquilo que está on-line e é acessado pelos usuários); guarda temporária (podem ser solicitados em algum momento e, portanto, devem ser facilmente recuperáveis) e guarda permanente (somente para fins históricos ou legais, podem ficar armazenados em mídias off-line).

UNIDADE 3 – RESTAURO DE BANCO DE DADOS
MÓDULO 2 – TÉCNICAS DE RECUPERAÇÃO DE BANCO DE DADOS (PARTE 1)
01

1 - RECUPERAÇÃO E CATEGORIZAÇÃO DOS ALGORITMOS DE RECUPERAÇÃO

Olá, seja bem-vindo a mais um módulo do nosso curso. Já tratamos anteriormente de conceitos gerais sobre informação, dispositivos para *backup* e políticas de *backup* e restauro. Agora, trataremos dos conceitos e técnicas relacionados à recuperação de banco de dados.

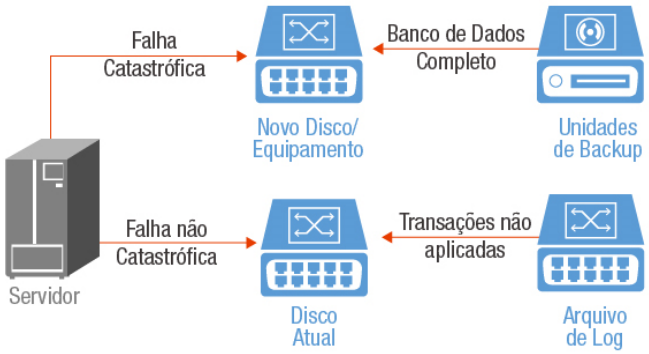
A recuperação de falhas de transação em geral significa que o banco de dados é restaurado ao estado consistente mais recente antes do momento da falha.

Para fazer isso, o sistema precisa manter informações sobre as mudanças que foram aplicadas aos itens de dados pelas diversas transações. Essa informação costuma ser mantida no log do sistema, conforme discutimos quando falamos sobre o arquivo de transações do banco de dados.

Uma estratégia típica para recuperação pode ser resumida informalmente da seguinte maneira:

1. Se houver dano extensivo a uma grande parte do banco de dados devido à **falha catastrófica**, como uma falha de disco, o método de recuperação **restaura uma cópia antiga do banco de dados que teve *backup* para o arquivamento** (normalmente, fita ou outro meio de armazenamento off-line de grande capacidade) e reconstrói um estado mais recente, reaplicando ou refazendo as operações das transações confirmadas do log em *backup*, até o momento da falha.

2. Quando o banco de dados no disco não está danificado fisicamente e uma **falha não catastrófica** tiver ocorrido (como, por exemplo, bloqueio mútuo), a estratégia de recuperação é **identificar quaisquer mudanças que possam causar uma inconsistência no banco de dados**. Veja um exemplo. Também pode ser preciso refazer algumas operações a fim de restaurar um estado consistente do banco de dados. Exemplo. Para a falha não catastrófica, o protocolo de recuperação não precisa de uma cópia de arquivamento completa do banco de dados. Em vez disso, as entradas mantidas no log do sistema on-line no disco são analisadas para determinar as ações apropriadas para recuperação.



Recuperação de falha catastrófica x falha não catastrófica.

Veja um exemplo

Por exemplo, uma transação que atualizou alguns itens do banco de dados no disco, mas não confirmou as necessidades de ter suas mudanças revertidas ao desfazer suas operações de gravação (nesse caso, os efeitos da transação devem ser desfeitos).

Exemplo

Por exemplo, se uma transação tiver sido confirmada, mas algumas de suas operações de gravação ainda não tiverem sido gravadas em disco.

02

Conceitualmente, podemos distinguir duas técnicas principais para recuperação de falhas de transação não catastróficas:

- **atualização adiada** e
- **atualização imediata**.

As técnicas de **atualização adiada** não atualizam fisicamente o banco de dados no disco até que uma transação atinge seu ponto de confirmação; então as atualizações são registradas no banco de dados.

Antes de atingir a confirmação, todas as atualizações de transação são registradas no espaço de trabalho de transação local ou nos buffers da memória principal que o SGBD mantém (o cache de memória principal do SGBD). Antes da confirmação, as atualizações são gravadas persistentemente no log e, após a confirmação, elas são gravadas no banco de dados no disco.

Se uma transação falhar antes de atingir seu ponto de confirmação, ela não terá alterado o banco de dados de forma alguma, de modo que o **UNDO** (desfazer) não é necessário. Pode ser preciso um **REDO** (refazer) para desfazer o efeito das operações de uma transação confirmada com base no log, pois seu efeito pode ainda não ter sido registrado no banco de dados em disco. Assim, a atualização adiada também é conhecida como algoritmo **NO-UNDO/REDO**. Discutiremos sobre isso mais à frente.

Modelo atualização adiada

03

Nas técnicas de **atualização imediata**, o banco de dados pode ser atualizado por algumas operações de uma transação antes que a transação alcance seu ponto de confirmação. Porém, essas operações também precisam ser registradas no log no disco ao forçar a gravação antes que elas sejam aplicadas ao banco de dados no disco, tornando a recuperação ainda possível.

Se uma transação falhar depois de gravar algumas mudanças no disco, mas antes de atingir seu ponto de confirmação, o efeito de suas operações no banco de dados precisa ser desfeito; ou seja, a transação deve ser revertida.

No caso geral da atualização imediata, tanto undo quanto redo podem ser exigidos durante a recuperação. Essa técnica, conhecida como algoritmo **UNDO/REDO**, requer as duas operações durante a recuperação. Uma variação do algoritmo, em que todas as atualizações precisam ser registradas no banco de dados em disco antes que a transação confirme, requer apenas undo, de modo que é conhecida como algoritmo **UNDO/NO-REDO**. Discutiremos essas técnicas futuramente.

Modelo de atualização imediata

As operações UNDO e REDO precisam ser **idempotentes**, ou seja, a execução de uma operação várias vezes é equivalente a executá-la apenas uma vez.

De fato, o processo de recuperação inteiro deve ser idempotente, pois se o sistema falhasse durante o processo de recuperação, a próxima tentativa de recuperação poderia realizar um UNDO e um REDO de certas operações write_item que já tinham sido executadas durante o primeiro processo de recuperação. O resultado da recuperação de uma falha do sistema durante a recuperação deve ser igual ao resultado da recuperação quando não há falha durante esse processo!

04

2 - CACHING (BUFFERING) DE BLOCOS DE DISCO

O processo de recuperação em geral está bastante interligado às funções do sistema operacional; em particular, o *buffering* de páginas de disco do banco de dados no cache de memória principal do SGBD. Normalmente, várias páginas de disco que incluem os itens de dados a serem atualizados são mantidas em cache nos buffers da memória principal e, depois, atualizados na memória antes de serem gravados de volta no disco.

O *caching* de páginas de disco é tradicionalmente uma função do sistema operacional, mas devido a sua importância para a eficiência dos procedimentos de recuperação, ele é tratado pelo SGBD chamando rotinas de baixo nível do sistema operacional. Com isso o SGBD é capaz de dizer o momento exato e os dados (blocos) ao gerenciamento de discos que devem ser lidos ou gravados.

Em geral, é conveniente considerar a recuperação em relação às páginas de disco (blocos) do banco de dados. Normalmente, uma coleção de buffers na memória, chamada **cache do SGBD**, é mantida sob o controle do SGBD com a finalidade de manter esses buffers. Uma tabela dentro do cache é usada para acompanhar quais itens de banco de dados estão nos buffers. Isso pode ser uma tabela de entradas <Endereço_página_disco, localização_buffer, ...>.

Quando o SGBD solicita ação em algum item, primeiro ele verifica a tabela no cache para determinar se a página de disco que contém o item está no cache do SGBD. Se não estiver, o item precisa ser localizado no disco, e as páginas de disco apropriadas são copiadas para o cache. Pode ser necessário **substituir** (ou **esvaziar**) alguns dos buffers de cache para criar espaço disponível para o novo item.

Clique aqui

Para saber mais sobre mecanismos de substituição de página, pesquise pelas técnicas de **Usada Menos Recentemente** (**MRU**) ou **First-In-First-Out** (**FIFO**). Atualmente há estratégias mais específicas para SGBDs, como **DBMIN** ou **Least-Likely-to-Use**.

05

As entradas na tabela de cache do SGBD mantêm informações adicionais relevantes ao gerenciamento de buffer. Associado a cada item de buffer na cache está um **bit sujo**, que é utilizado para indicar se o buffer foi modificado ou não.

Quando um bloco é lido inicialmente do disco do banco de dados para o buffer no cache, uma nova entrada é inserida na tabela de cache com o novo endereço do bloco do disco, e o bit sujo é definido como (**zero**). Assim que o buffer é modificado, o bit sujo para a entrada na tabela correspondente é definido como 1 (um**).**

Informações adicionais, como a(s) id(s) das transações que modificaram o buffer, também podem ser mantidas no cache. Quando o conteúdo do buffer é substituído (esvaziado) do cache, o conteúdo primeiro precisa ser gravado de volta à página de disco correspondente somente se seu bit sujo for 1.

Outro bit, chamado bit de **preso-solto**, também é necessário — uma página no cache está presa (valor de bit 1 (um)) se ainda não puder ser gravada de volta ao disco.

Por exemplo, o protocolo de recuperação pode impedir que certas páginas de buffer sejam gravadas no disco até que as transações que mudaram esse buffer tenham sido confirmadas. Após liberar a possibilidade de gravação, esse bit volta para o valor 0 (zero), dessa forma, o gerenciador do disco pode operar e gravar as informações de log e dados.

06

Duas estratégias principais podem ser empregadas quando se esvazia um buffer modificado para o disco:

A primeira estratégia, conhecida como **atualização no local**, grava o buffer no mesmo local de disco original, modificando assim o valor antigo de quaisquer itens de dados alterados no disco. Logo, uma única cópia de cada bloco de disco do banco de dados é mantida.

A segunda estratégia, conhecida como **sombreamento**, grava um buffer atualizado em um local diferente no disco, de modo que múltiplas versões dos itens de dados podem ser mantidas, mas essa técnica normalmente não é utilizada na prática por gerar dados desnecessários em disco (desperdiçando espaço).

Em geral, o valor antigo do item de dados antes da atualização é chamado de **Imagem Antes** (BFIM — *before image*), e o novo valor após a atualização é chamado de **Imagem Depois** (AFIM — *after image*).

Se o sombreamento for usado, tanto a BFIM quanto a AFIM podem ser mantidas no disco; assim, não é estritamente necessário manter um log para recuperação.

Mais à frente, discutiremos rapidamente a recuperação baseada no sombreamento.

07

3 - LOGGING WRITE-AHEAD, STEAL/NO-STEAL E FORCE/NO-FORCE

Quando a **atualização no local** é utilizada, é necessário usar um log para recuperação. Nesse caso, o mecanismo de recuperação precisa garantir que a BFIM do item de dados esteja registrada na entrada de log apropriada e que a entrada de log seja esvaziada para o disco antes de a BFIM ser modificada pela AFIM no banco de dados em disco. Esse processo geralmente é conhecido como ***logging write-ahead***, e é necessário poder desfazer (UNDO) a operação se isso for exigido durante a recuperação.

Antes de podermos descrever um protocolo para o *logging write-ahead*, precisamos distinguir entre dois tipos de informação de entrada de log incluída para um comando de gravação:

- a informação necessária para UNDO e
- a informação necessária para REDO.

Uma entrada de log tipo **REDO** inclui um **valor novo** (AFIM) do item gravado pela operação, pois isso é necessário para refazer seu efeito com base no log (ao definir o valor do item no banco de dados em disco para a sua AFIM).

As entradas de log tipo **UNDO** incluem o **valor antigo** (BFIM) do item, visto que isso é necessário para desfazer o efeito da operação baseada no log (ao definir o valor do item no banco de dados de volta para a sua BFIM). Em um algoritmo **UNDO/REDO**, os dois tipos de entradas de log são combinados. Além disso, quando o *rollback* em cascata é possível, entradas read_item no log são consideradas entradas tipo **UNDO**.

Protocolo *Logging write-ahead*.

08

Como dissemos, o cache do SGBD mantém os blocos de disco do banco de dados em cache nos buffers da memória principal, que incluem não apenas blocos de dados, mas também **blocos de índice** e **blocos de log do disco**.

Quando um registro de log é gravado, ele é armazenado no buffer de log atual no cache do SGBD. O log é simplesmente um arquivo de disco sequencial (apenas para acréscimo) e o cache do SGBD pode conter vários blocos de disco nos buffers da memória principal (em geral, os últimos blocos de log do arquivo de log).

Quando é feita uma atualização em um bloco de dados, armazenado no cache do SGBD, um registro de log associado é gravado no último buffer de log no cache do SGBD. Com a técnica de *logging write-ahead*, os buffers (blocos) de log que contêm os **registros de log** associados para determinada atualização do bloco de dados precisam primeiro ser gravados em disco, antes que o próprio **bloco de dados** possa ser gravado de volta no disco com base em seu buffer de memória principal.

09

A terminologia de recuperação de SGBD padrão inclui os termos ***steal/no-steal*** e ***force/no-force***, que especificam as regras que controlam quando uma página do banco de dados pode ser gravada do cache para o disco:

1. Se uma página do buffer em cache atualizada por uma transação não puder ser gravada em disco antes que a transação confirme, o método de recuperação é chamado de técnica ***no-steal***. O bit de preso-solto será usado para indicar se uma página não puder ser gravada de volta no disco. Contudo, se o protocolo de recuperação permitir gravar um buffer atualizado antes que a transação confirme, isso é chamado de ***steal***.

Steal é usado quando o gerenciador de cache (buffer) do SGBD precisa de um frame buffer para outra transação e o gerenciador de buffer substitui uma página existente que tinha sido atualizada, mas cuja transação não foi confirmada. A regra do ***no-steal*** significa que UNDO nunca será necessário durante a recuperação, pois uma transação confirmada não terá qualquer uma de suas atualizações no disco antes de ser confirmada.

2. Se todas as páginas atualizadas por uma transação forem imediatamente gravadas em disco antes que a transação confirme, essa é chamada de técnica ***force***. Caso contrário, ela é chamada ***no-force***. A regra do *force* significa que REDO nunca será necessário durante a recuperação, pois qualquer transação confirmada terá todas as suas atualizações em disco antes de ser confirmada.

10

O esquema de recuperação com atualização adiada (NO-UNDO) segue uma técnica ***no-steal***. Porém, os sistemas de banco de dados típicos empregam uma estratégia ***steal/no-force***. A vantagem do *steal* é que ele evita a necessidade de um espaço de buffer muito grande para armazenar todas as páginas atualizadas na memória. A vantagem do *no-force* é que uma página atualizada de uma transação confirmada ainda pode estar no buffer quando outra transação precisar atualizá-la, eliminando assim o custo de Entrada/Saída para gravar essa página várias vezes em disco, e possivelmente ter de lê-la novamente do disco. Isso pode oferecer uma economia substancial no número de operações de E/S de disco quando uma página específica é bastante atualizada por várias transações.

Para permitir a recuperação quando a **atualização no local** é usada, as entradas apropriadas exigidas precisam ser permanentemente gravadas no log em disco antes que as mudanças sejam aplicadas ao banco de dados. Por exemplo, considere o seguinte protocolo de *logging write-ahead* (WAL) para um algoritmo de recuperação que exige tanto UNDO quanto REDO:

1. A **imagem antes (BFIM)** de um item não pode ser modificada por sua **imagem depois (AFIM)** no banco de dados em disco até que todos os registros de log do tipo UNDO para a transação em atualização, até este ponto, tenham sido gravados à força no disco.

2. A operação de confirmação de uma transação não pode ser concluída até que todos os registros de log tipo REDO e tipo UNDO para essa transação tenham sido gravados à força no disco.

Para facilitar o processo de recuperação, o subsistema de recuperação do SGBD pode manter uma série de listas relacionadas às transações que estão sendo processadas no sistema. Estas incluem uma lista para transações ativas que começaram, mas ainda não foram confirmadas, e também podem incluir listas de todas as transações confirmadas e abortadas desde o último *checkpoint* (ver a próxima seção). Manter essas listas torna o processo de recuperação mais eficiente.

11

4 - CHECKPOINT NO LOG DO SISTEMA E CHECKPOINT FUZZY

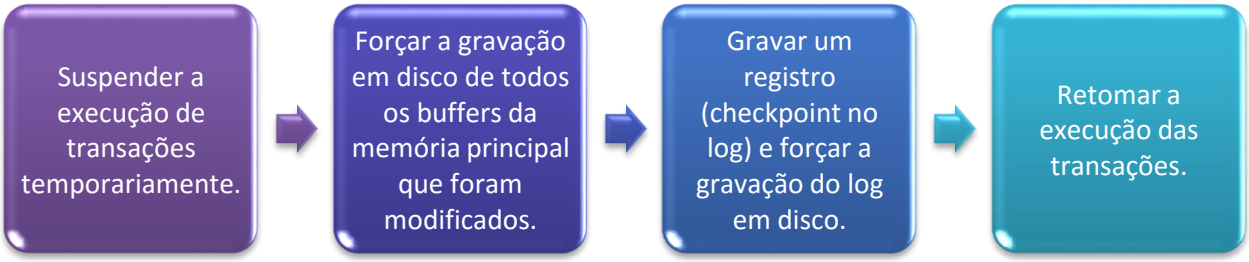
Outro tipo de entrada no log é chamado de ***checkpoint***. Um registro do tipo <*checkpoint*, lista de transações ativas> é gravado no log periodicamente no ponto em que o sistema grava, no banco de dados em disco, todos os buffers do SGBD que foram modificados.

Como consequência disso, todas as transações que têm suas entradas <*commit*, T> no log antes de uma entrada <checkpoint> não precisam ter suas operações WRITE refeitas no caso de uma falha do sistema, pois todas as suas atualizações serão registradas no banco de dados em disco durante o *checkpoint*.

Como parte do *checkpoint*, a lista de ids de transação para transações ativas no momento do *checkpoint* é incluída no registro do *checkpoint*, de modo que essas transações possam ser facilmente identificadas durante a recuperação.

O gerenciador de recuperação de um SGBD precisa decidir em que intervalos realizar um *checkpoint*. O intervalo pode ser medido em tempo, digamos, a cada **m** minutos, ou no número **t** de transações confirmadas desde o último *checkpoint*, em que os valores de **m** ou **t** são parâmetros do sistema.

Realizar um *checkpoint* consiste nas seguintes ações:



12

Como uma consequência da etapa 2, um registro de *checkpoint* no log também pode incluir informações adicionais, como uma lista de ids de transação ativas e os locais (endereços) do primeiro e mais recente (último) registros no log para cada transação ativa. Isso pode facilitar o desfazer de operações de transação caso uma transação tenha de ser desfeita.

O tempo necessário para forçar a gravação de todos os buffers de memória modificados pode atrasar o processamento da transação por causa da etapa 1. Para reduzir esse atraso, é comum usar uma técnica chamada ***checkpoint fuzzy***.

Com a técnica *checkpoint fuzzy*, o sistema pode retomar o processamento da transação após um registro (begin_checkpoint) ser gravado no log sem esperar que a etapa 2 termine.

Quando a etapa 2 é concluída, um registro (end_checkpoint, ...) é gravado no log com a informação relevante coletada durante o *checkpoint*. Porém, até que a etapa 2 termine, o registro do *checkpoint* anterior deve permanecer válido. Para isso, o sistema mantém um arquivo no disco que contém um ponteiro para o *checkpoint* válido, que continua a apontar para o registro do checkpoint anterior no log. Quando a etapa 2 termina, o ponteiro é mudado de modo a apontar para o novo *checkpoint* no log.

13

5 - *ROLLBACK* DE TRANSAÇÃO E *ROLLBACK* EM CASCATA

Se uma transação falhar por um motivo qualquer depois de atualizar o banco de dados, mas antes que a transação seja confirmada, pode ser preciso reverter (*roll back*) a transação. Se quaisquer valores de item de dados tiverem sido alterados pela transação e gravados no banco de dados, eles precisam ser restaurados para seus valores anteriores (BFIMs).

As entradas de log do tipo UNDO são usadas para restaurar os valores antigos dos itens de dados que precisam ser revertidos. Se uma transação “A” for revertida, qualquer transação “B” que tenha, enquanto isso, lido o valor de algum item de dados gravado por “A” também deve ser revertida.

De modo semelhante, quando “B” for revertida, qualquer transação “C” que tenha lido o valor de algum item de dados gravado por “B” também precisa ser revertida, e assim por diante.

Esse fenômeno é chamado de ***rollback* em cascata** (propagação de cancelamento), e pode ocorrer quando o protocolo de recuperação garante a propagação de efeitos em dados compartilhados por mais de uma transação.

14

Vamos a um exemplo para ficar mais claro:

1. A transação A estava alterando a nota da prova de matemática de um aluno, mudando de nota 8 para nota 8,5...
2. Antes de esta transação A terminar, outra transação B estava lendo as notas das provas de matemática de todos os alunos e gravando a nota média das provas no banco de dados...
3. Ainda, uma terceira transação estava lendo a nota média das provas de matemática que a transação B estava atualizando...

4. Neste momento, a transação A falha, e a atualização na nota é cancelada, revertendo o valor da nota daquele aluno para 8.
5. A transação B é ordenada então para que seja cancelada, e a média das notas não é salva.
6. Como consequência dos efeitos em cascata, a transação C que iria ler a média das notas também deve ser cancelada.
7. Dessa forma, observamos que um erro na transação A fez cancelar a transação B que lia um dado alterado por A, e a transação C foi cancelada porque lia um dado afetado pela transação B que foi cancelada também. Isso é o efeito em cascata.

É fácil entender que o *rollback* em cascata pode ser muito complexo e demorado. Por isso que quase todos os mecanismos de recuperação são projetados de modo que o *rollback* em cascata nunca seja necessário.

Na prática, o *rollback* em cascata das transações nunca é exigido nos SGBDs modernos, porque os métodos de recuperação atuais garantem sequenciamento das transações sem cascata. Logo, não é necessário gravar quaisquer operações de leitura de dados (read_item) no log, pois estas só seriam necessárias para determinar o *rollback* em cascata.

15

6 - Ações de transação que não afetam o banco de dados

Em geral, uma transação terá ações que não afetam o banco de dados, como a geração e impressão de mensagens ou relatórios das informações recuperadas do banco de dados. Se uma transação falhar antes de concluir, podemos não querer que o usuário receba esses relatórios, pois a transação deixou de completar.

Se esses relatórios errôneos forem produzidos, parte do processo de recuperação teria de informar ao usuário que esses relatórios estão errados, visto que o usuário pode tomar uma ação, com base nesses relatórios, que afeta o banco de dados. Logo, esses relatórios só devem ser gerados **depois que a transação atinge seu ponto de confirmação**.

Um método comum de tratar tais ações é emitir os comandos que geram os relatórios, mas mantê-las como tarefas em *batch*, que são executadas somente depois que a transação atinge seu ponto de confirmação. Se a transação falha, as tarefas em batch são canceladas.

Nosso estudo sobre esse assunto não terminou, continuaremos no próximo módulo.

16

RESUMO

Neste módulo, aprendemos que:

a) A recuperação de falhas de transação em geral significa que o banco de dados é restaurado ao estado consistente mais recente antes do momento da falha.

b) Para que as recuperações sejam possíveis, o sistema precisa manter informações sobre as mudanças que foram aplicadas aos itens de dados pelas diversas transações.

c) As informações de recuperação costumam ser mantidas no log do sistema.

d) Uma falha catastrófica normalmente é recuperada restaurando o *backup* mais recente e aplicando as transações confirmadas no arquivo de log de *backup*.

e) Uma falha não catastrófica normalmente é recuperada pelo cancelamento de transações que possam causar inconsistência no banco de dados.

f) Uma falha não catastrófica utiliza o log do sistema on-line para recuperação.

g) As duas principais técnicas para recuperação de falhas de transação não catastróficas são atualização adiada (quando o banco de dados só é fisicamente atualizado após a confirmação da gravação do registro transacional no arquivo de log) e atualização imediata, quando as operações físicas de gravação de log e de gravação dos dados ocorre em paralelo.

h) Ao falhar uma transação, dependendo da técnica empregada, serão necessários comandos de UNDO ou REDO para corrigir os problemas da falha. Para a técnica de atualização adiada, um REDO pode ser necessário. Para a técnica de atualização imediata, um UNDO podem ser necessários.

17

i) O SGBD interage com o cache de disco por meio de rotinas de baixo nível do sistema operacional.

j) O SGBD possui dois mecanismos de esvaziamento de buffer: atualização no local, onde o buffer grava no mesmo local de disco original, e sombreamento, no qual o buffer é gravado em outro local do disco, mantendo múltiplas versões dos itens de dados (esta segunda técnica não é mais utilizada).

k) A técnica de bit sujo controla quando o valor do item de dado no cache está diferente do item de dado no disco, necessitando, portanto, de atualização no disco.

l) A técnica de bit preso-solto controla quando o gerenciador de disco pode gravar o valor do item de dado no log e no disco.

m) O cache do SGBD mantém os blocos de disco do banco de dados em cache nos buffers da memória principal, que incluem não apenas blocos de dados, mas também blocos de índice e blocos de log do disco.

n) O método de recuperação é chamado *no-steal* quando uma página atualizada no buffer não puder ser gravada em disco.

o) O método de recuperação é chamado de *steal* quando uma página atualizada no buffer puder ser gravada em disco mesmo antes da confirmação da transação.

p) O método de recuperação é chamado de *force* quanto todas as páginas atualizadas no buffer forem imediatamente gravadas no disco, mesmo antes da confirmação da transação. Do contrário, é chamado de *no-force*.

q) Os sistemas de banco de dados típicos empregam uma estratégia *steal*/*no-force*.

r) O *checkpoint* refere-se a um registro no log de que todas as operações pendentes no buffer foram descarregadas em disco. A partir deste momento, o buffer pode ser totalmente esvaziado.

s) O *rollback* em cascata é a propagação de rotinas de desfazimento de transações interligadas por itens de dados utilizados por mais de uma delas.

UNIDADE 3 – RESTAURO DE BANCO DE DADOS

MÓDULO 3 – TÉCNICAS DE RECUPERAÇÃO DE BANCO DE DADOS (PARTE 2)

01

1 - MÉTODOS DE RECUPERAÇÃO

Olá, seja bem-vindo a mais uma etapa do nosso estudo. Recentemente estudamos os conceitos básicos de como o SGBD se recupera de uma falha não catastrófica. Agora, estudaremos os métodos de recuperação de banco de dados.

Os **métodos de recuperação** são:

- recuperação baseada em atualização adiada,
- recuperação baseada em atualização imediata,
- paginação de sombra e
- algoritmo ARIES.

Na sequência vamos detalhar cada um deles.

02

1.1 - Recuperação baseada em atualização adiada (NO-UNDO/REDO)

A ideia por trás da atualização adiada é adiar ou postergar quaisquer atualizações reais para o banco de dados em disco até que a transação termine sua execução com sucesso e atinja seu ponto de confirmação.

Durante a execução da transação, as atualizações são registradas apenas no log e nos buffers de cache. Depois que a transação atinge seu ponto de confirmação e o log é forçado a gravar em disco, as atualizações são registradas no banco de dados. Se uma transação falhar antes de atingir seu ponto de confirmação, não é preciso desfazer qualquer operação, pois a transação não afetou o banco de dados no disco de forma alguma. Portanto, somente entradas de log tipo **REDO** são necessárias no log, que incluem o **valor novo** (AFIM) do item gravado por uma operação de gravação.

As entradas de log tipo **UNDO** não são necessárias, pois não será preciso desfazer as operações durante a recuperação. Embora isso possa simplificar o processo de recuperação, não pode ser usado na prática, a menos que as transações sejam curtas e que cada transação mude poucos itens.

Para outros tipos de transações, existe o potencial de esgotar o espaço de buffer, pois as mudanças na transação devem ser mantidas nos buffers de cache até o ponto de confirmação.

03

Podemos declarar um **protocolo de atualização adiada** típico da seguinte forma:

1. Uma transação não pode mudar o banco de dados no disco até que atinja seu ponto de confirmação.
2. Uma transação não atinge seu ponto de confirmação até que todas as suas entradas de log tipo REDO sejam registradas no log e o buffer de log seja gravado à força no disco.

Observe que a etapa 2 desse protocolo é uma reafirmação do protocolo de *logging write-ahead*. Como o banco de dados nunca é atualizado em disco antes de a transação ser confirmada, nunca há necessidade de desfazer (UNDO) quaisquer operações. REDO é necessário caso o sistema falhe depois que a transação for confirmada, mas antes que todas as mudanças sejam gravadas no banco de dados em disco. Nesse caso, as operações da transação são refeitas das entradas de log durante a recuperação.

Para sistemas multiusuários com controle de concorrência, os processos de controle de concorrência e recuperação são inter-relacionados. Considere um sistema em que o controle de concorrência usa o bloqueio estrito em duas fases, de modo que os bloqueios nos itens permanecem em vigor até que a transação atinja seu ponto de confirmação. Depois disso, os bloqueios podem ser liberados. Isso garante schedules estritos e serializáveis (processamento sequencial de transações).

04

Supondo que entradas (*checkpoint*) sejam incluídas no log, um algoritmo de recuperação possível para esse caso funciona da seguinte maneira:

1- Use duas listas de transações mantidas pelo sistema: as transações confirmadas desde o último checkpoint (lista de confirmação) e as transações ativas (lista ativa).

2- Refaça (REDO) todas as operações write_item das transações confirmadas com base no log, na ordem em que foram gravadas nele.

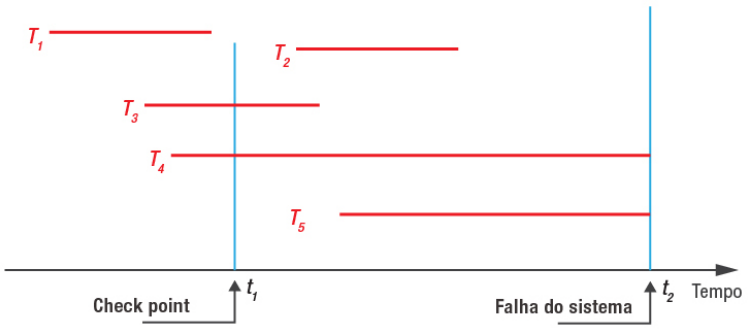
3- As transações que estão ativas e não confirmaram são efetivamente canceladas e devem ser submetidas novamente.

O procedimento REDO é definido da seguinte maneira:

Refazer uma operação write_item consiste em examinar sua entrada de log <write_item, T, X, valor_novo> e definir o valor do item X no banco de dados para valor_novo, que é a imagem depois (AFIM).

05

A Figura a seguir ilustra uma linha de tempo para um schedule possível de transações executáveis.



Exemplo de uma linha de tempo de recuperação para ilustrar o efeito do checkpoint.

Quando o checkpoint foi tomado no tempo t1, a transação T1 tinha sido confirmada, enquanto as transações T3 e T4 não o tinham. Antes da falha do sistema no tempo t2 , T1, T2 e T3 tinham sido confirmadas, mas T4 e T5 não.

De acordo com o método de recuperação, não é preciso refazer as operações write_item da transação T1, ou quaisquer transações confirmadas antes do momento do último checkpoint em t1. As operações write_item de T2 e T3 devem, contudo, ser refeitas, pois as duas transações atingiram seus pontos de confirmação após o último checkpoint. Lembre-se de que o log é gravado à força antes de confirmar uma transação. As transações T4 e T5 serão ignoradas: elas serão efetivamente canceladas ou revertidas porque nenhuma de suas operações write_item foram gravadas no banco de dados em disco sob o protocolo de atualização adiado.

06

A recuperação NO-UNDO/REDO é realizada refazendo (REDO) a última atualização da transação com base no log. Nesse caso, sempre que um item for refeito, ele é acrescentado a uma lista de itens refeitos. Antes que o REDO seja aplicado a um item, a lista é verificada; se o item aparecer na lista, ele não é refeito novamente, pois seu último valor já foi recuperado.

Se uma transação for abortada por algum motivo (digamos, pelo método de detecção de *deadlock*), ela é simplesmente submetida novamente, pois não alterou o banco de dados no disco.

Uma **desvantagem** do método descrito aqui é que ele limita a execução concorrente das transações, porque todos os itens bloqueados para a gravação permanecem bloqueados até que a transação atinja seu ponto de confirmação. Além disso, pode ser exigido um espaço de buffer excessivo para manter todos os itens atualizados até que as transações sejam confirmadas.

O principal **benefício** do método é que as **operações da transação nunca precisam ser desfeitas**, por dois motivos:

1. Uma transação não registra quaisquer mudanças no banco de dados em disco até que atinja seu ponto de confirmação — ou seja, até que complete sua execução com sucesso. Portanto, uma transação nunca é revertida por falha durante a execução da transação.

2. Uma transação nunca lerá o valor de um item que é gravado por uma transação não confirmada, visto que os itens permanecem bloqueados até que uma transação atinja seu ponto de confirmação. Assim, não haverá *rollback* em cascata.

A figura anterior mostra um exemplo de recuperação para um sistema multiusuário que utiliza o método de recuperação e controle de concorrência que descrevemos.

07

1.2 - Técnicas de recuperação baseadas em atualização imediata

Nessas técnicas, quando uma transação emite um comando de atualização, o banco de dados no disco pode ser atualizado imediatamente, sem qualquer necessidade de esperar que a transação atinja seu ponto de confirmação.

Observe que **não é obrigatório** que cada atualização seja aplicada imediatamente ao disco; é apenas **possível** que algumas atualizações sejam aplicadas ao disco antes que a transação seja confirmada.

Devem-se tomar providências para desfazer o efeito das operações de atualização que foram aplicadas ao banco de dados por uma transação com falha. Isso é obtido ao reverter a transação e desfazer o efeito das operações write_item da transação. Portanto, as entradas do log tipo **UNDO**, que incluem o **valor antigo (BFIM)** do item, devem ser armazenadas no log. Como o UNDO pode ser necessário durante a recuperação, esses métodos seguem uma **estratégia *steal*** para decidir quando os buffers da memória principal atualizados podem ser gravados de volta no disco.

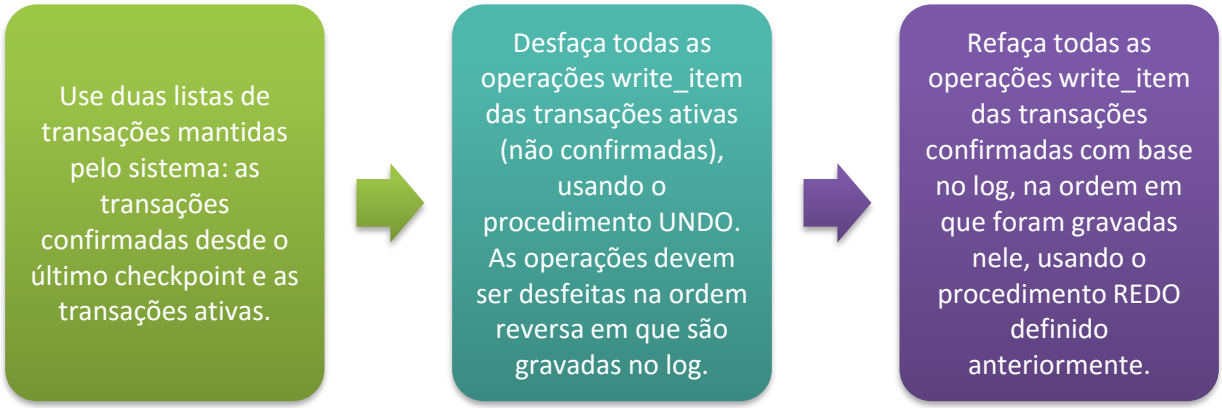
Teoricamente, podemos distinguir duas categorias principais de algoritmos de atualização imediata:

1. Se a técnica de recuperação garante que todas as atualizações de uma transação são gravadas no banco de dados em disco antes que a transação seja confirmada, não há motivo para refazer (REDO) quaisquer operações das transações confirmadas. Isso é chamado de algoritmo de **recuperação UNDO/NO-REDO**. Nesse método, **todas as atualizações** por uma transação devem ser gravadas em disco antes que a transação seja confirmada, de modo que o REDO nunca é necessário. Assim, esse método precisa utilizar a estratégia **force** para decidir quando os buffers atualizados da memória principal são gravados de volta no disco.

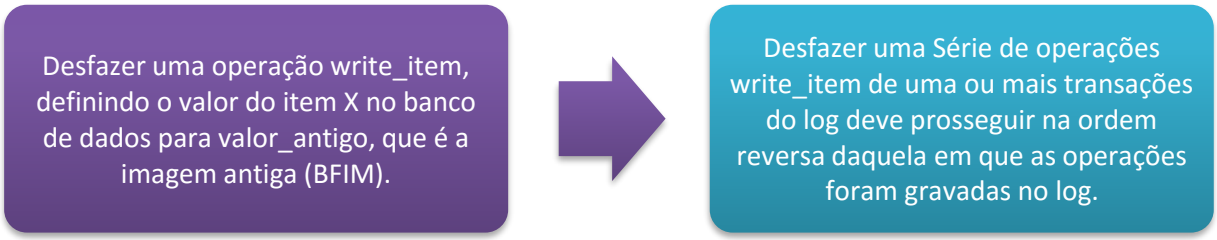
2. Se a transação puder confirmar antes que todas as mudanças sejam gravadas no banco de dados, temos o caso mais geral, conhecido como algoritmo de **recuperação UNDO/REDO**. Nesse caso, a estratégia ***steal*/no-force** é aplicada. Essa também é a técnica mais complexa.

08

O procedimento **UNDO/REDO** com *checkpoint* funciona da seguinte forma:



O procedimento **UNDO** é definido da seguinte forma:



Sempre que um item é refeito, ele é acrescentado à lista de itens refeitos e não é refeito novamente. Sempre que um item é desfeito, ele é acrescentado a uma lista de itens desfeitos e não é desfeito novamente.

09

1.3 - Paginação de sombra

Esse esquema de recuperação não exige o uso de um log em um ambiente monousuário. Em um ambiente multiusuário, um log pode ser necessário para o método de controle de concorrência.

A paginação de sombra considera o banco de dados composto de uma série de páginas de disco (ou blocos de disco) de tamanho fixo para fins de recuperação. Um diretório com várias entradas é construído, no qual cada entrada aponta para cada página de banco de dados no disco.

O diretório é mantido **na memória principal** se não for muito grande, e todas as referências (leituras e gravações) a páginas do banco de dados no disco passam por ela. Quando uma transação começa a ser executada, o diretório atual — cujas entradas apontam para as páginas de banco de dados mais recentes no disco — é copiado para um diretório de sombra. O diretório de sombra é, então, salvo em disco enquanto o diretório ativo é usado pela transação.

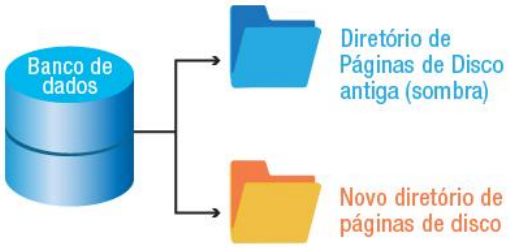
Diagrama de um diretório de páginas de disco e um diretório de sombra criado para suportar uma transação que posteriormente é gravada fisicamente no banco de dados.

A imagem acima mostra a situação antes de a transação ser confirmada.

10

Durante a execução da transação, o diretório de sombra nunca é modificado. Quando uma operação write_item é realizada, uma nova cópia da página de banco de dados modificada é criada, mas a cópia antiga dessa página não é modificada. Em vez disso, a nova página é gravada em outro lugar — em algum bloco de disco anteriormente não utilizado.

A entrada do diretório atual é modificada para que aponte para o novo bloco de disco, enquanto o diretório de sombra não é modificado e continua a apontar para o antigo bloco de disco não modificado. Para as páginas atualizadas pela transação, duas versões são mantidas. A versão antiga é referenciada pelo diretório de sombra e a nova versão, pelo diretório atual.



Após a confirmação da transação, o item de dado passa a apontar para uma nova página de disco. A página que contém os dados anteriores é mantida na área de sombra.

11

Para recuperar-se de uma falha durante a execução da transação, é suficiente liberar as páginas de banco de dados modificadas e descartar o diretório ativo. O estado do banco de dados antes da execução da transação está disponível por meio do diretório de sombra, e esse estado é recuperado ao restaurar o diretório de sombra. O banco de dados, assim, é retornado ao seu estado anterior à transação que estava executando quando ocorreu a falha, e quaisquer páginas modificadas são descartadas. A confirmação de uma transação corresponde a descartar o diretório de sombra anterior.

Como a recuperação não envolve desfazer nem refazer itens de dados, essa técnica pode ser categorizada como uma técnica NO-UNDO/NO-REDO para recuperação. Em um ambiente multiusuário com transações concorrentes, logs e checkpoint precisam ser incorporados à técnica de paginação de sombra.

Uma **desvantagem** da página de sombra é que as páginas de banco de dados atualizadas mudam de local no disco. Isso torna difícil manter páginas de banco de dados relacionadas próximas no disco sem o uso de complexas estratégias de gerenciamento de armazenamento. Além do mais, se o diretório for grande, o *overhead* de gravar diretórios de sombra em disco, à medida que as transações são confirmadas, é significativo. Outra complicação é o modo como se trata a coleta de lixo quando uma transação é confirmada. Saiba+

Saiba+

As páginas antigas referenciadas pelo diretório de sombra que foram atualizados devem ser liberadas e acrescentadas à lista de páginas livres para uso futuro. Essas páginas não são mais necessárias após a confirmação da transação. Outra questão é que a operação para migrar entre os diretórios atual e de sombra deve ser implementada como uma operação atômica.

12

1.4 - O algoritmo de recuperação ARIES

O algoritmo ARIES é um exemplo de algoritmo de recuperação usado em sistemas de banco de dados. Ele é utilizado em muitos produtos relacionados a banco de dados relacional da IBM.

O ARIES possui uma técnica ***steal*/no-force** para gravação, e é baseado em três conceitos:

- *logging write-ahead*,
- histórico repetitivo durante o redo e
- mudanças no *logging* durante o undo.

Já discutimos o ***logging write-ahead*** anteriormente.

O segundo conceito, **histórico repetitivo**, significa que o ARIES retraçará todas as ações do sistema de banco de dados antes da falha para reconstruir o estado do banco de dados quando a falha ocorrer.

As transações que não foram confirmadas no momento da falha (transações ativas) são desfeitas.

O terceiro conceito, **logging durante o undo**, impedirá que o ARIES repita as operações de undo completadas se houver uma falha durante a recuperação, causando um reinício do processo de recuperação.

13

O procedimento de recuperação ARIES consiste em três etapas principais: análise, REDO e UNDO:

Análise	REDO	UNDO
A etapa de análise identifica as páginas sujas (atualizadas) no buffer e o conjunto de transações ativas no momento da falha. **O ponto apropriado no log em que a operação REDO deveria começar também é determinado.**	Esta fase reaplica as atualizações do log ao banco de dados. Em geral, a operação REDO é aplicada apenas a transações confirmadas. Porém, isso não acontece no ARIES. Certas informações no log do ARIES oferecerão o ponto de partida para o REDO, com base no qual as operações de REDO são aplicadas até o final do log ser alcançado. Além disso, as informações armazenadas pelo ARIES e nas páginas de dados permitirão que o ARIES determine se a operação a ser refeita realmente foi aplicada ao banco de dados e, portanto, não precisa ser reaplicada. Assim, somente as operações de REDO necessárias são aplicadas durante a recuperação.	Durante a fase de **UNDO**, o log é varrido de trás para frente e as operações das transações que estavam ativas no momento da falha são desfeitas na ordem contrária. As informações necessárias para o ARIES realizar seu procedimento de recuperação incluem o log, a tabela de transações e a tabela de páginas sujas. Além disso, o checkpoint é utilizado. Essas tabelas são mantidas pelo gerenciador de transação e gravadas no log durante o checkpoint.

14

2 - RECUPERAÇÃO EM SISTEMAS DE MÚLTIPLOS BANCOS DE DADOS

Até aqui, assumimos implicitamente que uma transação acessa um único banco de dados. Em alguns casos, uma única transação, chamada **transação multibanco de dados**, pode exigir acesso a vários bancos de dados.

Esses bancos de dados podem ainda ser armazenados em diferentes tipos de SGBDs; por exemplo, alguns SGBDs podem ser relacionais, enquanto outros são orientados a objeto, hierárquicos ou de rede. Nesse caso, cada SGBD envolvido na transação multibanco de dados pode ter a própria técnica de recuperação e gerenciador de transação separados daqueles dos outros SGBDs. Essa situação é um tanto quanto semelhante ao caso de um sistema de gerenciamento de banco de dados distribuído, em que partes do banco de dados residem em diferentes locais que estão conectados por uma rede de comunicação.

Para manter a atomicidade de uma transação multibanco de dados, é preciso ter um mecanismo de recuperação de dois níveis. Um gerenciamento de **recuperação global**, ou coordenador, é necessário para manter informações usadas para recuperação, além dos gerenciadores de **recuperação local** e as informações que eles mantêm (log, tabelas).

O coordenador costuma seguir um protocolo chamado **protocolo de confirmação em duas fases**, cujas fases podem ser indicadas da seguinte forma:

a) **Fase 1**.

b) **Fase 2**.

O efeito final do protocolo de confirmação em duas fases é que ou todos os bancos de dados participantes confirmam o efeito da transação ou nenhum deles o faz.

Caso qualquer um dos participantes — ou o coordenador — falhe, sempre é possível recuperar para um estado em que ou a transação é confirmada ou ela é revertida. Uma falha durante ou antes da Fase 1 normalmente requer que a transação seja revertida, enquanto uma falha durante a Fase 2 significa que uma transação bem-sucedida pode se recuperar e ser confirmada.

Fase 1

Quando todos os bancos de dados participantes sinalizam ao coordenador que a parte da transação multibanco de dados que envolve cada um tiver sido concluída, o coordenador envia uma mensagem de preparação para confirmação a cada participante, para que se preparem para confirmar a transação. Cada banco de dados participante, ao receber essa mensagem, forçará a gravação de todos os registros do log e as informações necessárias para a recuperação local em disco, e depois enviará um sinal **pronto para confirmação** ou **OK** ao coordenador. Se a gravação forçada em disco falhar ou a transação local não puder confirmar por alguma razão, o banco de dados participante enviará um sinal **não posso confirmar** ou **não OK** ao coordenador. Se o coordenador não receber uma resposta do banco de dados dentro de certo limite de tempo, ele assume uma resposta **não OK**.

Fase 2

Se todos os bancos de dados participantes responderem **OK** e o voto do coordenador também for **OK**, a transação terá sido bem-sucedida, e o coordenador envia um sinal de **confirmação** para a transação aos bancos de dados participantes. Como todos os efeitos locais da transação e as informações necessárias para a recuperação local foram registrados nos logs dos bancos de dados participantes, a recuperação da falha agora é possível. Cada banco de dados participante completa a confirmação da transação ao gravar uma entrada (*commit*) para a transação no log e ao atualizar permanentemente o banco de dados, se necessário. Contudo, se um ou mais dos bancos de dados participantes ou o coordenador tiverem uma resposta **não OK**, a transação terá falhado, e o coordenador enviará uma mensagem para **reverter** ou **UNDO** (desfazer) o efeito local da transação a cada banco de dados participante. Isso é feito ao desfazer as operações da transação, usando o log.

15

3 - BACKUP E RECUPERAÇÃO DE BANCO DE DADOS CONTRA FALHAS CATASTRÓFICAS

Até aqui, todas as técnicas que discutimos se aplicam a falhas não catastróficas. Uma suposição chave foi a de que o **log do sistema é mantido no disco e não se perde como resultado da falha**. De modo semelhante, o diretório de sombra precisa ser armazenado no disco para permitir a recuperação quando a página de sombra for utilizada.

As técnicas de recuperação que discutimos usam as entradas no log do sistema ou no diretório de sombra para se recuperarem da falha ao retornar o banco de dados a um estado consistente.

O gerenciador de recuperação de um SGBD também precisa ser equipado para lidar com falhas mais catastróficas, como as falhas de discos.

A principal técnica utilizada para lidar com essas falhas é um *backup* do banco dos dados, em que o banco de dados inteiro e o log são periodicamente copiados para **um meio de armazenamento externo**, como fitas magnéticas ou outros dispositivos de armazenamento *off-line* de grande capacidade.

16

No caso de uma **falha catastrófica** do sistema, a cópia de *backup* mais recente pode ser recarregada da fita para o disco, e o sistema, reiniciado. Os dados de aplicações críticas, como bancos, seguros, mercado de ações e outros bancos de dados, são copiados de tempos em tempos em sua totalidade e movidos para locais seguros e fisicamente separados. Câmaras de armazenamento subterrâneas têm

sido usadas para proteção contra danos ocasionados por inundação, tempestade, terremoto ou incêndio.

Eventos como o ataque terrorista de 11 de setembro em Nova York (em 2001) e o desastre do furacão Katrina em Nova Orleans (em 2005) criaram uma maior conscientização da recuperação de desastres dos bancos de dados críticos aos negócios.

Para evitar perder todos os efeitos das transações que foram executadas desde o último *backup*, é comum fazer o *backup* do log do sistema em intervalos mais frequentes do que o do banco de dados inteiro, copiando-o periodicamente para dispositivos externos. O log do sistema costuma ser muito menor do que o próprio banco de dados, e, portanto, pode ser copiado com mais frequência. Portanto, os usuários não perdem todas as transações que realizaram desde o último *backup* do banco de dados.

Todas as transações confirmadas e registradas na parte do log do sistema que foi copiada para unidades de *backup* podem ter efeito sobre o banco de dados refeito. Um novo log é iniciado após cada *backup* do banco de dados. Assim, para recuperar-se da falha do disco, o banco de dados é primeiro recriado no disco com base em sua cópia de *backup* mais recente em fita. Depois disso, os efeitos de todas as transações confirmadas, cujas operações foram registradas nas cópias do log do sistema, são refeitos.

17

RESUMO

Neste módulo, aprendemos que:

k) Há várias técnicas para recuperação de falhas na transação. O objetivo principal da recuperação é garantir a propriedade de atomicidade de uma transação. Se uma transação falhar antes de terminar sua execução, o mecanismo de recuperação precisa garantir que a transação não possui efeitos duradouros no banco de dados.

l) Para que seja possível a recuperação, o SGBD de implementar mecanismos como caching, atualização no local ou sombra, imagens antes ou depois da cópia de um item de dados, operações de recuperação UNDO ou REDO, políticas *steal*/no-*steal* e force/no-force, check point do sistema e o protocolo de *logging write-ahead*.

m) Há duas técnicas diferentes para a recuperação: atualização adiada e atualização imediata.

n) As técnicas de atualização adiada postergam qualquer atualização real do banco de dados em disco até que uma transação atinja seu ponto de confirmação (*commit*). A transação força a gravação do log em disco antes de gravar as atualizações no banco de dados. Essa técnica, quando usada com certos métodos de controle de concorrência, é projetada para nunca exigir a

reversão (*rollback*) da transação, e a recuperação consiste simplesmente em refazer as operações das transações confirmadas após o último checkpoint do log. A desvantagem é que muito espaço em buffer pode ser necessário, pois as atualizações são mantidas nos buffers e não são aplicadas ao disco até que a transação seja confirmada. A atualização adiada pode levar a um algoritmo de recuperação conhecido como NO-UNDO/REDO. As técnicas de atualização imediata podem aplicar mudanças ao banco de dados no disco antes que a transação alcance uma conclusão bem-sucedida. Quaisquer mudanças aplicadas ao banco de dados devem primeiro ser registradas no log e forçar a gravação para o disco, de modo que essas operações possam ser desfeitas, se for preciso.

18

o) O algoritmo de recuperação para atualização imediata é conhecido como UNDO/REDO. Outro algoritmo, conhecido como UNDO/NO-REDO, também pode ser desenvolvido para atualização imediata se todas as ações da transação forem registradas no banco de dados antes da confirmação.

p) A técnica de paginação de sombra para a recuperação registra as antigas páginas do banco de dados usando um diretório de sombra. Essa técnica, que é classificada como NO-UNDO/NO-REDO, e não exige um log nos sistemas monousuário, mas ainda precisa do log para sistemas multiusuários.

q) O ARIES é um esquema de recuperação específico utilizado em muitos produtos de banco de dados relacional da IBM.

r) O protocolo de confirmação em duas fases é usado para recuperação de falhas que envolvem transações multibanco de dados.

s) A recuperação de falhas catastróficas costuma ser feita com o *backup* do banco de dados e do log em unidade externa (como fitas magnéticas). O log pode ser copiado com mais frequência do que o banco de dados, e o log de *backup* pode servir para refazer operações com base no último *backup* completo do banco de dados.

UNIDADE 3 – RESTAURO DE BANCO DE DADOS
MÓDULO 4 – PLANO DE *BACKUP* E RESTORE

01

1 - BENEFÍCIOS DO *BACKUP*

Anteriormente, vimos as características funcionais que o SGBD deve prover para que o sistema restabeleça falhas não catastróficas.

Neste momento, descreveremos os benefícios do *backup* dos bancos de dados, as condições de *backup* e restauração básicas, apresentaremos estratégias de *backup* e restauração para SGBD e considerações

de segurança sobre *backup* e restauração. Esses procedimentos são essenciais para criarmos arquivos de segurança que possam ser utilizados em eventos de falhas catastróficas ou mesmo em eventuais necessidades de recuperação do sistema.

Muitas empresas utilizam *backups* de ambientes de produção para criar ambientes à parte, cópias fiéis de produção, para que sejam realizados testes de performance, homologação ou mesmo treinamento. As rotinas de *backup* e restauro permitem criar essas cópias fiéis.

O componente de *backup* e restauração dos SGBDs oferece uma proteção essencial para dados críticos armazenados em bancos de dados. Para minimizar o risco de perda de dados catastrófica, você precisa fazer *backup* dos bancos de dados frequentes para preservar as modificações feitas nos dados.

Uma estratégia de *backup* e restauração bem-planejada ajuda a proteger bancos de dados contra perda de dados causada por várias falhas. Teste sua estratégia restaurando um conjunto de *backups* e recuperando depois seu banco de dados para se preparar para responder com eficiência a um desastre.

Este módulo apresenta conceitos gerais de *Backup* que não são padronizados entre todos os SGBDs, dessa forma, consulte o manual do SGBD que você manipula, de forma a ajustar os diferentes conceitos e limitações.

02

O *backup* dos bancos de dados do SGBD, a execução de procedimentos de restauração de teste nos *backups* e o armazenamento de cópias de *backups* em um local externo seguro evitam a perda de dados potencialmente catastrófica. Essa é a única maneira de proteger os dados do SGBD de forma confiável.

Com *backups* válidos de um banco de dados, você pode recuperar seus dados de muitas **falhas**, como:

- **Falha de mídia**.
- **Erros de operadores de sistema**, que, por exemplo, removem uma tabela por engano.
- **Problemas de *hardware***, por exemplo, uma unidade de disco danificada ou perda permanente de um servidor.
- **Desastres naturais**.

Além disso, os *backups* de um banco de dados são úteis para fins administrativos rotineiros, como: copiar um banco de dados de um servidor para outro, configurar o espelhamento do banco de dados ou fazer arquivamento.

Já falamos anteriormente, mas vamos relembrar as formas de criarmos cópias de bancos de dados.

Desastres naturais

Neste caso, é imprescindível criar um *backup* externo em uma região diferente daquela do seu local, para usar no caso de um desastre natural afetar seu local.

03

2 - IMAGEM DO DISCO

Uma das alternativas de criação de *backups* trata-se da criação de um arquivo especial que representa a **imagem binária do disco rígido**. Esse tipo de *backup* geralmente é feito por meio de um *software* especial de *backups* de imagem, que não faz parte nem do sistema operacional, nem do SGBD.

Os arquivos de imagem têm a grande vantagem de representar fielmente todos os arquivos contidos no disco, que podem incluir o sistema operacional, documentos de escritório, aplicações instaladas etc. Assim, ao restaurar uma cópia de imagem, com apenas uma operação, restaura-se todo o computador.

Em ambientes corporativos, o banco de dados geralmente não reside no mesmo disco dos sistemas operacionais, aplicativos e documentos, dessa forma, essa estratégia pode não ser a mais recomendada nesse contexto.

Para realizar um *backup* de imagem, as ferramentas modernas permitem que essa operação ocorra "**a quente",** ou seja, mesmo o computador estando ligado e funcionando, o sistema é capaz de gerar o *backup*. Por sua vez, essa operação consome muito recurso de disco, deixando o computador bastante lento enquanto a operação é processada. Também é possível executar a operação "**a frio**", ou seja, com o computador desligado, um disco de boot especial carrega apenas a ferramenta de *backup* de imagem que executa a operação de forma idêntica, porém mais rápida.

O destino de um *backup* de imagem sempre é outro disco rígido (ou uma outra partição no mesmo disco rígido). No local de destino, é necessário possuir um espaço livre igual ou maior que o total de dados ocupados pelo disco (ou partição) de origem. Dessa forma, o arquivo de imagem a ser gerado será equivalente ao total de dados ocupados no disco de origem. **Rotinas de compactação** são capazes de criar arquivos de imagem menores, até uma ordem de aproximadamente 50% em média do total de bytes ocupados. Veja um **exemplo.**

Exemplo

Suponha que o disco C de um computador possua o tamanho total de 1 Terabyte, imagine que o espaço ocupado pelo sistema operacional, aplicativos e documentos totalizem 200 Gigabytes. Ao utilizar um disco D como destino para o *backup* de imagem, esse disco D deve possuir, no mínimo, 200 GB de espaço livre. Entretanto, a rotina de compactação pode acabar por gerar um arquivo de imagem com tamanho aproximado de 100 GB.

04

É possível criarmos vários arquivos de imagem, obtendo assim **versões** do ambiente ao longo do tempo. Por exemplo, poderíamos realizar *backups* de imagem uma vez por mês do ambiente, e dessa forma manteríamos versões mensais. Caso se perca um arquivo em um momento futuro, poderíamos recorrer a uma imagem mais antiga para recuperá-lo.

O restauro de um arquivo de imagem pode ser feito de duas formas:

Restauro completo da imagem	Montagem da imagem
Esse é o método mais comum. Nesse caso, o disco destino é formatado e o conteúdo da imagem é recriado nesse disco exatamente como era no momento do *backup*. Com isso, o disco restaurado torna-se uma cópia idêntica do arquivo de origem. Ainda, é possível utilizar um disco de marca, modelo e até mesmo tamanho diferente do disco de origem, desde que o disco utilizado possua tamanho compatível com a imagem a ser restaurada.	Nesse método, o sistema operacional monta o arquivo de imagem como um outro drive de disco, por exemplo, disco D, e esse disco D passa a apresentar os arquivos da imagem, porém, **apenas para leitura**. Uma das vantagens desse método é poder acessar um arquivo que foi apagado ou uma versão anterior de um arquivo sem precisar modificar o ambiente original. Copia-se o arquivo para o local desejado e depois libera-se a imagem montada.

05

3 - MÁQUINAS VIRTUAIS

Máquinas virtuais compõem uma tecnologia moderna e poderosíssima quando procuramos um gerenciamento eficiente dos ambientes computacionais.

Uma máquina virtual refere-se a um drive de um computador que é criado dentro de um arquivo hospedeiro. Esse drive virtual recebe uma instalação de sistema operacional e/ou arquivos. Um

software de gerenciamento de máquinas virtuais (como o VMware) é capaz de gerenciar esse ambiente.

Assim, um computador físico pode conter várias máquinas virtuais, podendo compartilhar os recursos da máquina física em diversas máquinas virtuais.

Por exemplo, uma máquina física poderia ter uma máquina virtual que seria o ambiente de programação, outra máquina virtual que seria o SGBD e outra máquina virtual que seria o sistema de aplicação do usuário. A máquina hospedeira é capaz de dividir seus recursos entre essas máquinas virtuais. Exemplo: suponha que o computador hospedeiro possua 128 Megabytes. Podemos deixar 8 Megabytes para a máquina virtual de desenvolvimento, 20 MB para a máquina virtual do banco de dados, 80 MB para a máquina virtual do aplicativo do usuário e o restante de RAM para a própria máquina hospedeira.

Não é escopo desta matéria tratarmos de máquinas virtuais, portanto, vamos direto ao ponto: como são feitos *backups* e restauros de máquina virtuais.

06

Para o ambiente hospedeiro, uma máquina virtual representa um ou alguns arquivos em uma pasta específica. Realizar o *backup* de uma máquina virtual é simplesmente copiar essa pasta para outro lugar. Como uma máquina virtual representa todo um ambiente, é de se esperar que os arquivos que compõe cada máquina virtual possam ter dezenas de gigabytes ou até mesmo alguns terabytes de tamanho.

O restauro de uma máquina virtual também é feito mediante a cópia desses arquivos, do local de armazenamento da cópia de *backup* para o local de destino. Exemplo: suponha que você esteja em um ambiente de rede e que nessa rede exista uma pasta compartilhada de nome \\Servidor*Backup**Backups*DeMaquinasVirtuais. Efetuar um *backup* seria copiar os arquivos locais referentes à máquina virtual para esse local de destino, separando-o possivelmente em uma subpasta.

Imagine que o disco da máquina virtual estrague e um novo disco seja instalado. Para recuperar as máquinas virtuais nesse disco, basta copiar os arquivos que você salvou no servidor de *backup* compartilhado da rede.

Um servidor de banco de dados pode residir em uma máquina virtual, assim como os bancos de dados podem residir em outra máquina virtual. Dessa forma, fica mais fácil gerenciar *backups* do servidor e *backups* dos dados.

07

4 - CÓPIA DOS ARQUIVOS DE DADOS E LOG

Em um ambiente convencional (não virtual), um banco de dados e seu arquivo de log, geralmente, numa instalação mais simples e padrão, ocupam apenas dois arquivos, um de dados e um de log. Já estudamos anteriormente que, para fins de melhoria de performance, podemos separar um banco de dados em vários arquivos de dados. De qualquer forma, sempre teremos arquivos binários em um disco específico.

O *backup* de arquivos de dados e de log por meio de cópia simples pode ser uma boa estratégia. Nesse método, copiamos todos os arquivos referentes aos bancos de dados e logs que existam no computador para um local específico, que normalmente é um disco removível, uma pasta compartilhada em um servidor na rede, um outro disco local ou mesmo uma outra partição local.

O grande **problema** dessa técnica é que a grande maioria dos SGBDs não permite cópia de arquivo "a quente", ou seja, ao tentar copiar os arquivos de banco de dados e de log com o SGBD funcionando pode gerar um erro de violação de acesso, impedindo a cópia. Pare resolver isso, é necessário desligar o serviço do SGBD (fechar o programa).

O restauro nessa técnica é feito pela cópia simples de arquivos, sobrescrevendo os arquivos de destino pelos arquivos da cópia de *backup*. Porém, **todos os SGBDs exigem estarem desligados** para efetuar essa cópia (também é chamado de *backup* "a frio"). Isso pode ser ruim para ambientes de produção, mas pode ser bastante interessante para ambientes de programação, teste e homologação.

08

5 - OPERAÇÕES SQL PARA *BACKUP* E RESTAURO DE DADOS E DE LOG

Tradicionalmente, a forma mais utilizada de operações de *backup* são as **rotinas SQL de *backup* de dados e de log**. Essas rotinas permitem criar arquivos à parte com os dados das tabelas e as transações do log. Esse tipo de operação é o foco do nosso estudo e será aqui que investiremos nosso maior tempo de estudo. Começaremos apresentando informações básicas de como montar uma estratégia de *backup* e restauro e finalizaremos com os procedimentos básicos para *backup* e restauro.

5.1 - Componentes e conceitos

Precisamos inicialmente definir alguns nomes de operações e componentes comuns. Desta forma, ao tratarmos desses objetos, facilmente entenderemos o objetivo de cada um. Tenha muita atenção aos termos, pois são muito próximos na escrita, mas representam assuntos totalmente diferentes:

- Fazer *backup* (verbo);
- *Backup* (substantivo);
- Dispositivo de *backup*;
- Mídia de *backup*;
- *Backup* de dados;
- *Backup* de banco de dados;
- *Backup* diferencial;
- *Backup* completo;
- *Backup* de log;
- Recuperação;
- Modelo de recuperação;
- Restaurar.

Fazer *backup* (verbo)

Significa copiar os dados ou registros de log de um banco de dados e/ou de seu log de transações para um dispositivo de *backup*, como um disco ou uma fita, a fim de criar um *backup* de dados ou *backup* de log.

***Backup* (substantivo)**

Refere-se a uma cópia dos dados que podem ser usados para restaurar e recuperar os dados após uma falha. Os *backups* de um banco de dados também podem ser usados para restaurar uma cópia do banco de dados em um novo local.

Dispositivo de *backup*

Representa um disco ou dispositivo de fita no qual os *backups* serão gravados e nos quais eles poderão ser restaurados. Os *backups* também podem ser gravados em um serviço de armazenamento *on-line* ou na nuvem.

Mídia de backup

Refere-se a uma ou mais fitas ou arquivos de disco nos quais um ou mais backups foram gravados.

***Backup* de dados**

Representa um *backup* de dados em um banco de dados completo (um *backup* de banco de dados), um banco de dados parcial (um *backup* parcial) ou um conjunto de arquivos de dados ou grupos de arquivos (um *backup* de arquivo).

***Backup* de banco de dados**

Refere-se a um *backup* de um banco de dados. Os *backups* completos de banco de dados representam todo o banco de dados no momento em que o *backup* é concluído. Os *backups* de banco de dados diferenciais contêm somente alterações feitas no banco de dados desde seu *backup* completo de banco de dados mais recente.

***Backup* diferencial**

Representa um *backup* de dados que se baseia no *backup* completo mais recente de um banco de dados completo ou parcial ou um conjunto de arquivos de dados ou grupos de arquivos (a base diferencial) que contém somente os dados alterados desde essa base (explicaremos melhor mais adiante quando tratarmos das estratégias de *backup*).

***Backup* completo**

Trata-se de um *backup* de dados que contém todos os dados em um banco de dados ou em um conjunto de grupos de arquivos ou arquivos, além de log suficiente para permitir a recuperação desses dados.

***Backup* de log**

Representa um *backup* de logs de transações que inclui todos os registros de log dos quais não foi feito *backup* em um *backup* de log anterior (modelo de recuperação completa**).**

Recuperação

A recuperação refere-se ao ato de retornar um banco de dados a um estado estável e consistente. Para tal, a recuperação coloca o banco de dados em uma fase anterior à atual. Essa fase pode ser qualquer momento anterior à falha.

Modelo de recuperação

Refere-se à estratégia de recuperação utilizada, normalmente os SGBDs oferecem várias estratégias de recuperação, como por exemplo, a recuperação apenas do arquivo de dados, o desfazimento de transações até determinado momento ou a cópia fiel do sistema de acordo com o último *backup* gerado. O modelo de recuperação de banco de dados determina seus requisitos de *backup* e de restauração.

Restaurar

É o processo de copiar todos os dados e páginas de log de um *backup* do SGBD para um banco de dados especificado e, em seguida, aplicam-se todas as transações registradas no *backup*, realizando as alterações registradas para avançar os dados no tempo.

09

5.2 - Estratégias de *backup* e restauração

O uso confiável de *backup* e restauração para recuperação requer uma estratégia de *backup* e restauração. Uma estratégia de *backup* e restauração bem-planejada maximiza a disponibilidade dos dados e minimiza a perda de dados, considerando, ao mesmo tempo, seus requisitos empresariais específicos.

Coloque o banco de dados e os *backups* em dispositivos separados. Caso contrário, se o dispositivo que contém o banco de dados falhar, seus *backups* ficarão indisponíveis. Colocar os dados e *backups* em dispositivos separados também aprimora o desempenho de E/S dos *backups* gravados e o uso de produção do banco de dados.

Uma estratégia de *backup* e restauração contém um capítulo referente à **estratégia do *backup*** e outro capítulo referente à **estratégia da restauração**.

A parte de ***backup*** da estratégia define o tipo e a frequência dos *backups*, a natureza e velocidade do *hardware* exigido para eles, como os *backups* serão testados, e onde e como as mídias de *backup* devem ser armazenadas (incluindo considerações de segurança).

A parte de **restauração** da estratégia define quem é o responsável pela execução da restauração e como a restauração deve ser executada para atender às metas de disponibilidade do banco de dados e minimizar perda de dados. É recomendado que toda organização documente seus procedimentos de *backup* e restauração e mantenha uma cópia da documentação junto aos responsáveis pelo *backup* e pelo restauro.

10

A especificação de uma estratégia de *backup* e restauração eficaz requer planejamento, implementação e teste cuidadosos. O teste é obrigatório. Não existirá uma estratégia de *backup* até que você tenha restaurado com êxito os *backups* em todas as combinações incluídas na estratégia de restauração.

Para isso, você deve considerar uma variedade de fatores, que incluem o seguinte:

- As **metas de produção de sua organização** para os bancos de dados, especialmente os requisitos para disponibilidade e proteção contra perda de dados.
- A **natureza de cada um dos seus bancos de dados**: o tamanho, os padrões de uso, a natureza de seu conteúdo, os requisitos dos dados, e assim por diante.
- **Restrições de recursos**, como *hardware*, pessoal, espaço para armazenagem de mídia de *backup*, a segurança física da mídia armazenada, e assim por diante.

11

5.3 - Impacto do modelo de recuperação no *backup* e na restauração

As operações de *backup* e restauração ocorrem dentro do contexto de um modelo de recuperação. Um modelo de recuperação é uma propriedade de banco de dados que controla a forma de gerenciamento do log de transações. Além disso, o modelo de recuperação de um banco de dados determina para quais tipos de *backups* e cenários de restauração o banco de dados oferece suporte.

Geralmente, um banco de dados usa o modelo de recuperação simples ou o modelo de recuperação completa.

A melhor escolha do modelo de recuperação para o banco de dados depende de seus requisitos corporativos. Para evitar o gerenciamento de log de transações e simplificar o *backup* e a restauração, use o modelo de **recuperação simples**. Para minimizar a exposição à perda de trabalho, às custas de uma sobrecarga administrativa, use o modelo de **recuperação completa**.

12

5.4 - Planejar a estratégia de *backup*

Depois de selecionar um modelo de recuperação que satisfaça seus requisitos empresariais para um banco de dados específico, você precisa planejar e implementar uma estratégia de *backup* correspondente.

A melhor estratégia de *backup* depende de uma série de **fatores**, dos quais os seguintes são especialmente significativos:

- Quantas horas ao dia os aplicativos precisam acessar o banco de dados?
- Com que frequência as alterações e atualizações deverão ocorrer?
- As alterações ocorrem geralmente em uma pequena parte do banco de dados ou em uma grande parte do banco de dados?
- Quanto espaço em disco é necessário para um *backup* completo de banco de dados?

Quantas horas ao dia os aplicativos precisam acessar o banco de dados?

Se houver um período de pouca atividade previsível, recomendamos que você agende *backups* de banco de dados completos para aquele período.

Com que frequência as alterações e atualizações deverão ocorrer?

Se as alterações forem frequentes, considere o seguinte:

- No modelo de recuperação simples, agende *backups* diferenciais entre os *backups* de banco de dados completos. Um *backup* diferencial captura só as alterações desde o último *backup* completo do banco de dados (detalharemos mais adiante).
- No modelo de recuperação completa, você deve agendar *backups* de log frequentes. O agendamento de *backups* diferenciais entre *backups* completos pode reduzir o tempo de restauração reduzindo o número de *backups* de log a serem restaurados após a restauração dos dados.

As alterações ocorrem geralmente em uma pequena parte do banco de dados ou em uma grande parte do banco de dados?

Para um banco de dados grande no qual mudanças estão concentradas em uma parte dos arquivos ou grupos de arquivos, *backups* parciais e *backups* de arquivo podem ser úteis.

Quanto espaço em disco é necessário para um *backup* completo de banco de dados?

Dependendo da quantidade de versões de *backup* que seja desejável armazenar, o espaço de *backup* pode ser muito maior do que o próprio espaço ocupado pelo banco de dados.

13

5.5 - Estimar o tamanho de um *backup* de banco de dados completo

Antes de implementar uma estratégia de *backup* e restauração, calcule quanto espaço em disco um *backup* de banco de dados completo usará.

A operação de *backup* copia os dados no banco de dados para o arquivo de *backup*. O *backup* contém só os dados reais no banco de dados e não qualquer espaço não utilizado (exemplo: blocos de disco alocado e não utilizado). Portanto, o *backup* é geralmente menor do que o próprio banco de dados. Entretanto, ao se desejar armazenar muitas versões de *backups* pode-se ocupar um espaço em disco muito maior que o banco de dados.

5.6 - Agendar *backups*

A execução do *backup* tem um efeito mínimo sobre as transações em andamento; portanto, as operações de *backup* podem ser realizadas durante a operação regular. Você pode executar um *backup* do SQL Server com um efeito mínimo sobre as cargas de trabalho de produção.

Depois de decidir os tipos de *backups* necessários e a frequência de execução de cada tipo, recomendamos que você agende *backups* regulares como parte de um plano de manutenção de banco de dados para o banco de dados.

Para obter informações sobre planos de manutenção e como criá-los para fazer *backups* de banco de dados e *backups* de log, consulte o manual de agendamento de tarefas do seu SGBD.

14

5.7 - Testar seus *backups*

Não existirá uma estratégia de restauração até que você tenha testado seus *backups*.

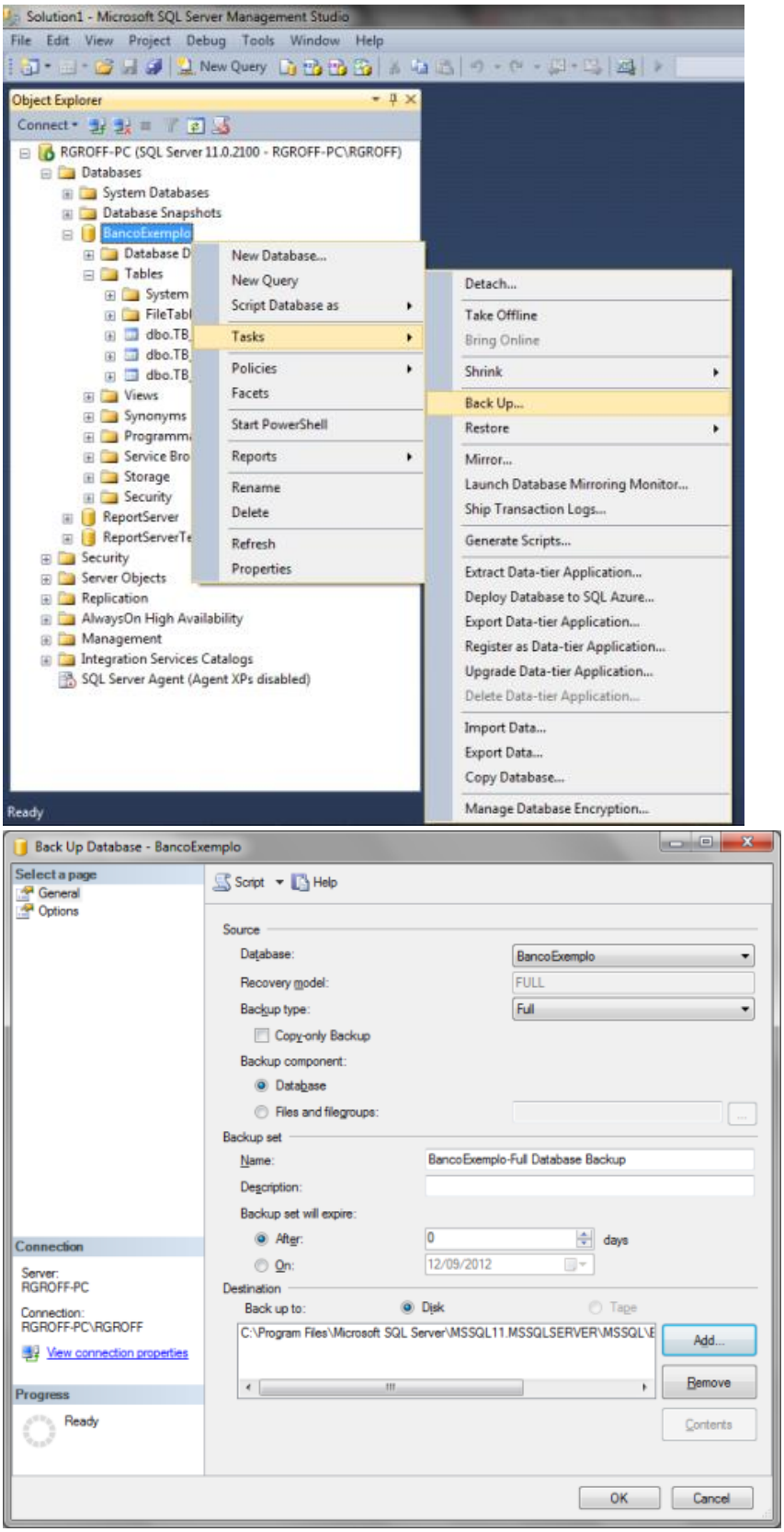
É muito importante testar sua estratégia de *backup* completamente para cada um dos bancos de dados, restaurando uma cópia do banco de dados em um sistema de teste. É necessário testar a restauração de cada tipo de *backup* que você pretende usar.

É recomendável que você mantenha um manual de operações para cada banco de dados. Esse manual operacional deve documentar o local dos *backups*, os nomes do dispositivo de *backup* (se houver) e o tempo necessário para restaurar os *backups* de teste.

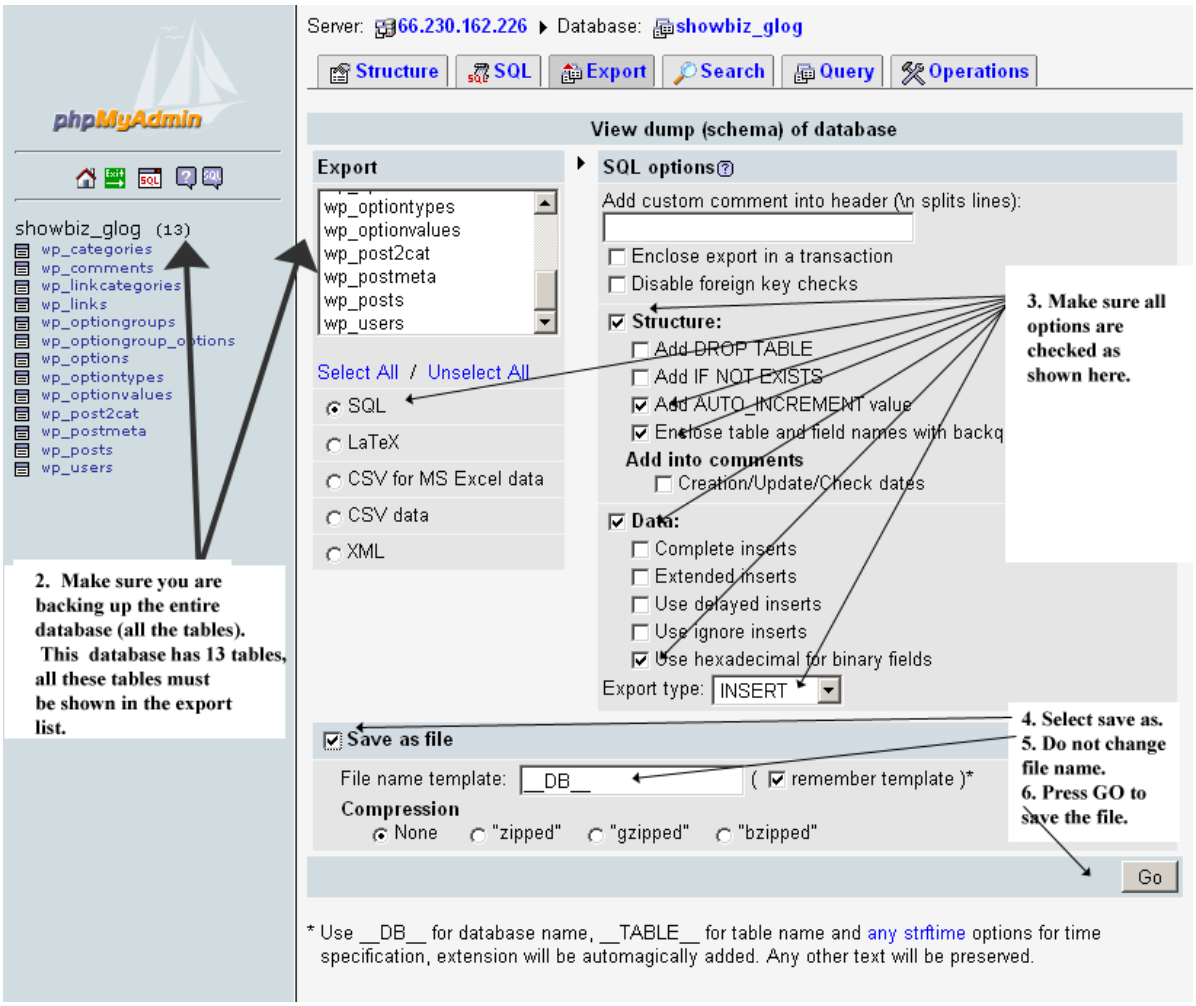
5.8 - Procedimentos de *backup* e restauro

Embora a SQL possua comandos de linha para a execução de *backup*, a grande parte dos SGBDs utilizam aplicativos complementares com interface gráfica que permitem a realização dos *backups* e restauros. Consulte o manual do seu SGBD para verificar o funcionamento dos procedimentos de *backup* e restore.

Veja abaixo dois exemplos de interfaces gráficas, para o MS SQL Server e o MySQL:



Interface gráfica do MS SQL Server



Interface gráfica do My SQL (PHPMyAdmin)

15

O comando SQL básico para realizar um *backup* é o comando *BACKUP* e, por padrão, tem o seguinte formato:

BACKUP DATABASE <nome do banco de dados> TO <dispositivo> <configurações complementares>

Os dispositivos mais comuns são DISK (um disco, uma pasta compartilhada, um IP de um servidor de armazenamento remoto ou um endereço de storage) e TAPE (uma fita removível).

As configurações complementares mais comuns envolvem opções para compactação do arquivo de *backup*, criptografia do arquivo de *backup*, adição de textos descritivos para o *backup*, entre outras.

Por exemplo, para realizar o *backup* completo do banco LOJA no arquivo de nome LOJA_BCK, usaríamos o seguinte comando:

```
BACKUP DATABASE LOJA TO DISK = 'd:\Backup\LOJA_BCK'
```

Outro exemplo, para realizar o *backup* completo do banco LOJA numa fita magnética de nome Tape0, usaríamos o seguinte comando:

```
BACKUP DATABASE LOJA TO TAPE = '\\.\Tape0'
```

16

Para realizar o *backup* do log de um banco de dados, utiliza-se o comando *BACKUP* LOG, com a mesma sintaxe do comando *BACKUP* DATABASE. Veja um exemplo do *backup* do log do banco LOJA no arquivo de nome LOJA_LOG_BCK

```
BACKUP LOG LOJA TO DISK = 'd:\Backup\LOJA_LOG_BCK'
```

O comando SQL básico para realizar um restauro é o comando RESTORE e, por padrão, tem o seguinte formato:

```
RESTORE DATABASE <nome do banco de dados> FROM <dispositivo> <configurações complementares>
```

As configurações complementares mais comuns envolvem a técnica de restauro (parcial ou completo), se o log deve ou não ser restaurado, se deve ser restaurado até um determinado momento, entre outras opções.

Por exemplo, para restaurar o banco LOJA do arquivo de nome LOJA_BCK, usaríamos o seguinte comando:

```
RESTORE DATABASE LOJA FROM DISK = 'd:\Backup\LOJA_BCK'
```

Novamente, cada SGBD possui uma série de funcionalidades extras cuja sintaxe SQL não é padronizada, consulte o manual do SGBD que você estiver utilizando para ver a sintaxe desses comandos.

17

5.9 - Tipos e estratégias de *backup*

Os SGBDs modernos permitem a realização de vários tipos de *backup*. Cada modelo apresenta suas características. **Não há técnica melhor ou pior**, cada técnica apresenta-se melhor em cada situação ou política da organização. Bancos de dados com diferentes usos em uma organização podem usar estratégias diferentes de *backup*. Portanto, não assuma um modelo como o ideal para toda sua organização, analise cada banco de dados para ver qual a melhor estratégica em cada caso.

5.9.1 - Backup no modelo de recuperação simples

Este modelo realiza **apenas o *backup* dos dados** mantidos no banco de dados. A restauração deste modelo permite restaurar apenas os dados contidos no banco de dados no momento da realização do *backup*. Neste modelo, as transações realizadas após o *backup* são perdidas e não é possível recuperar o banco de dados em um momento específico, a recuperação será feita para o momento em que o último *backup* foi realizado.

Veja um exemplo: Suponha que uma rotina de *backup* de recuperação simples seja agendada para ocorrer todo dia 1 e 15, de todos os meses. Suponha que no dia 14 ocorra uma falha do banco de dados. A recuperação fará com que o banco de dados volte à situação em que ele se encontrava no dia do último *backup*, ou seja, no dia 1º daquele mês. Todas as alterações sofridas entre os dias 1º e 14 serão perdidas. Perceba que nesse modelo, o risco de perda de informação no dia 1º é zero (pois nesse dia ocorreu um *backup*), no dia 2 o risco é um pouco maior (pois informações foram alteradas nesse dia), no dia três o risco é ainda maior (pois mais informações foram alteradas) e assim sucessivamente, até o dia do próximo *backup* onde o risco torna-se zero novamente. Ou seja, a cada ciclo, o risco inicia como zero e vai aumento até o próximo *backup*.

18

Dessa forma, no modelo de recuperação simples, depois de cada *backup*, o banco de dados fica sujeito à possível perda de trabalho na eventualidade de um desastre. A possibilidade de perda de trabalho aumenta a cada atualização até que o próximo *backup* seja feito, quando a possibilidade de perda de trabalho retorna a zero, tendo início um novo ciclo de exposição à perda de trabalho. A possibilidade de perda de trabalho aumenta com o passar do tempo entre os *backups*.

A ilustração a seguir mostra a exposição à perda de trabalho de uma estratégia de *backup* que usa apenas *backups* de banco de dados completos:

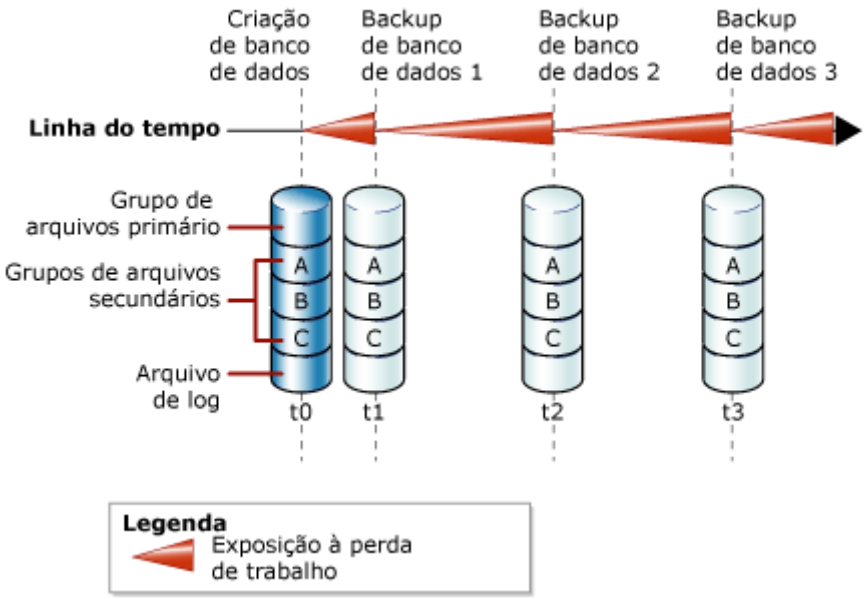


Diagrama que representa o risco à perda de informações entre os ciclos de *backup*.

O comando SQL que realiza este tipo de operação é o *BACKUP* DATABASE. Nessa estratégia, apenas um agendamento de *backup* é necessário, por exemplo, uma vez por semana.

19

5.9.2 - Backup no modelo de recuperação completa

Este modelo realiza **o *backup* dos dados e do log** mantidos no banco de dados. A restauração deste modelo permite restaurar não apenas os dados contidos no banco de dados no momento da realização do *backup*, mas de todas as transações até o momento da falha.

Neste modelo, as transações realizadas após o *backup* dos dados são *backup*eadas entre os ciclos de *backup* e, portanto, possível de recuperar o banco de dados em um momento específico até o momento da falha.

Veja um exemplo: Suponha que uma rotina de *backup* de recuperação simples seja agendada para ocorrer todo dia 1 e 15, de todos os meses e uma rotina de *backup* de log seja agendada para ocorrer todos os dias. Suponha que no dia 14 ocorra uma falha do banco de dados. A recuperação **do arquivo de dados** fará que o banco de dados volte à situação em que ele se encontrava no dia do último *backup*, ou seja, no dia 1º daquele mês. Após a recuperação do arquivo de dados, é possível aplicar os *backups* dos arquivos diários de log, trazendo o banco para o momento do último *backup* de log, ou seja, provavelmente no dia 13 ou dia 14 (dependendo do horário da falha e do *backup* do log). Dessa forma,

nenhuma alteração sofrida entre os dias 1º e 13 (ou 14) será perdida. Perceba que nesse modelo, o risco de perda de informação é de no máximo um dia.

20

Dessa forma, no modelo de recuperação completa, os *backups* de banco de dados são necessários, mas não são suficientes. Os *backups* de log de transações também são necessários.

A ilustração a seguir mostra a estratégia de *backup* menos complexa possível no modelo de recuperação completa.

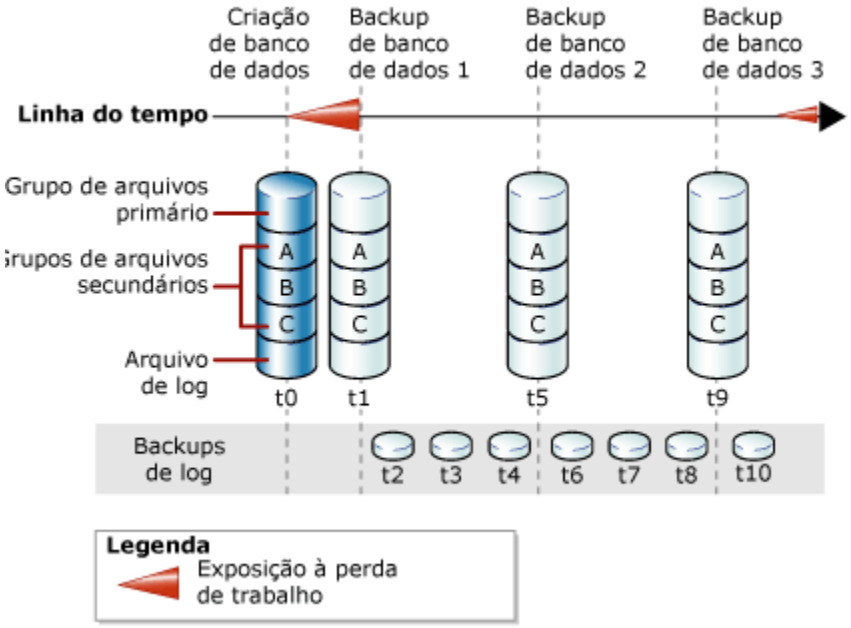


Diagrama que representa o risco à perda de informações entre os ciclos de *backup*.

O comando SQL que realiza este tipo de operação é o *BACKUP* DATABASE para o *backup* dos dados e *BACKUP* LOG para o *backup* dos logs. Nessa estratégia, dois agendamentos são necessários, um de ciclos mais espaçados para o *backup* dos dados (por exemplo, uma vez por semana) e outro em ciclos bem pequenos para o *backup* dos logs (por exemplo, uma vez ao dia).

21

5.9.3 - *Backup* completo x *backup* diferencial

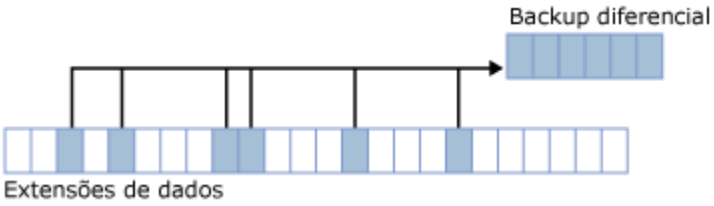
O *backup* completo contém todos os dados de um banco de dados. Caso o banco de dados seja muito grande, a operação de *backup* desse banco pode ser bem demorada (às vezes pode levar horas). O

espaço necessário para guardar várias versões de *backups* completos dessa natureza pode ser inviável.

Uma estratégia para esse tipo de situação é a realização de *backup* diferencial.

Um ***backup* diferencial** contém apenas os dados alterados após o último *backup*, ou seja, ele funciona como um complemento de um *backup* de dados. Enquanto que o restauro de um *backup* de dados dá-se apenas pelo restauro do arquivo de *backup* de dados, o restauro de um *backup* diferencial existe primeiramente o restauro do último arquivo de dados e o restauro do arquivo diferencial.

Veja a imagem abaixo. Suponha que o banco de dados é composto por todos os pequenos retângulos da imagem, e suponha que apenas as caixinhas em azul representam os dados modificados desde o último *backup* completo. O *backup* diferencial representará essas caixinhas em azul:



Banco de dados e o *backup* diferencial

22

As maiores **vantagens** de um *backup* diferencial são:

- A criação de um *backup* diferencial pode ser muito rápida se comparada à criação de um *backup* completo.
- Mesmo em um banco muito grande, se o número de alterações que ele sofrer for muito pequeno, o *backup* diferencial é igualmente pequeno.
- A restauração de um *backup* diferencial é muito mais simples do que a restauração de um *backup* usando *backups* de log de transações.

Exemplo: suponha que no dia 1º do mês você faça um *backup* completo, e nos demais dias você faça *backups* diferenciais. Caso no dia 14 ocorra uma falha do banco, o restauro desse banco se dará pelo restauro do *backup* de disco realizado no dia 1º e pelo restauro diferencial do banco realizado no dia 13. Os *backups* diferenciais feitos entre o dia 2 e o dia 12 podem ser desprezados. Dessa forma, basta guardar apenas dois arquivos, o arquivo do último *backup* de dados e o arquivo do último *backup*

diferencial. Após realizar um novo *backup* de dados, podem ser eliminados todos os *backups* diferenciais antigos.

A **diferença** entre o restauro de um banco diferencial e o restauro dos arquivos de log é que no caso do *backup* diferencial, apenas um arquivo diferencial é necessário, enquanto que o restauro do log, todos os arquivos sequenciais de log são necessários.

Vejamos dois exemplos:

- Estratégia 1
- Estratégia 2

Embora possa parecer vantajoso possuir apenas *backups* diferenciais, eliminando os *backups* de log, isso não é correto. O *backup* dos logs permite retornarmos um banco de dados a um momento específico, enquanto o *backup* diferencial não permite.

Estratégia 1

Backup de dados dias 1º e 15 de cada mês, *backups* de log diários, falha no dia 14. O restauro desse banco é feito restaurando o *backup* de dados do dia 1º e o restauro dos treze arquivos de log subsequentes (restauro do log do dia 2, restauro do log do dia 3, restauro do log do dia 4 etc., até o restauro do log do dia 13).

Estratégia 2

Backup de dados dias 1º e 15 de cada mês, *backups* diferenciais diários, falha no dia 14. O restauro desse banco é feito restaurando o *backup* de dados do dia 1º e o restauro do *backup* diferencial do dia 13.

23

RESUMO

Neste módulo, aprendemos que:

a) Um plano de *backup* e restauro é essencial para que os dados possam ser preservados. Um bom plano deve prever todos os tipos de eventos de falha possíveis, bem como as estratégias para restauro de cada uma dessas falhas.

b) Todos os SGBDs modernos fornecem componentes para a realização de *backups* e restauros. Cabe ao DBA programar esses componentes para que as rotinas ocorram conforme os planos estabelecidos.

c) As políticas de *backup* devem prever todos os *backups* realizados bem como as diversas rotinas de restauro.

d) A imagem de disco é uma técnica de *backup* muito utilizada para *backup* de computadores. Também pode ser utilizada para *backup* de bancos de dados. Nessa técnica, é criado um arquivo que representa a cópia binária fiel do disco rígido ou de uma partição.

e) Uma máquina virtual reside em um ou mais arquivos dentro de um servidor. O *backup* de uma máquina virtual é realizado pelo *backup* desses arquivos.

f) A forma mais tradicional de realização de *backups* de bancos de dados é por meio de rotinas SQL de *backup* e restauro. Embora seja possível utilizar comandos SQL básicos para realização de *backups* e restauros, a grande maioria dos SGBDs modernos oferecem interfaces gráficas para essas operações.

g) O uso confiável de *backup* e restauração para recuperação requer uma estratégia de *backup* e restauração.

h) Um *backup* diferencial pode otimizar a recuperação de um sistema por precisar de apenas um *backup*. Já o *backup* dos logs, exige todos os *backups* após o último *backup* de dados.