Sistemas Distribuídos

Transações atômicas

Conteúdo

- O modelo transacional
  - Armazenamento estável
  - Primitivas transacionais
  - Propriedades das transações
  - Transações aninhadas
- Implementação
  - Área de trabalho privada
  - Listas de escrita adiantadas
  - Protocolo de aceitação em duas fases
- Controle de concorrência
  - Bloqueio
  - Controle otimista
  - Método do carimbo

INTRODUÇÃO

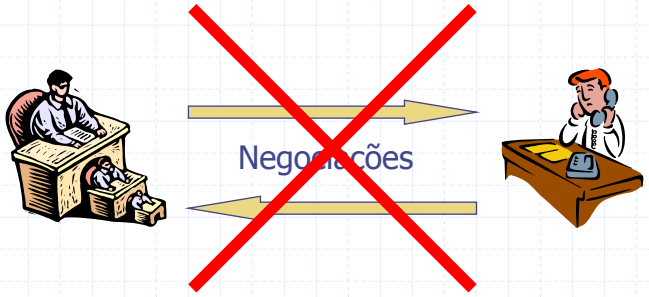
TRANSAÇÕES ATÔMICAS

Método de abstração de sincronização em sistemas distribuídos de nível mais alto, ocultando semáforos, monitores e troca de mensagens

VANTAGEM: o programador concentra-se na maneira que os processos trabalham em paralelo, nos algoritmos das aplicações

História -- Transações Atômicas

Modelo baseado no mundo de negócios: negociação entre empresas para fechar um contrato



Conceito evoluiu do funcionamento das máquinas dos anos 60 (tudo-ou-nada)

Transações Atômicas em Sistemas Distribuídos

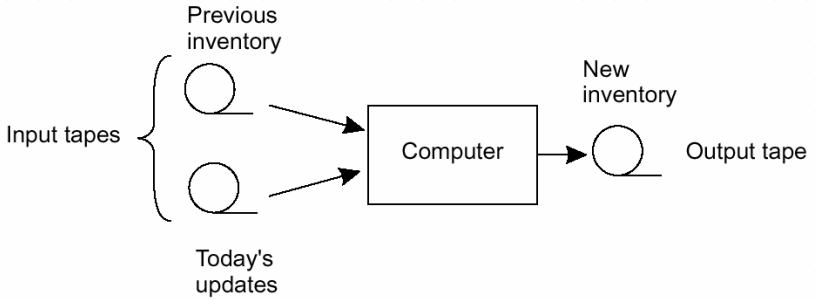
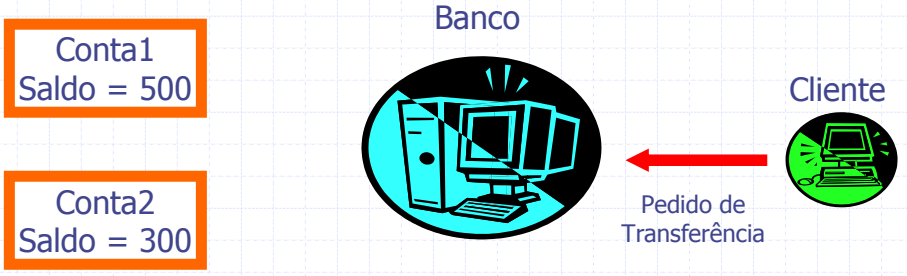


Fig. 5-17. Updating a master tape is fault tolerant.

Transações Atômicas em Sistemas Distribuídos

Exemplo: Transferência de Fundos Bancários

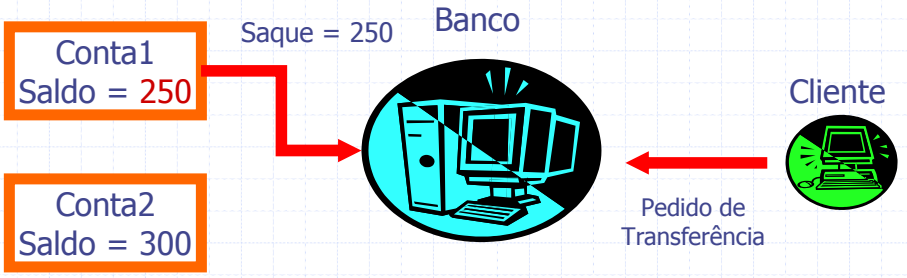
1- Saque (250, conta1)
2- Depósito (250, conta2)



Transações Atômicas em Sistemas Distribuídos

Exemplo: Transferência de Fundos Bancários

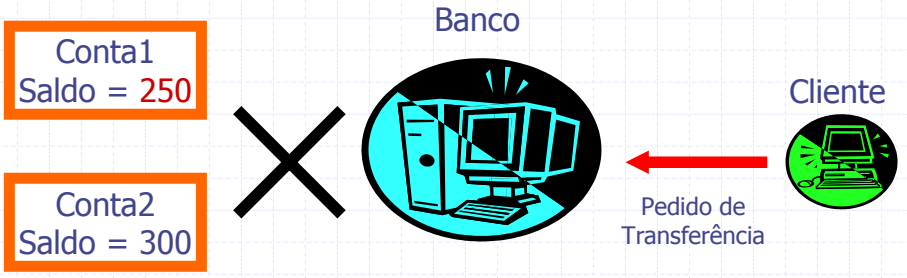
1- Saque (250, conta1)
2- Depósito (250, conta2)



Transações Atômicas em Sistemas Distribuídos

Exemplo: Transferência de Fundos Bancários

1- Saque (250, conta1)
2- Depósito (250, conta2)



Transações Atômicas em Sistemas Distribuídos

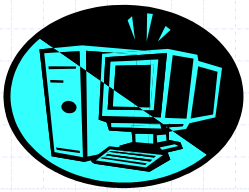
Exemplo: Transferência de Fundos Bancários

1- Saque (250, conta1)
2- Depósito (250, conta2)

Conta1
Saldo = 500

Conta2
Saldo = 300

Banco



Cliente

Operação
Abortada!

Modelo Transacional

O sistema transacional é constituído de um conjunto de processos independentes, podendo cada um falhar aleatóriamente.

A comunicação entre eles geralmente não é confiável, podendo haver perda de mensagens. Assim, veremos como os erros na comunicação serão tratados de forma transparente pelo software.

ARMAZENAMENTO ESTÁVEL

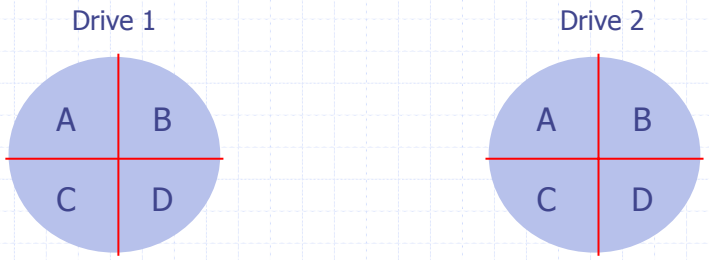
A informação é armazenada de três maneiras:

1- RAMs comuns – a informação é perdida quando há falha na alimentação ou quando a máquina apresenta algum defeito (volátil)

2- Discos – sobrevivem à falha do processador, porém podem se perder no caso de problemas da cabeça de leitura e gravação, entre outros.

ARMAZENAMENTO ESTÁVEL

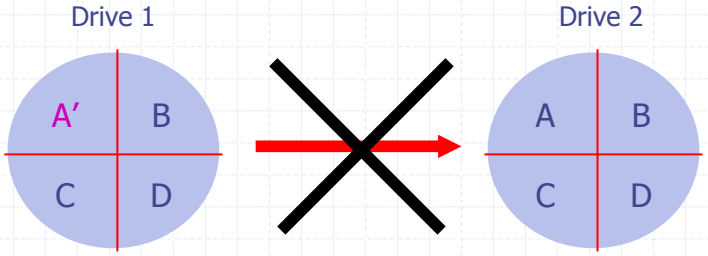
3- Memória estável – projetada para sobreviver a tudo, exceto a calamidades. Alto grau de tolerância a falhas.



O drive 2 possui uma cópia do drive 1

ARMAZENAMENTO ESTÁVEL

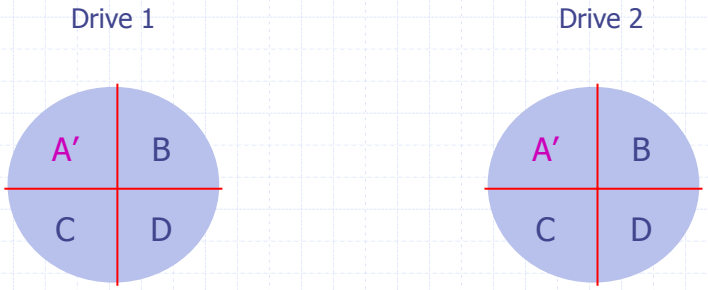
3- Memória estável – projetada para sobreviver a tudo, exceto a calamidades. Alto grau de tolerância a falhas.



Atualizado o Drive 1

ARMAZENAMENTO ESTÁVEL

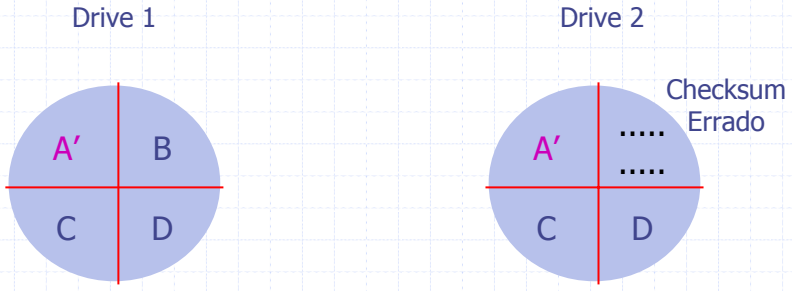
3- Memória estável – projetada para sobreviver a tudo, exceto a calamidades. Alto grau de tolerância a falhas.



O drive 2 é atualizado assim que o sistema é recuperado

ARMAZENAMENTO ESTÁVEL

3- Memória estável – projetada para sobreviver a tudo, exceto a calamidades. Alto grau de tolerância a falhas.



Regenera o bloco defeituoso a partir do drive 1

PRIMITIVAS TRANSACIONAIS

- Primitivas para programação. Ex.:
  - •BEGIN_TRANSACTION: marca o início da transação
  - •END_TRANSACTION: termina a transação e tenta fazer o *commit*
  - •ABORT_TRANSACTION: destrói a transação; restaura os valores anteriores (ao início da transação)
  - •READ: lê dados de um objeto (por exemplo, um arquivo)
  - •WRITE: escreve dados em um objeto
- Primitivas dependem da aplicação.

PRIMITIVAS TRANSACIONAIS

BEGIN_TRANSACTION

.....
.....
.....
.....
.....
.....

} CORPO DA TRANSAÇÃO

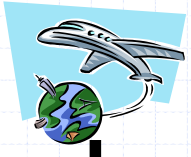
END_TRANSACTION

SÃO USADAS PARA DELIMITAR O ESCOPO DE UMA TRANSAÇÃO

PS. A REGRA É QUE TODAS OU NENHUMA DESTAS OPERAÇÕES SÃO EXECUTADAS.

PRIMITIVAS TRANSACIONAIS

Exemplo de uma transação completa



São Paulo

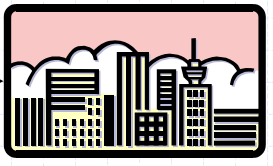
BEGIN_TRANSACTION

Reserva São Paulo - Campinas

Reserva Campinas - Araçatuba

Reserva Araçatuba - Campo Grande

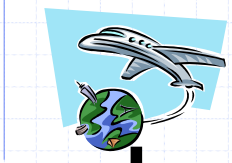
END_TRANSACTION



Campo Grande

PRIMITIVAS TRANSACIONAIS

Exemplo de uma transação abortada



São Paulo

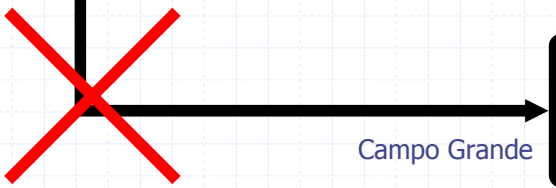
BEGIN_TRANSACTION

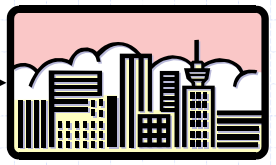
Reserva São Paulo - Campinas

Reserva Campinas - Araçatuba

Araçatuba - Campo Grande (LOTADO)

ABORT_TRANSACTION





Campo Grande

PROPRIEDADES DAS TRANSAÇÕES

- SERIALIZAÇÃO – As transações concorrentes não podem interferir umas com as outras.
- ATOMICIDADE – As transações devem parecer indivisível para o mundo exterior.
- PERMANÊNCIA – Havendo acordo entre as partes, as mudanças efetuadas passam a ser permanentes.

SERIALIZAÇÃO

- Serialização: assegura que se duas ou mais transações estiverem rodando, o resultado final aparece como se todas rodassem seqüencialmente em alguma ordem, ou seja, cada uma não deve poder interferir nas outras.
- Exemplo da execução de 3 transações para diferentes escalonamentos
  - escalonamento 1: serializado e legal
  - escalonamento 2: não serializado e legal
  - escalonamento 3: não serializado e ilegal

SERIALIZAÇÃO

```
BEGIN_TRANSACTION
        x = 0;
        x = x + 1;
END_TRANSACTION
```

```
BEGIN_TRANSACTION
        x = 0;
        x = x + 2;
END_TRANSACTION
```

```
BEGIN_TRANSACTION
        x = 0;
        x = x + 3;
END_TRANSACTION
```

TRÊS TRANSAÇÕES SENDO EXECUTADAS SIMULTANEAMENTE

SERIALIZAÇÃO ESCALONAMENTOS

Escalonamento 1	Escalonamento 2	Escalonamento 3
x = 0;	x = 0;	x = 0;
x = x + 1;	x = 0;	x = 0;
x = 0;	x = x + 1;	x = x + 1;
x = x + 2;	x = x + 2;	x = 0;
x = 0;	x = 0;	x = x + 2;
x = x + 3;	x = x + 3;	x = x + 3;
Legal	Legal	Ilegal
Serializado	Não Serializado	Livre Ordenação

ATOMICIDADE

- Atomicidade: as transações devem permanecer indivisíveis para o mundo exterior, ou seja, ou todos os comandos da transação são executados ou nenhum comando é executado.
- Enquanto a transação estiver em curso nenhum outro processo tem acesso aos estados intermediários
  - Ex.: Um determinado arquivo tem 10 bytes quando se inicia uma transação que tem como objetivo acrescentar novos bytes a tal arquivo. Se algum outro processo vier a ler o arquivo enquanto a transação estiver se realizando, ele só vai enxergar os 10 bytes originais.

PERMANÊNCIA

- Permanência: uma vez que as partes envolvidas em uma transação entrem em acordo, as mudanças efetuadas passam a ser permanentes
  - Ex.: Entre 2 empresas, primeiramente são feitos acordos verbalmente, podendo ser cancelado a qualquer momento. A partir do instante em que esses acordos são firmados através de um contrato, ambas as partes devem seguir em frente e os resultados obtidos devem ser permanentes, isto é, nenhuma alteração interna em ambas as partes poderá alterar o resultado obtido.

TRANSAÇÕES ANINHADAS

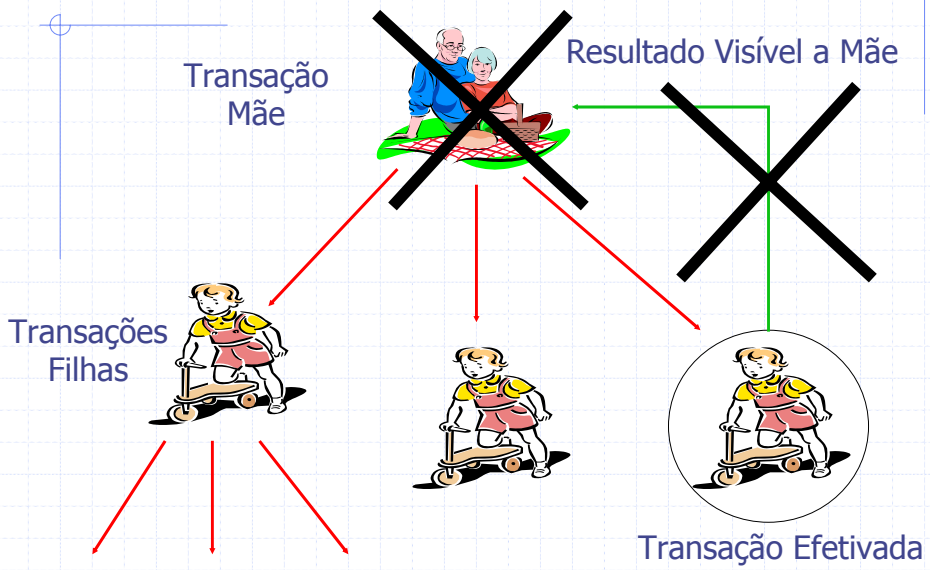
- As transações podem conter subtransações
- Estas subtransações podem rodar em processos diferentes
  - Subtransação de alto nível, que pode dar origem a vários filhos para rodar em paralelo uns com os outros, em processadores diferentes.
- OBJETIVO: Melhorar a performance e simplificar a programação.

TRANSAÇÕES ANINHADAS

PROBLEMA: Caso uma transação de mais alto nível venha a abortar (restarurando os valores iniciais), os resultados processados por uma transação filha devem ser desconsiderados, furando a propriedade da permanência

→ a propriedade de permanência só vale para a transação de mais alto nível

TRANSAÇÕES ANINHADAS



IMPLEMENTAÇÃO

Métodos usados na Implementação das Transações

- Área de Trabalho Privada
- Lista de Escrita Adiantadas
- Protocolo de Aceitação em Duas Fases

ÁREA DE TRABALHO PRIVADA

Quando uma transação se inicia ela recebe uma área de trabalho privada. Tudo que o processo ler ou escrever vai para esta área, ao invés de ir para a área "real" (sistema de arquivos e objetos normal do sistema).

DESVANTAGEM: O custo de copiar tudo para uma área privada é proibitivo.

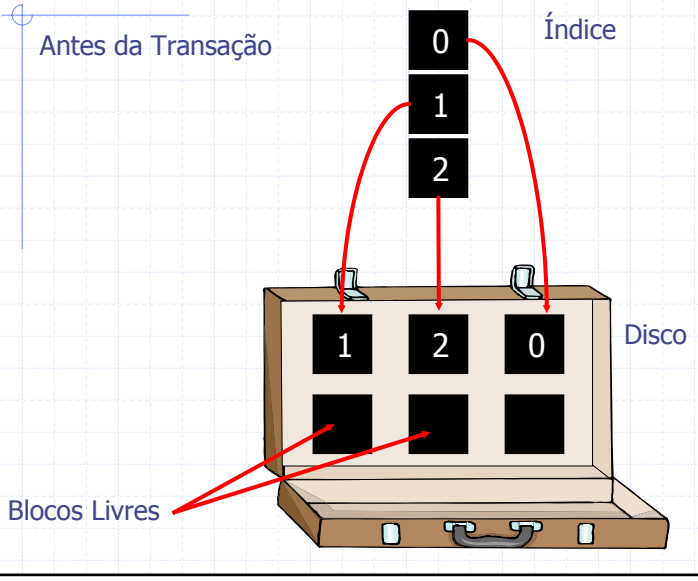
ÁREA DE TRABALHO PRIVADA

Otimizações:

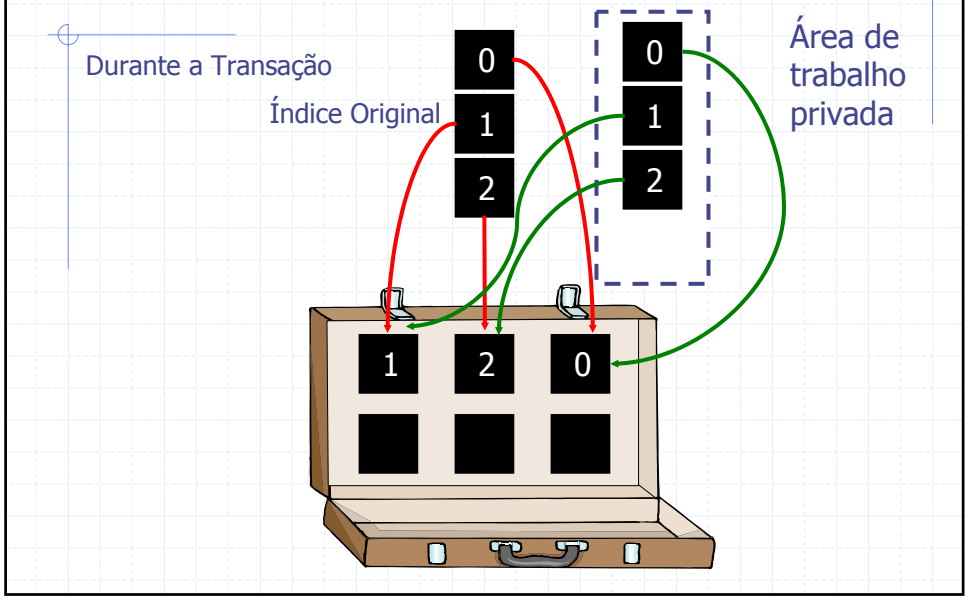
1ª - Se o processo somente lê o arquivo não há necessidade da cópia. Necessidade de cópia apenas para os arquivos modificados, fazendo leituras da área de trabalho real. Quando iniciada uma transação, a área de trabalho privada é vazia.

2ª - Copia-se para a área privada apenas a tabela de índices dos arquivos utilizados. Se o processo modificar o arquivo faz-se a cópia dos blocos modificados para a área privada e atualiza-se a tabela.

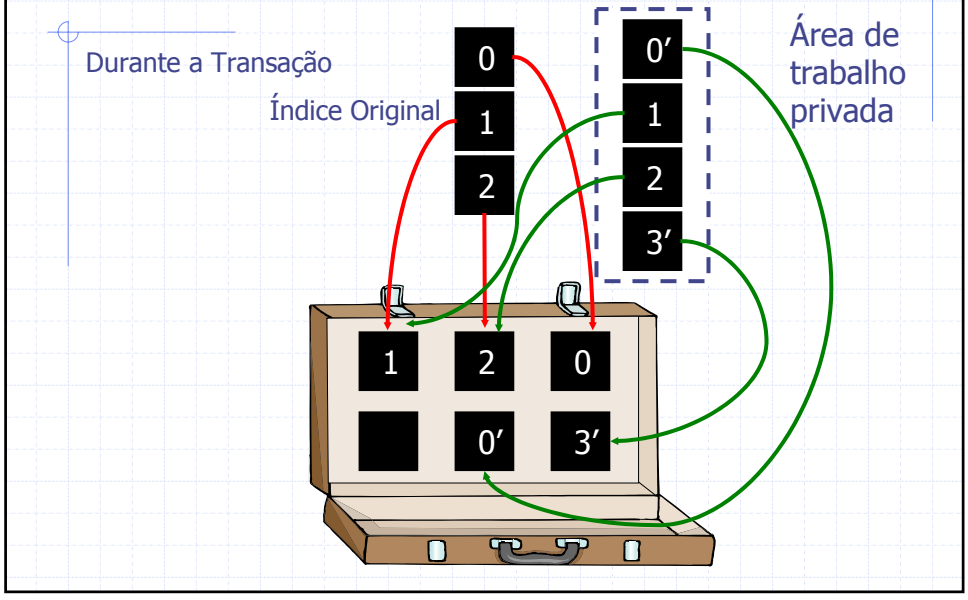
ÁREA DE TRABALHO PRIVADA



ÁREA DE TRABALHO PRIVADA

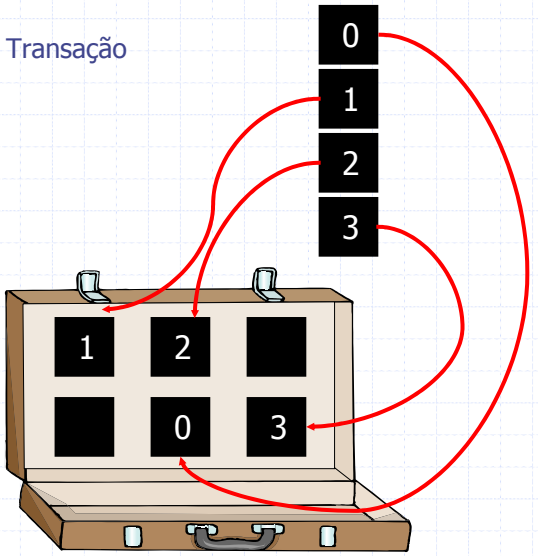


ÁREA DE TRABALHO PRIVADA



ÁREA DE TRABALHO PRIVADA

Após o término da Transação



LISTA DE ESCRITAS ADIANTADAS

As modificações são realmente realizadas em suas devidas posições, mas antes de modificar é feito um registro na lista de escritas adiantadas, na área de memória estável, informando:

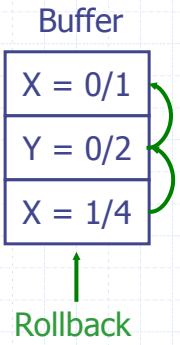
- qual a transação que está fazendo a mudança,
- qual o arquivo e bloco que está sendo modificado
- quais os valores antigos e os novos.

Só depois que estas informações forem escritas com sucesso, o arquivo é modificado

- se a transação tiver sucesso OK
- se a transação abortar a lista é utilizada para recuperar os valores originais (rollback)

LISTA DE ESCRITAS ADIANTADAS

```
x = 0;
y = 0;
BEGIN_TRANSACTION
        x = x + 1;
        y = y + 2;
        x = y * y;
END_TRANSACTION
```



O processo também pode ser utilizado na recuperação de falhas

PROTOCOLO DE ACEITAÇÃO EM DUAS FASES

- A ação de aceitação de uma determinada transação deve ser feita atomicamente e pode requerer a cooperação de vários processos rodando em máquinas diferentes, onde cada um trata de algumas das variáveis, arquivos e base de dados a serem modificadas no decorrer da execução.
- Protocolo de aceitação de transação (Gray - 1978).
- Um dos processos funciona como coordenador.

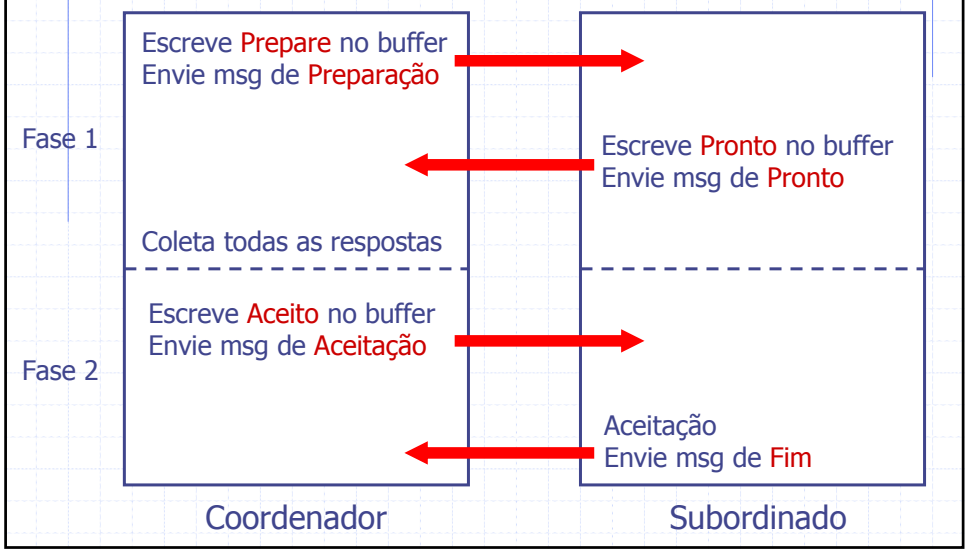
PROTOCOLO DE ACEITAÇÃO EM DUAS FASES

- O coordenador escreve em um buffer (memória estável) PREPARE, indicando início dos procedimentos de aceitação e envia msgs aos outros processos envolvidos na transação para que preparem a aceitação.
- Quando um subordinado recebe msg ele verifica se está preparado e escreve em um buffer PRONTO ou NÃO-PRONTO e envia a decisão de volta ao coordenador.

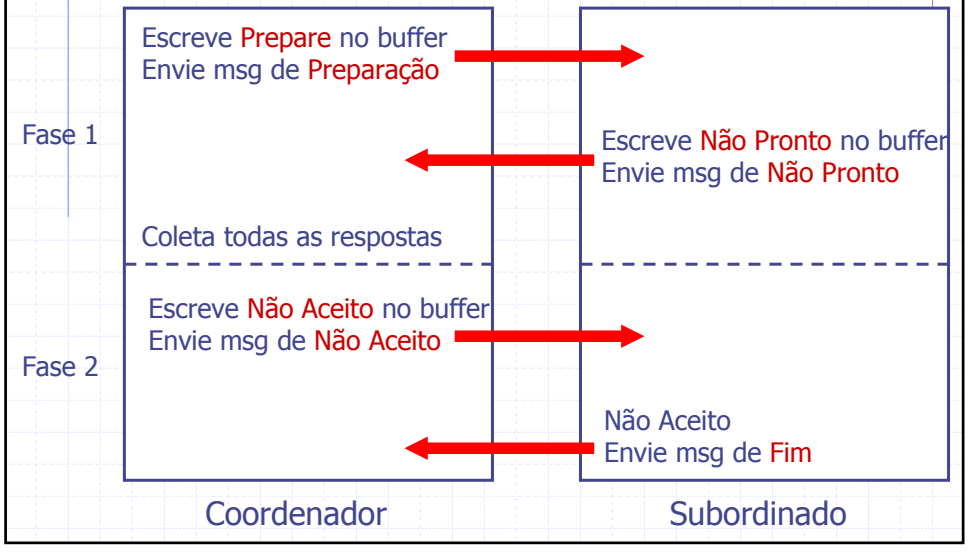
PROTOCOLO DE ACEITAÇÃO EM DUAS FASES

- Quando o coordenador tiver recebido todas as respostas, ele decide se a transação foi aceita ou não. Se um dos processos não tiver aceitado, a transação é abortada. Em ambos os casos ele escreve ACEITA ou NÃO-ACEITA no buffer e manda msgs aos subordinados.
- Como o buffer está em memória estável, este protocolo é tolerante a falhas.

PROTOCOLO DE ACEITAÇÃO EM DUAS FASES



PROTOCOLO DE ACEITAÇÃO EM DUAS FASES



PROTOCOLO DE ACEITAÇÃO EM DUAS FASES

Memória Estável

Se o coordenador falhar:

- após ter escrito "prepare" no buffer: (após o boot ele poderá continuar de onde parou)
- após ter escrito no buffer o resultado da consulta: (após o boot ele consulta o buffer e informa o resultado aos subordinados)

Se o subordinado falhar:

- antes de ter respondido a mensagem: (coordenador continua enviando mensagens por um determinado tempo)
- após ter enviado mensagem de "pronto": (consulta o buffer e continua de onde parou)

CONTROLE DE CONCORRÊNCIA

Algoritmo para tratar que processos diferentes sendo executados concorrentemente não interfiram uns com os outros.

1- Bloqueio (Travamento)

2- Controle Otimista da Concorrência

3- O Método do Carimbo

BLOQUEIO

Quando um processo precisa ler ou escrever em um arquivo ou em outro objeto, ele primeiro bloqueia o acesso dos demais processos ao arquivo.

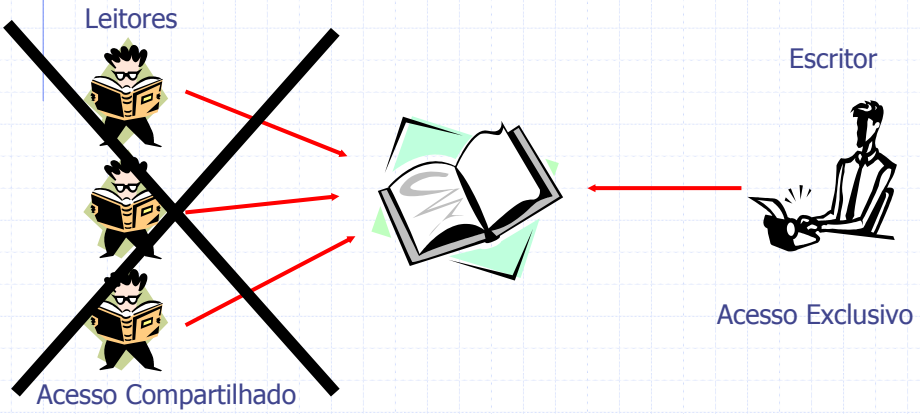
Esse bloqueio é realizado utilizando um único gerenciador de bloqueio podendo ser

- um gerenciador centralizado ou
- um gerenciador local.

Este por sua vez possui uma lista dos arquivos bloqueados e rejeita todas as tentativas de bloquear arquivos já bloqueados.

BLOQUEIO

Pode-se distinguir um bloqueio de leitura de um bloqueio de escrita

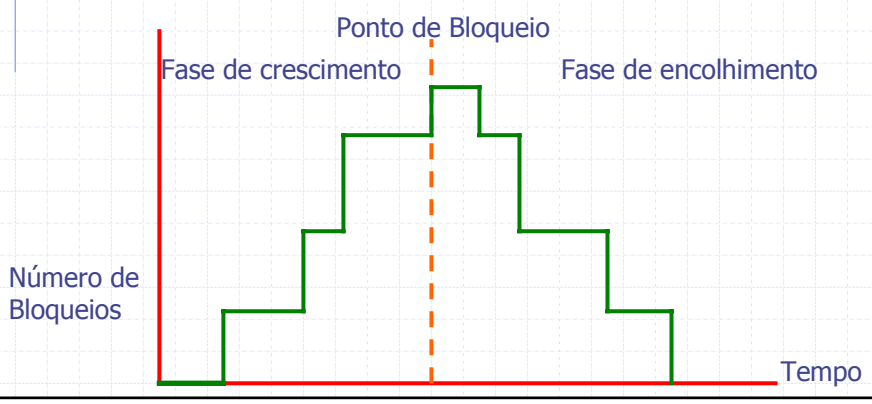


BLOQUEIO

- Granularidade de bloqueio: unidade usada no bloqueio. Ex.: registro, página, bloco etc
  - custo do bloqueio x grau de paralelismo

BLOQUEIO EM 2 FASES

Em primeiro momento o processo adquire todos os bloqueios que ele necessita, que é chamado de **fase de crescimento** e no segundo momento libera os bloqueios durante a **fase de encolhimento.**



BLOQUEIO EM 2 FASES

Em muitos sistemas a fase de encolhimento começa apenas quando a transação tiver terminado o seu processamento, isto é denominado **bloqueio estrito em duas fases.**

Garante a **serialização** dos processos.

Se 2 processos tentarem adquirir o mesmo par de bloqueios, mas na ordem oposta, pode ocorrer um **deadlock**.

* se ocorrer algum problema, libera-se todos os bloqueios, espera-se um tempo e reinicia-se o processo

CONTROLE OTIMISTA DA CONCORRÊNCIA

IDÉIA (Kung e Robinson - 1981): Vá em frente e faça tudo o que você precisa fazer, sem prestar atenção naquilo que os outros estão fazendo. Se houver algum problema preocupe-se com ele mais tarde.

Conflitos não ocorrem com freqüência portanto na maior parte do tempo o algoritmo funciona bem.

Controla-se quais arquivos foram lidos ou escritos. Ao final da transação verifica-se se algum deles foi modificado. Se isso ocorreu aborta-se a transação

Vantagens: Imune ao deadlock e permite um alto grau de paralelismo.

Desvantagem: Caso falhe, a transação é executada novamente.

MÉTODO DO CARIMBO

Cada transação recebe um carimbo (único), quando ela executa um BEGIN_TRANSACTION.

Os arquivos (objetos) são carimbados com um carimbo de leitura e/ou um de escrita a cada transação que os use.

Os carimbos são únicos e ordenados o que significa que as transações são processadas em ordem.

O algoritmo de Lamport garante a ordenação dos carimbos.

Portanto, quando for verificado que uma transação iniciada depois da que está corrente, acessou o arquivo e foi aceita, significa que a ordem não foi respeitada e que a transação corrente está atrasada e portanto deve ser abortada.

Regra: As transações com carimbo de menor valor sempre vem na frente.

USO DE TRANSAÇÕES

VANTAGENS: Técnica muito promissora para a construção de sistema distribuídos confiáveis, pois não há possibilidade de deadlock.

DESVANTAGENS: Imensa complexidade de sua implementação, resultando em uma baixa performance

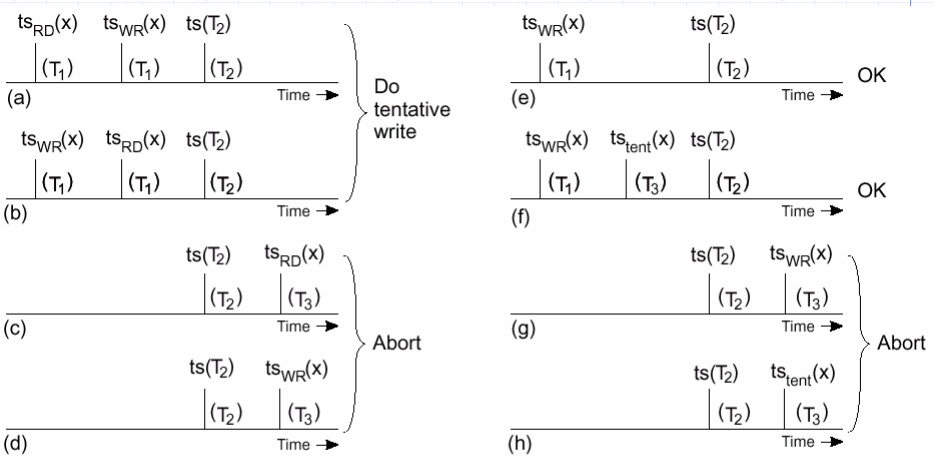


Fig. 5-28. Concurrency control using timestamps.