UNIVERSIDADE FEDERAL DE ITAJUBÁ

Banco de Dados II
COM 231

Recuperação baseada em Log
Aula 11

Vanessa Cristina Oliveira de Souza

Problema

- Um sistema de computação, como qualquer outro dispositivo elétrico ou mecânico, está sujeito a falhas, como:
  - quebra do disco
  - queda da energia elétrica
  - erros de software
  - fogo na sala da máquina
  - sabotagem,
  - etc.
- Essas falhas podem levar a perda de informações no banco de dados ou torná-lo inconsistente
- Como garantir a consistência do banco nessas situações?

Transação

"Transação é uma unidade lógica de trabalho, envolvendo diversas operações de bancos dados."

(C. J. Date - Introdução a Sistemas de Bancos de Dados

Transação

- Está envolvida com os comandos Update, Delete e Insert.

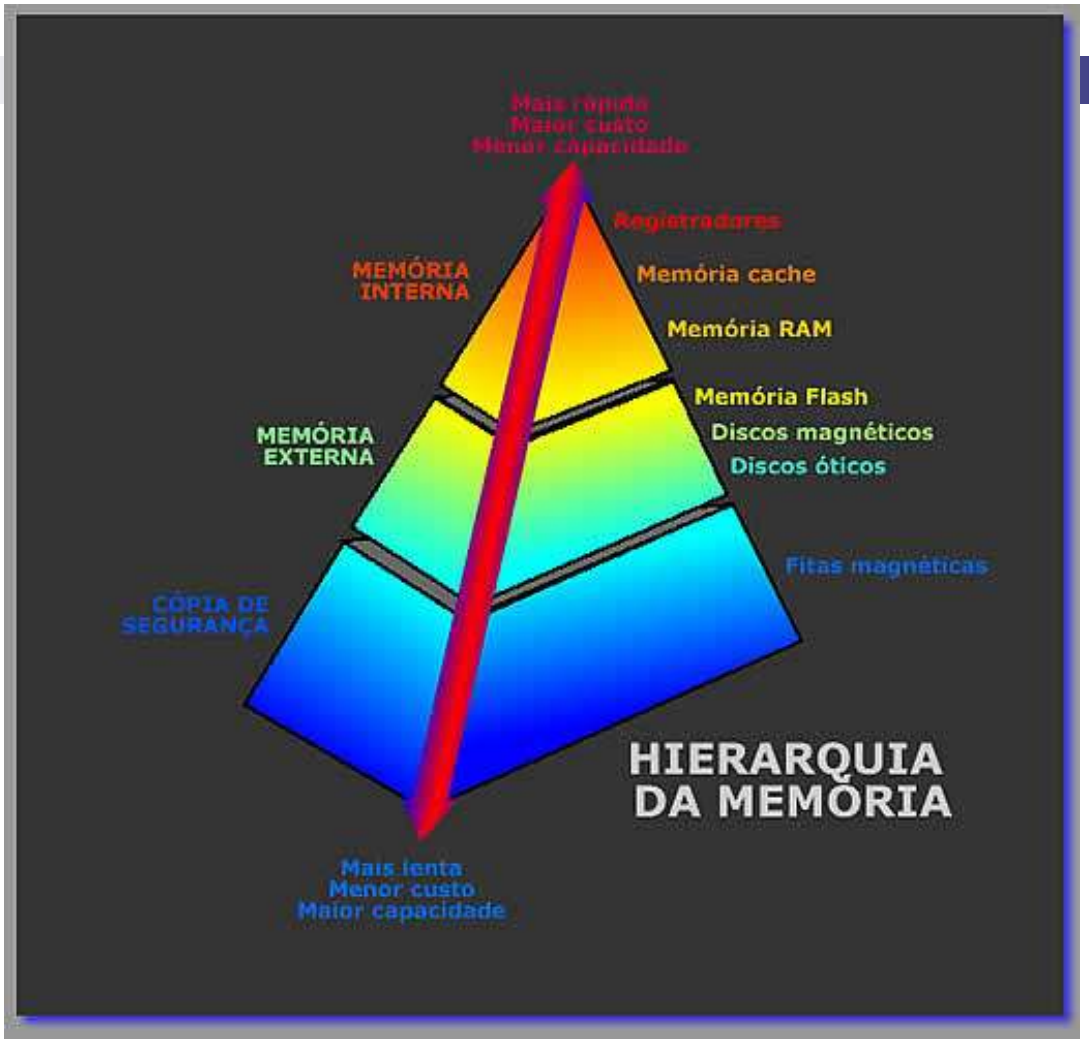
- Do ponto de vista do SGBD, uma transação é uma sequência de operações que são tratadas como um bloco único e indivisível (atômico) no que se refere à sua recuperação.

- Assegura que atualizações inacabadas ou atividades corrompidas não sejam executadas no banco de dados.

Tipos de Armazenamento

- Armazenamento volátil
  - Memória principal e cache
- Armazenamento não-volátil
  - Discos e fitas
- Armazenamento estável
  - Duplicar informações em diversos meios não-voláteis, com modos independentes de falhas, e atualizar a informação de uma maneira controlada.

Hierarquia de Armazenamento



Fonte: http://www.bpiropo.com.br/fpc20070903.htm

Hierarquia de Armazenamento

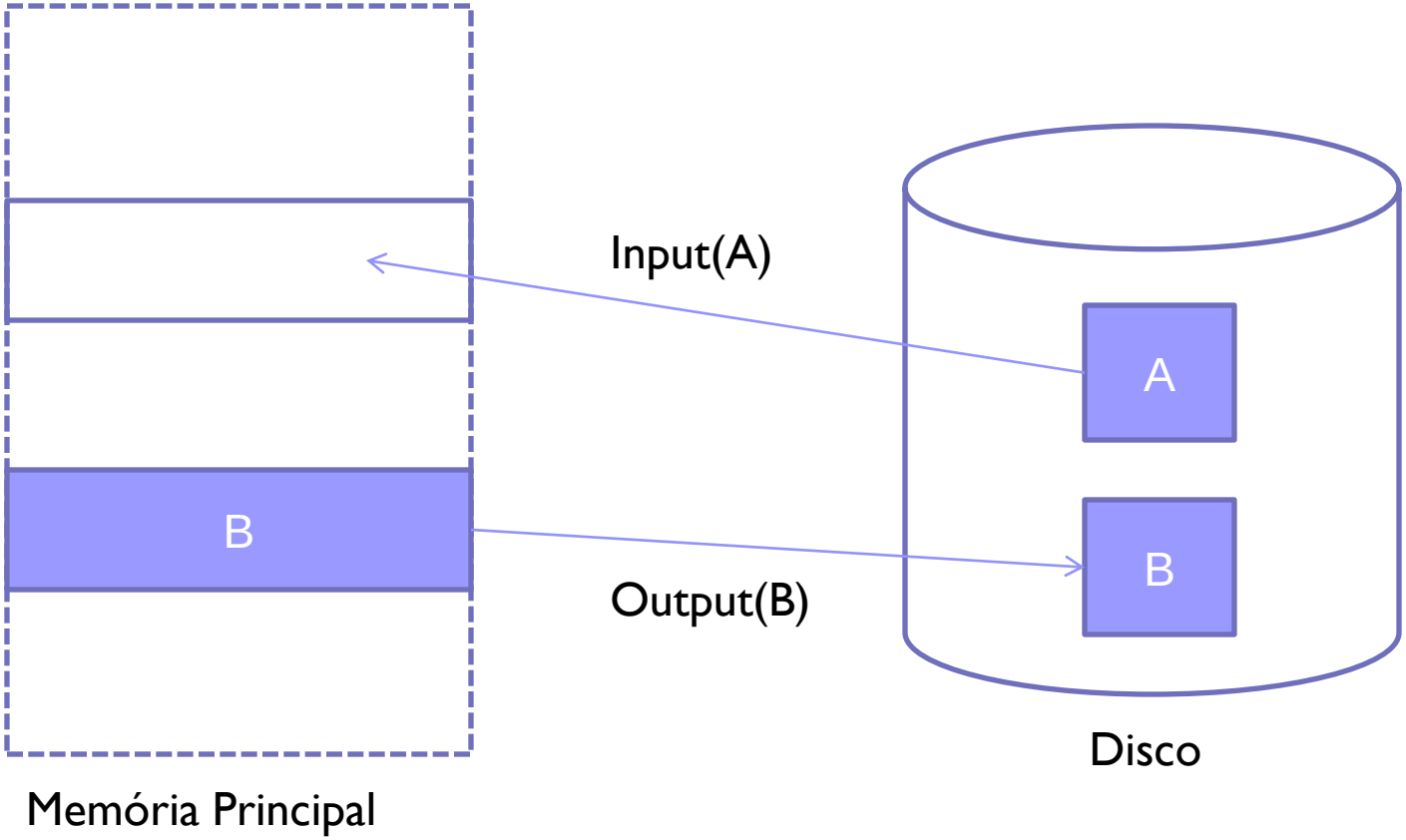
- As transações transferem blocos de informações do disco para a memória principal e depois os devolvem para o disco.

- Os blocos residentes no disco são chamados blocos físicos, e os residentes na memória, de blocos de *buffer*.

Hierarquia de Armazenamento

- Os movimentos de blocos entre o disco e a memória são feitos por duas operações:
  - Input(X)
    - Transfere do disco para a memória o bloco físico X
  - Output(X)
    - Transfere da memória para o disco o bloco de buffer X e substitui o bloco físico apropriado.

Hierarquia de Armazenamento



Hierarquia de Armazenamento

- As transações interagem com o sistema de banco de dados transferindo dados de variáveis de programa para o banco de dados e, do banco de dados para as variáveis de programa.

Hierarquia de Armazenamento

- Esta transferência de dados é realizada usando outras duas operações:
  - Read(X, xi)
    - Atribui o valor de X para a variável local xi

      Se o bloco no qual X reside não estiver na memória, então emite input(X)

      Atribui o valor do bloco de buffer X para xi
  - Write(X,xi)
    - Atribui o valor da variável local xi para o bloco de buffer X

      Se o bloco no qual X reside não estiver na memória, então emite input(X)

      Atribui xi para o valor de X do bloco de buffer

Hierarquia de Armazenamento

- Note que ambas as operações podem requerer uma operação Input, mas não requerem especificamente uma operação output.

- Um bloco de buffer é gravado eventualmente no disco porque o gerenciador de buffer precisa de espaço na memória ou porque o SGBD deseja refletir a mudança de X no disco.

- Uma operação output não precisa ser feita imediatamente depois que write é executado, uma vez que o bloco no qual X reside pode conter outros dados aos quais ainda se está fazendo acesso.

- Se o sistema cair depois da operação write e antes da operação output, o novo valor de X nunca será escrito no banco.

Exemplo

- Considere um banco de dados de uma instituição financeira contendo o saldo da conta corrente de vários clientes, assim como o saldo total dos depósitos de cada agência.

- Suponha que se deseje transferir R$100,00 da conta da Alice, cujo saldo atual é R$1.000,00, para a conta do Bob, cujo saldo atual é de R$800,00.

Exemplo

- Suponha que se deseje transferir R$100,00 da conta da Alice, cujo saldo atual é R$1.000,00, para a conta do Bob, cujo saldo atual é de R$800,00.
  - UPDATE conta SET saldo= saldo - 100 WHERE cliente = ‘Alice’;
  - UPDATE conta SET saldo= saldo + 100 WHERE cliente = ‘Bob’;

Exemplo

- Essa transação (T) pode ser definida como:

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

- A soma de A e B não deve mudar com a execução de uma transação.

Transação ACID

- A integridade de uma transação depende de 4 propriedades, conhecidas como ACID:
  - Atomicidade
  - Consistência
  - Isolamento
  - Durabilidade

Voltando ao Exemplo...

- Suponha que se deseje transferir R$100,00 da conta da Alice, cujo saldo atual é R$1.000,00, para a conta do Bob, cujo saldo atual é de R$800,00.

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)

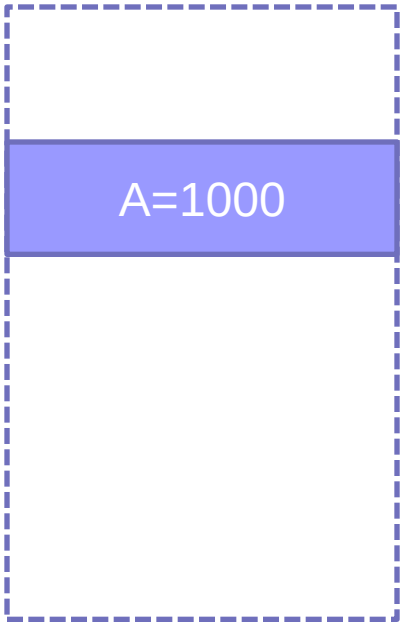
UPDATE conta SET saldo = saldo - 100
WHERE cliente = ‘Alice’

4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

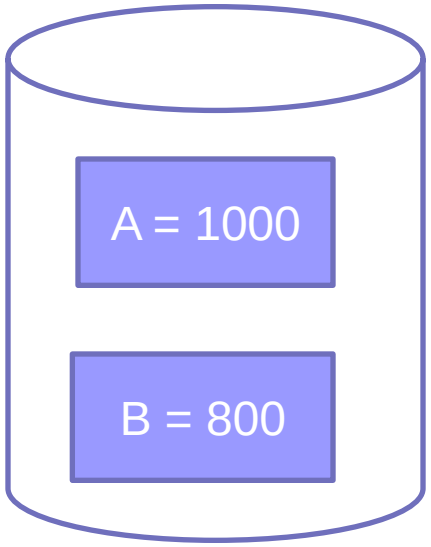
UPDATE conta SET saldo = saldo + 100
WHERE cliente = ‘Bob’

Exemplo – Memória e Disco antes da transação T

- Suponha que a memória principal contenha o bloco de buffer de Alice (A), mas não o de Bob (B).



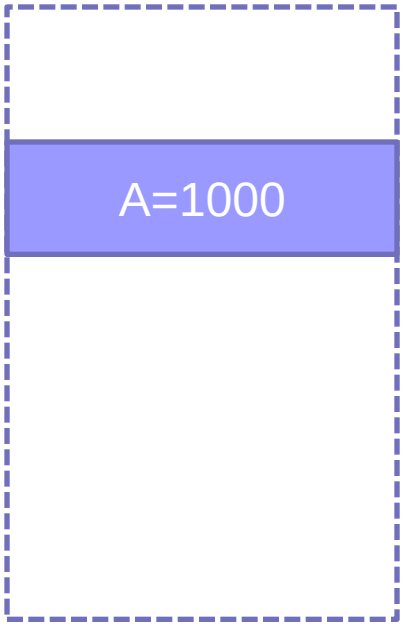
Memória Principal



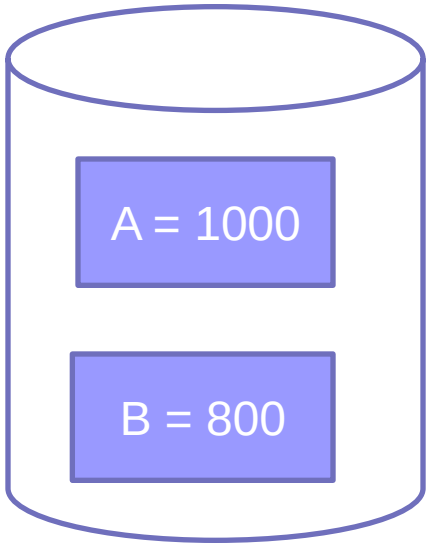
Disco

Exemplo – Memória e Disco antes da transação T

- Início da Transação



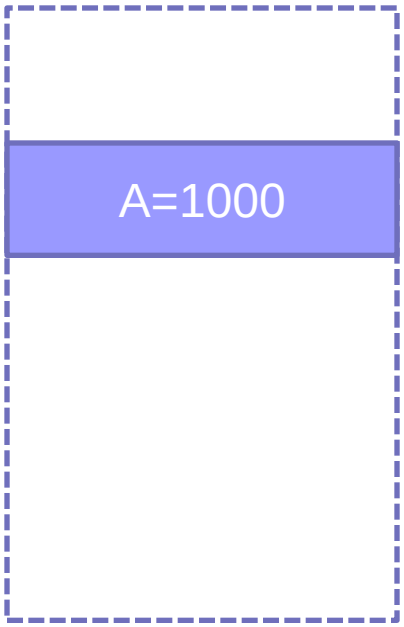
Memória Principal



Disco

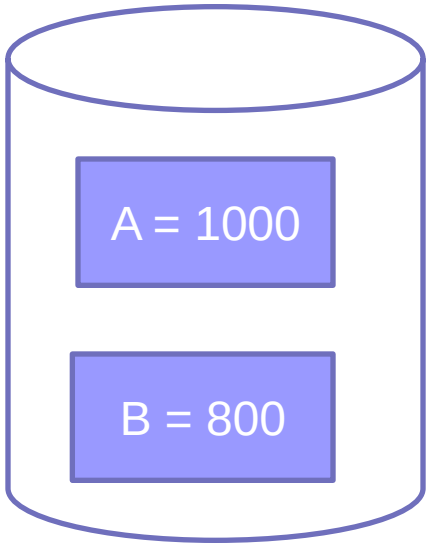
Exemplo – Memória e Disco antes da transação T

- Atualizar Alice



Memória Principal

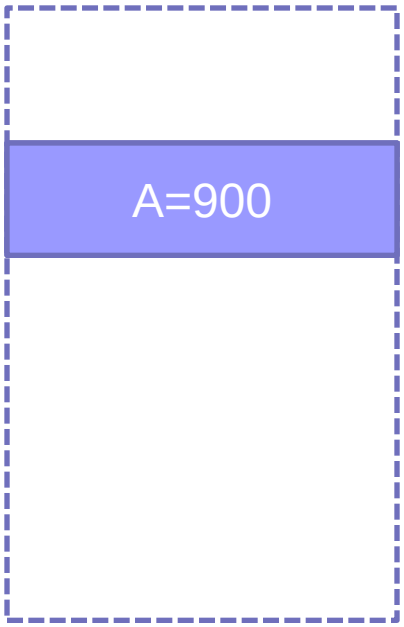
1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)



Disco

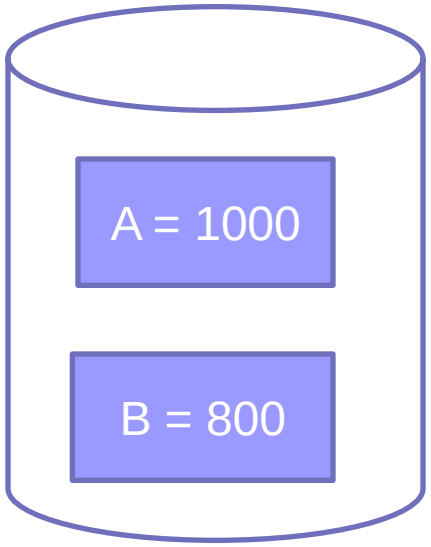
Exemplo – Memória e Disco antes da transação T

- Atualizar Alice



Memória Principal

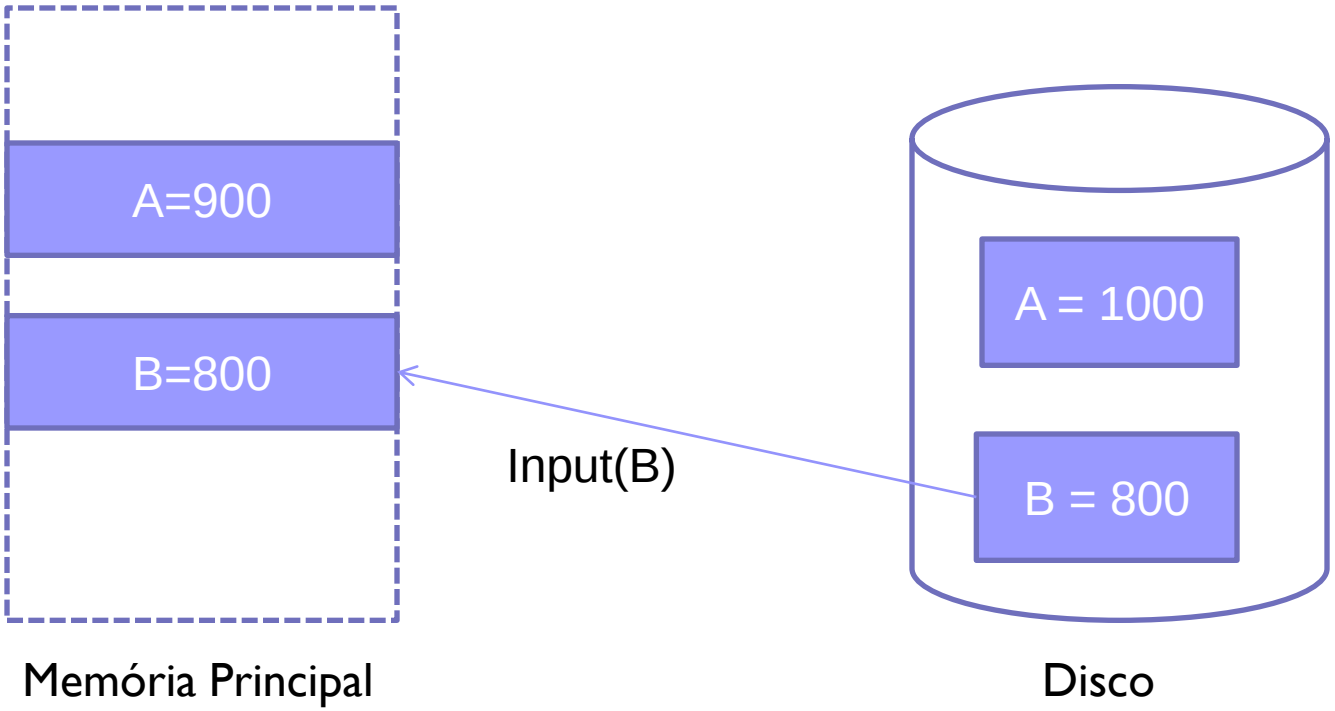
1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)



Disco

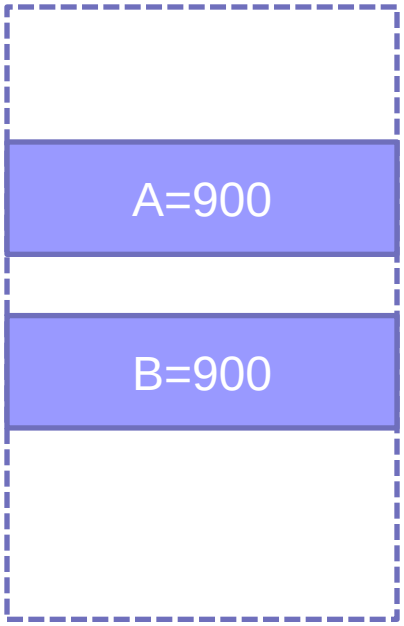
Exemplo

- Memória e disco antes do passo 4 da transação T



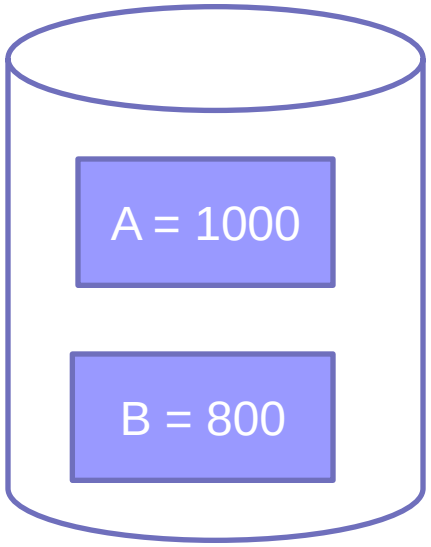
Exemplo

- Atualizar Bob



Memória Principal

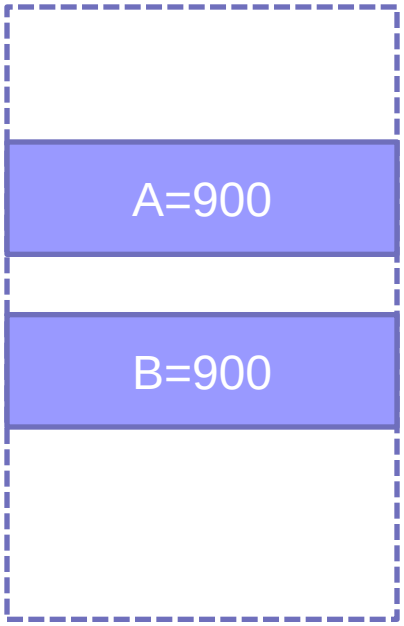
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)



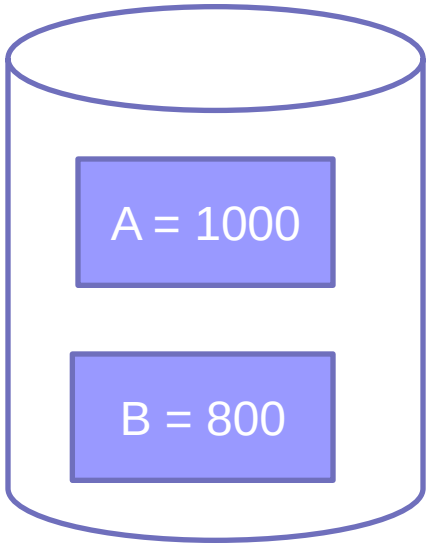
Disco

Exemplo

- Memória e disco depois do passo 6



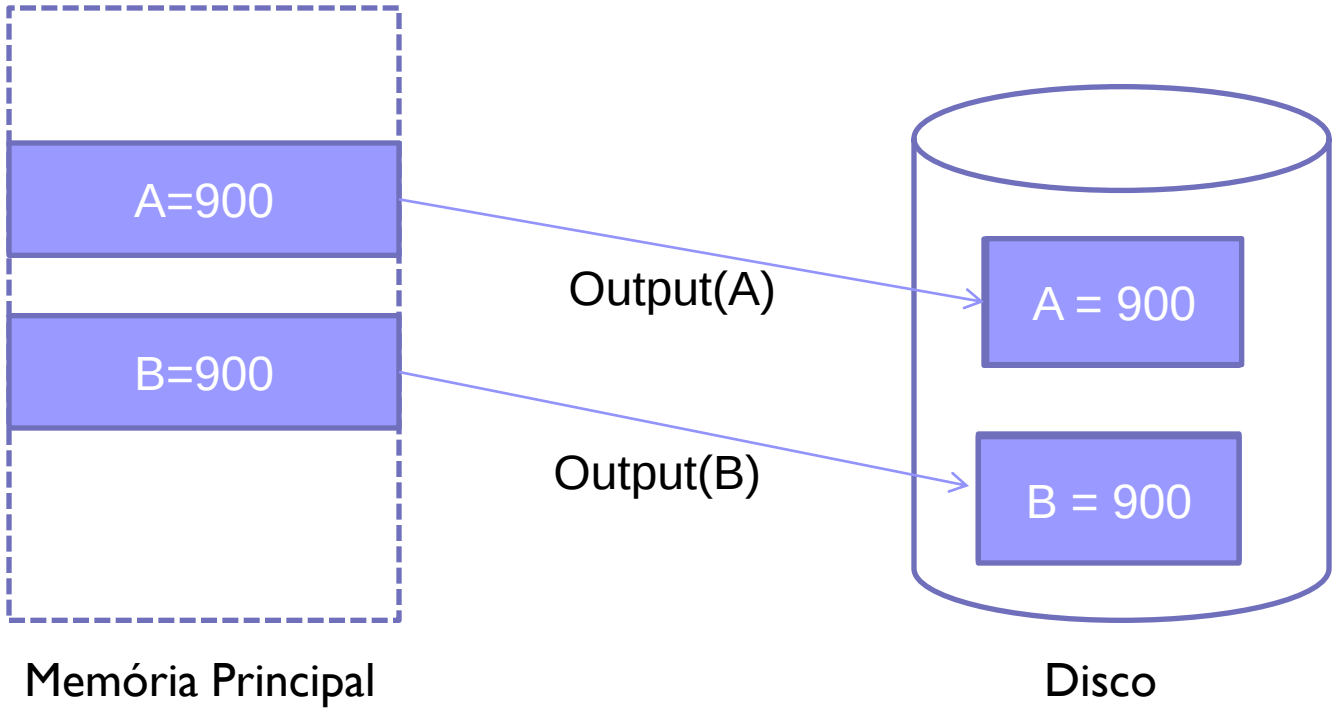
Memória Principal



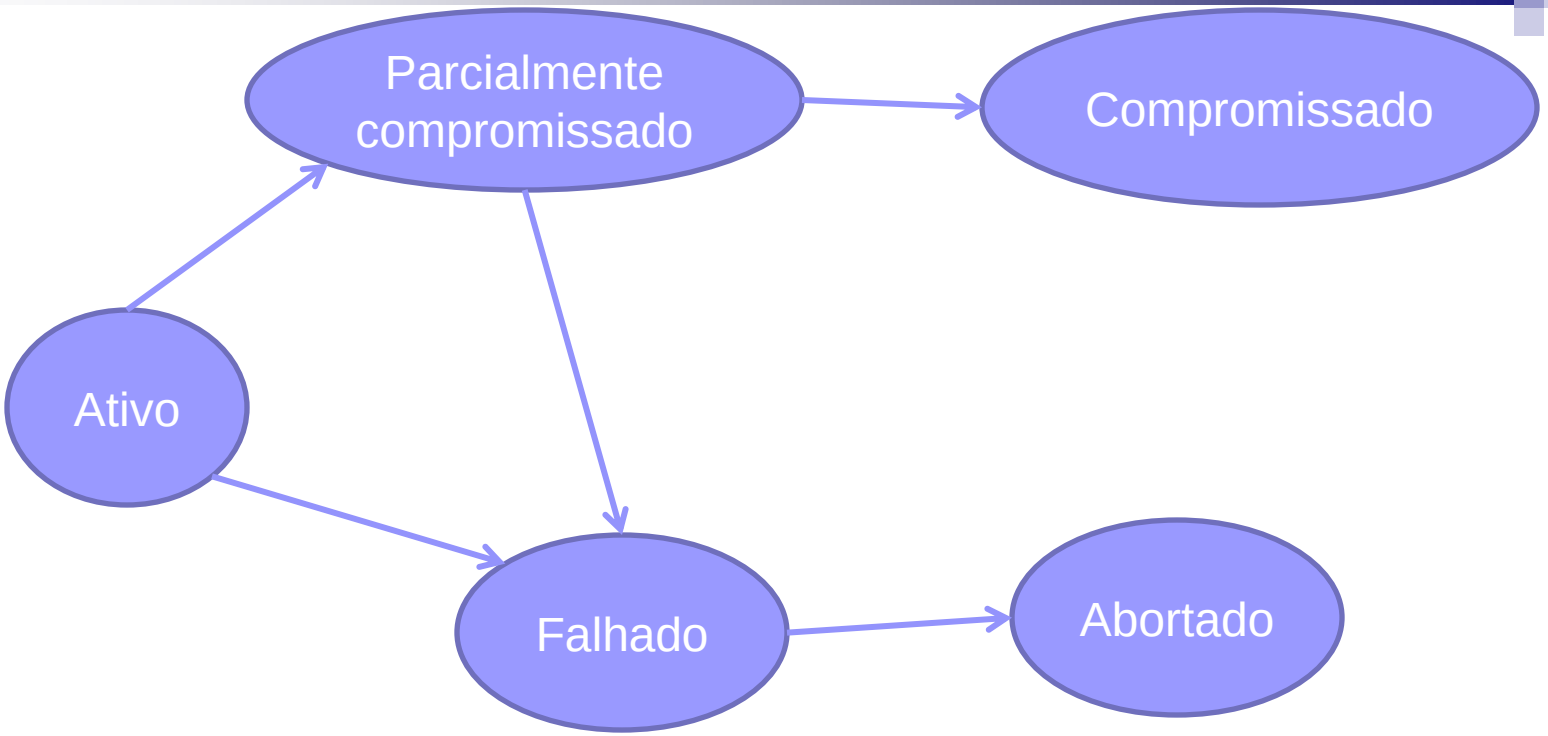
Disco

Exemplo

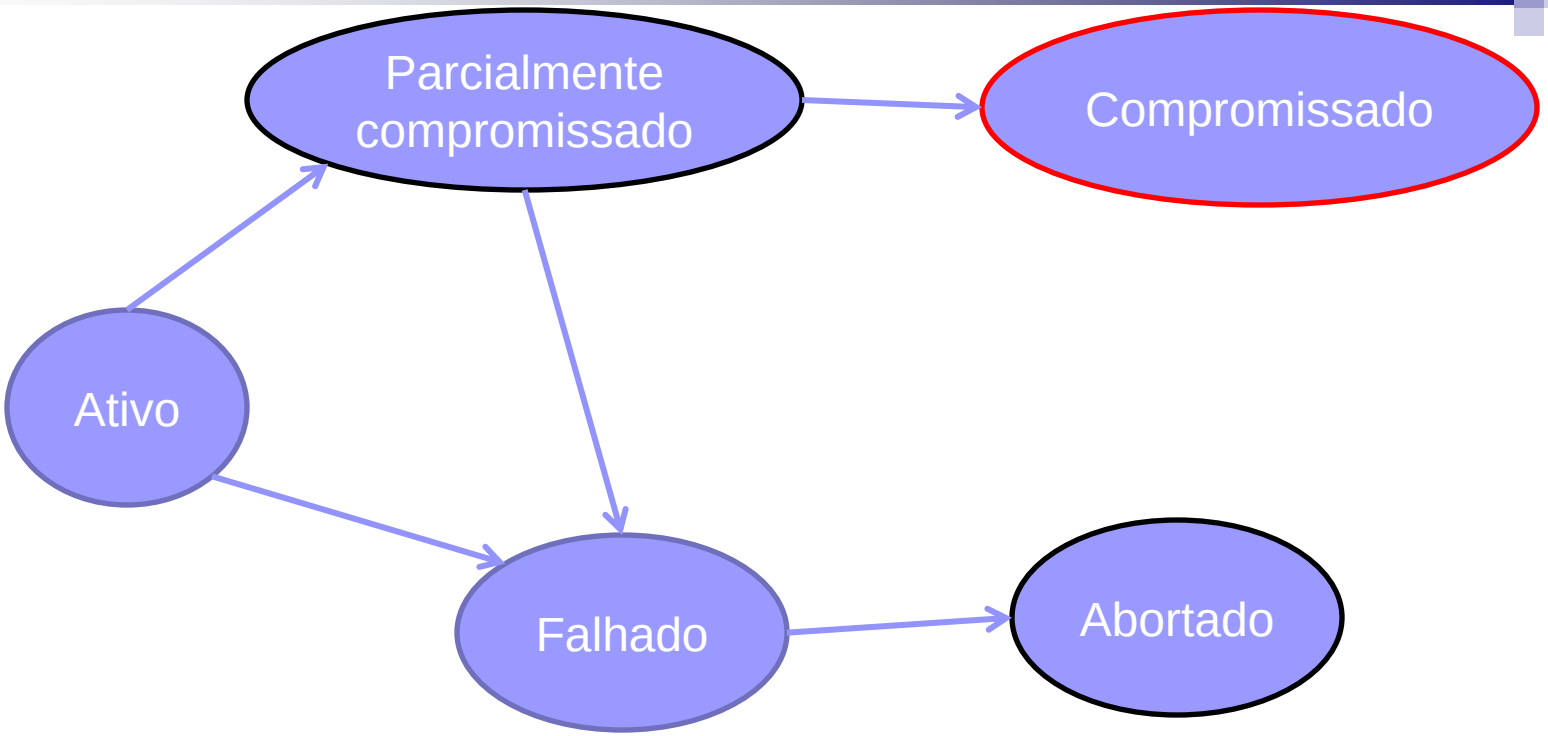
- É preciso validar as alterações no disco



Estados de Transações



Estados de Transações



Estado Compromissado

- ‘Commited’

- Uma transação entra no estado compromissado se já estiver parcialmente compromissada e for garantido que nunca será abortada, garantindo sua atomicidade e durabilidade.

Atomicidade e Durabilidade

- Existem três esquemas diferentes para assegurar a atomicidade e durabilidade de uma transação:
  - Log com modificações adiadas
  - Log com modificações imediatas
  - Paginação com imagem

UNIVERSIDADE FEDERAL DE ITAJUBÁ

RECUPERAÇÃO BASEADA EM LOG

Objetivo do log

- Gravar informações descrevendo as modificações no armazenamento estável sem modificar o banco de dados em si, garantindo a atomicidade e a consistência do banco de dados.

Estrutura do log

- Cada registro do log descreve uma única gravação no banco de dados e tem os seguintes campos:
  - Nome da transação
    - Só aquelas que executaram a operação write
  - Nome do item de dado
    - Atributo que será modificado
  - Valor antigo
  - Novo valor

Log de Transações

- <T_i start>
  - Marca o início de uma transação
- <T_i, X_j, V_{antigo}, V_{novo}>
  - A transação T_i executou uma gravação num item de dado X_j, o qual tinha o valor V_{antigo} antes da gravação e terá o valor V_2 depois.
- <T_i commits>
  - A transação T_i foi compromissada

Exemplo

- Voltando ao exemplo da transferência bancária entre Alice e Bob.

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

LOG

<T start>

<T,A, 1000, 900>

<T, B, 800, 900>

<T commits>

Modificação do banco de dados adiada

- Durante a execução de uma transação, todas as operações write são adiadas até que a transação seja parcialmente compromissada.
- Todas as atualizações são registradas no log, que precisa ser mantido em meio de armazenamento estável.
- Quando uma transação é parcialmente compromissada, a informação do log é usada na execução das gravações no disco.

Modificação do banco de dados adiada – Esquema 1

Transação Ativa

LOG

<T start>

Modificação do banco de dados adiada – Esquema 1

Transação Ativa

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

LOG

<T start>

<T,A, 1000, 900>

<T, B, 800, 900>

Modificação do banco de dados adiada – Esquema 1

Transação Ativa

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

Parcialmente compromissado

LOG

<T start>

<T,A, 1000, 900>

<T, B, 800, 900>

<T commits>

Modificação do banco de dados adiada – Esquema 1

Transação Ativa

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

LOG

<T start>

<T,A, 1000, 900>

<T, B, 800, 900>

<T commits>

Garantia que o Log esteja em armazenamento Estável

Output(A)

Output (B)

Parcialmente compromissado

Compromissado

Modificação do banco de dados adiada – Esquema 2

LOG

Transação Ativa

<T start>

<T,A, 1000, 900>

<T, B, 800, 900>

<T commits>

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

Garantia que o Log esteja em armazenamento Estável

Parcialmente compromissado

Output(A)

Falha!

O quê acontecerá quando o sistema voltar?

Modificação do banco de dados adiada

- Usando o log, o sistema pode manipular qualquer falha que resultar na perda de informações em dispositivo de armazenamento volátil.

- Dada uma falha, o sistema de recuperação consulta o log para determinar quais transações precisam ser refeitas.
  - Apenas aquelas que o log contiver o registro <T_i start> e o registro <T_i commits>

Modificação do banco de dados adiada

- Operação Redo
  - Redo(T_i)
  - Refaz T_i, ajustando o valor de todos os itens de dados atualizados pela transação ti, para os novos valores.
  - É idempotente

Modificação do banco de dados adiada – Esquema 2

Transação Ativa

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

LOG

<T start>

<T,A, 1000, 900>

<T, B, 800, 900>

<T commits>

Redo T

Output(A)

- novoValor = 900

Output(B)

- novoValor = 900

Parcialmente compromissado

Output(A)

Falha!

Modificação do banco de dados adiada – Esquema 3

Transação Ativa

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)

Falha!

LOG

<T start>

<T,A, 1000, 900>

Falhado

O quê acontecerá quando o sistema voltar?

Modificação imediata do banco de dados

- Permite a gravação de modificações no banco de dados enquanto a transação ainda está no estado ativo.

- As modificações gravadas por transações ativas são chamadas modificações descompromissadas (*uncommitted*).

Modificação imediata do banco de dados – Esquema 1

LOG

Transação Ativa

<T start>

<T,A, 1000, 900>

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)

Garantia que o Log esteja em armazenamento Estável

Output(A)

Modificação imediata do banco de dados – Esquema 1

Transação Ativa

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

LOG

<T start>

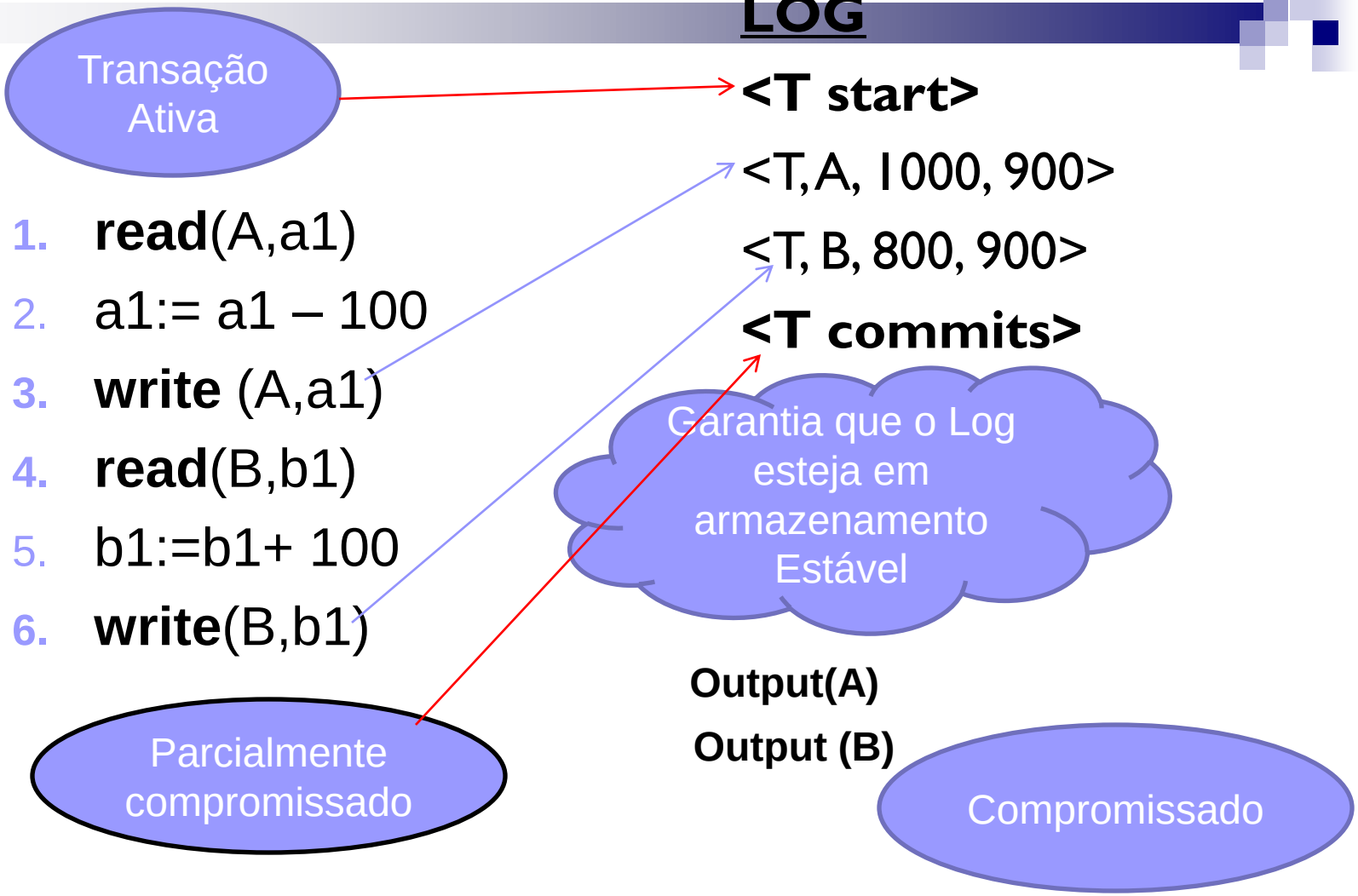
<T,A, 1000, 900>

<T, B, 800, 900>

Garantia que o Log esteja em armazenamento Estável

Output (B)

Modificação imediata do banco de dados – Esquema 1



Modificação imediata do banco de dados – Esquema 1

LOG

Transação Ativa

<T start>

<T,A, 1000, 900>

<T, B, 800, 900>

<T commits>

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

Compromissado

Parcialmente compromissado

Modificação imediata do banco de dados – Esquema 2

Transação Ativa

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

LOG

<T start>

<T,A, 1000, 900>

Garantia que o Log esteja em armazenamento Estável

Output(A)

Modificação imediata do banco de dados – Esquema 2

Transação Ativa

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)

LOG

<T start>

<T,A, 1000, 900>

<T, B, 800, 900>

Garantia que o Log esteja em armazenamento Estável

Falha!

O quê acontecerá quando o sistema voltar?

Modificação imediata do banco de dados – Esquema 3

Transação Ativa

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)

Falha!

LOG

<T start>

<T,A, 1000, 900>

Garantia que o Log esteja em armazenamento Estável

Output(A)

O quê acontecerá quando o sistema voltar?

Modificação imediata do banco de dados

- A transação Ti precisa ser desfeita se o log contiver o registro <T_i start>, mas não o registro <T_i commits>

- Operação Undo
  - undo(T_i)
  - Se ocorrer uma falha, o campo com o valor antigo dos registros do log deve ser usado para restaurar os itens de dados modificados com os valores que tinham antes do início da transação.

Modificação imediata do banco de dados – Esquema 3

Transação Ativa

1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)

Falha!

LOG

<T start>

<T,A, 1000, 900>

Undo T

Output(A)

- **valorAntigo = 1000**

Pontos de Verificação

- Quando uma falha no sistema ocorre, é necessário consultar o log para determinar aquelas transações que precisam ser refeitas e aquelas que precisam ser desfeitas.

- Em princípio, o log inteiro precisa ser varrido:
  - Consumo de tempo
  - A maioria das transações já foram comitadas

- SOLUÇÃO
  - Pontos de Verificação (checkpoints)

Pontos de Verificação

- O sistema mantém o log usando uma das duas técnicas descritas.
- Adicionalmente, o sistema estabelece periodicamente pontos de verificação.
  - <checkpoint>

Exemplo

T_1
read_item(A)
read_item(D)
write_item(D)

T_2
read_item(B)
write_item(B)
read_item(D)
write_item(D)

T_3
read_item(A)
write_item(A)
read_item(C)
write_item(C)

T_4
read_item(B)
write_item(B)
read_item(A)
write_item(A)

<start_transaction, T_1>

<T1, D, 20, 50>

<Commit, T1>

<checkpoint>

<start_transaction, T4>

<T4, B, 15, 19>

<T4, A, 20, 22>

<Commit, T4>

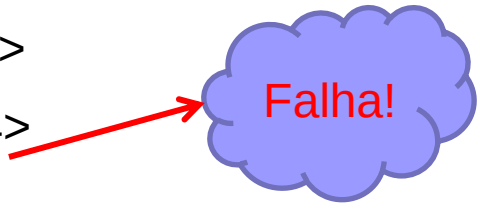
<start_transaction, T2>

<T2, B, 12, 59>

<start_transaction, T3>

<T3, A, 30, 12>

<T2, D, 25, 44>



O quê acontecerá quando o sistema voltar?

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PAGINAÇÃO COM IMAGEM

Paginação com imagem

- Duas tabelas são mantidas durante a existência de uma transação:
  - A tabela corrente
  - A tabela imagem
- Quando a transação se inicia, ambas são idênticas.
- A tabela imagem nunca é modificada durante a transação.
- A corrente pode ser alterada quando uma transação executa uma operação write

Paginação com imagem

- Todas as operações input e output usam a tabela corrente.
- Quando uma transação é parcialmente compromissada, a tabela imagem é descartada e a corrente torna-se a nova tabela no disco.
- Se a transação é abortada, a tabela corrente é simplesmente descartada.