UNIVERSIDADE FEDERAL DE ITAJUBÁ

Banco de Dados II

COM 231

Recuperação baseada em Log

Vanessa Cristina Oliveira de Souza

Problema

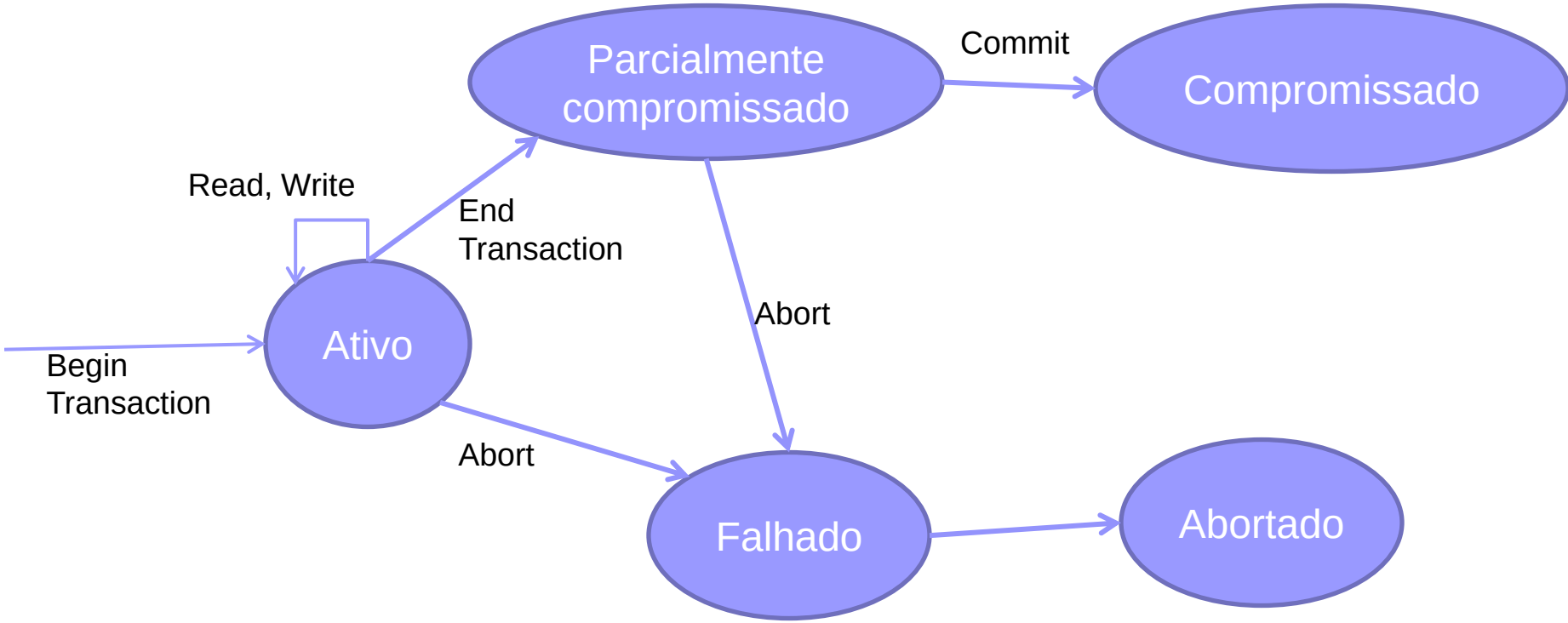
- Um sistema de computação, como qualquer outro dispositivo elétrico ou mecânico, está sujeito a falhas, como:
  - quebra do disco
  - queda da energia elétrica
  - erros de software
  - fogo na sala da máquina
  - sabotagem,
  - etc.
- Essas falhas podem levar a perda de informações no banco de dados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

Transação

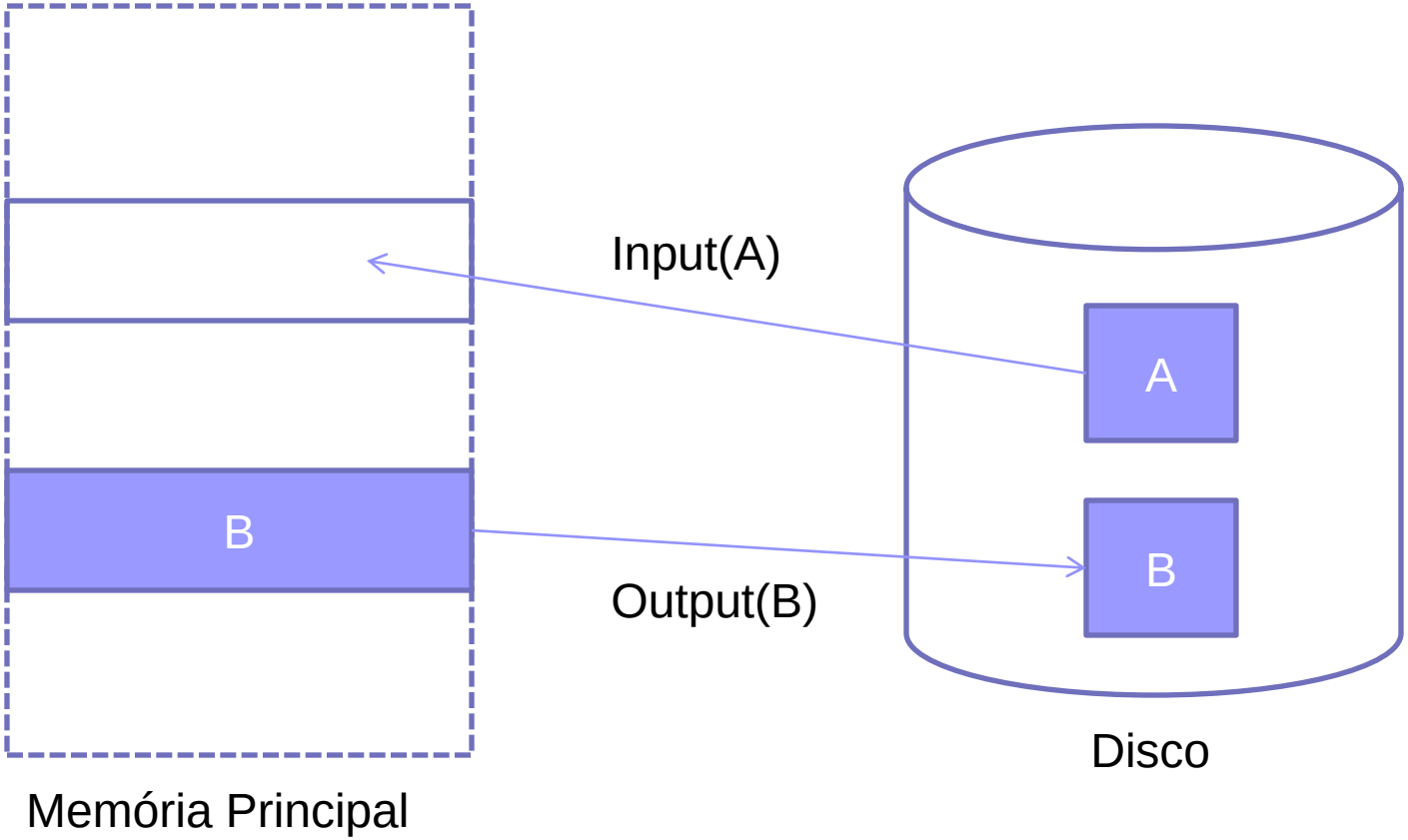
"Transação é uma unidade lógica de trabalho, envolvendo diversas operações de bancos dados."

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

Estados de Transações



Hierarquia de Armazenamento



Operações de Leitura e Gravação

- 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

Operações de Leitura e Gravação

- 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.

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)

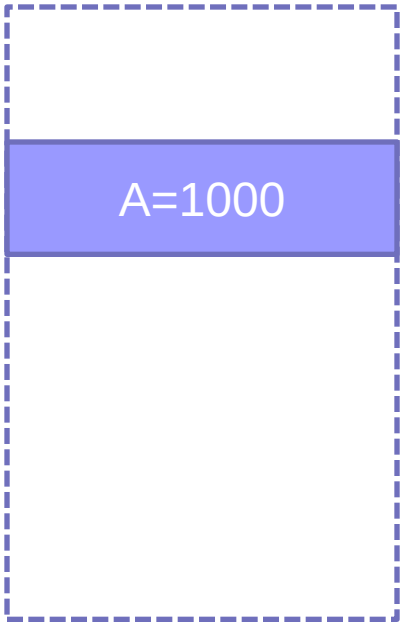
UPDATE conta SET saldo = saldo - 100
WHERE cliente = ‘Alice’

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

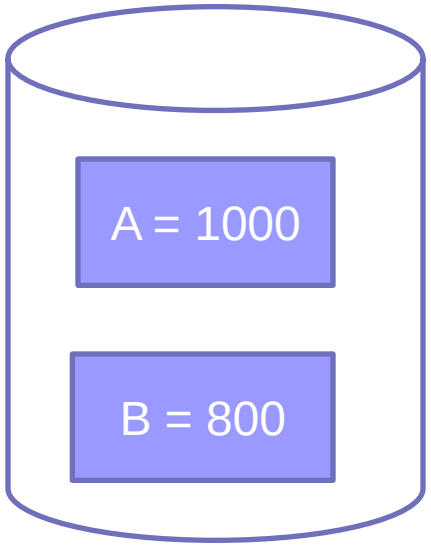
UPDATE conta SET saldo = saldo + 100
WHERE cliente = ‘Bob’

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

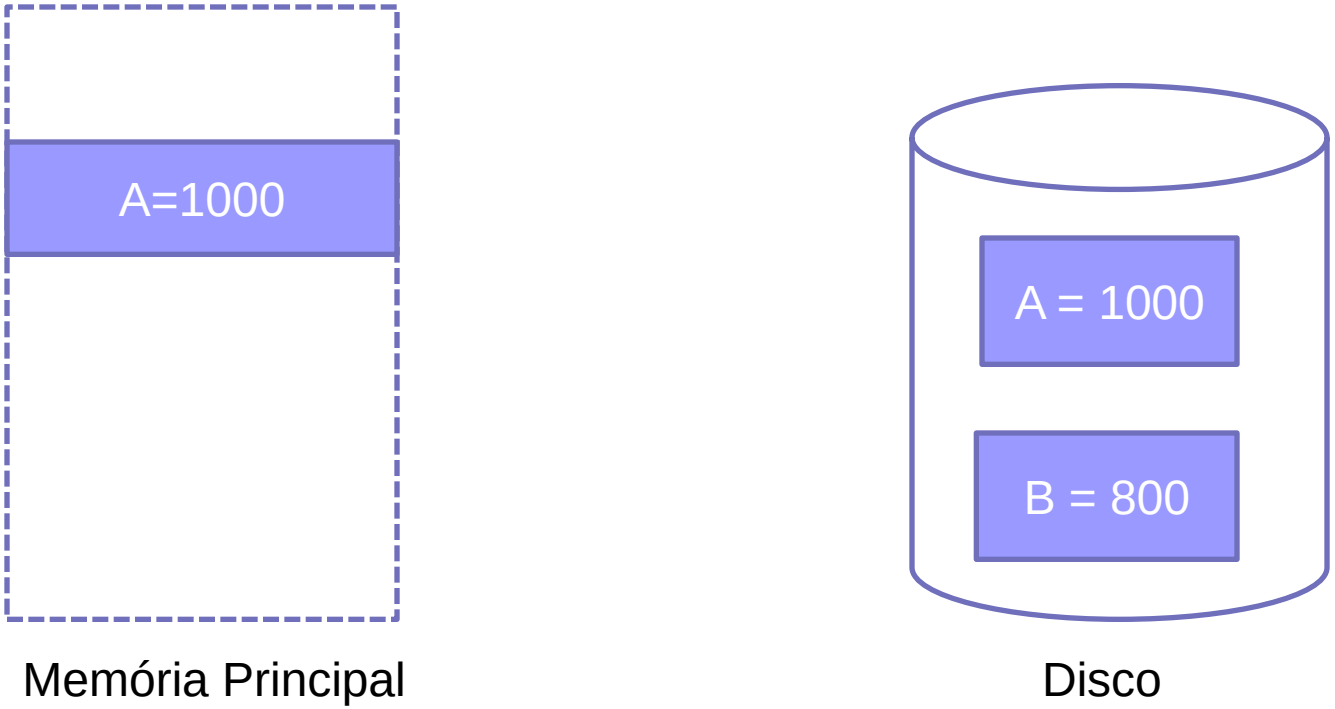


Disco

Exemplo – Memória e Disco antes da transação T

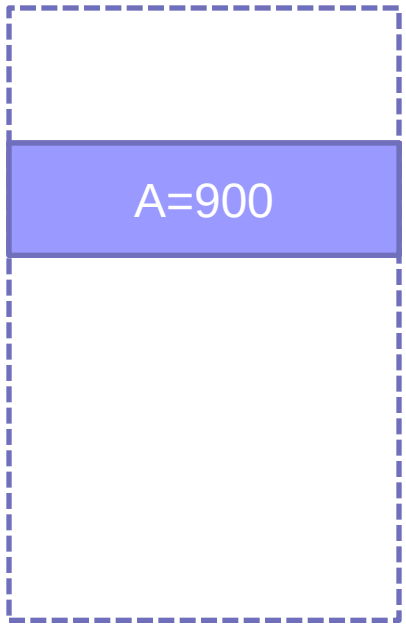
- Início da Transação

Estado da Transação : Ativo



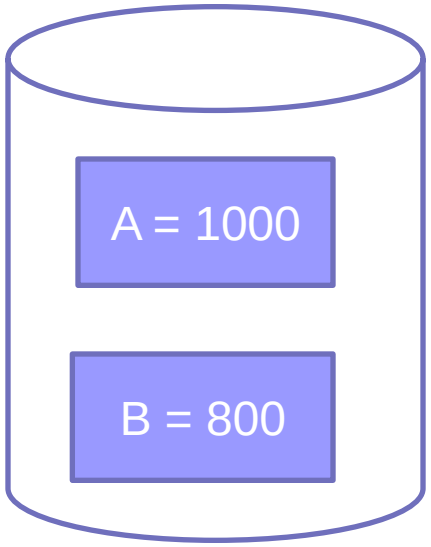
Exemplo – Memória e Disco antes da transação T

- Atualizar Alice



Memória Principal

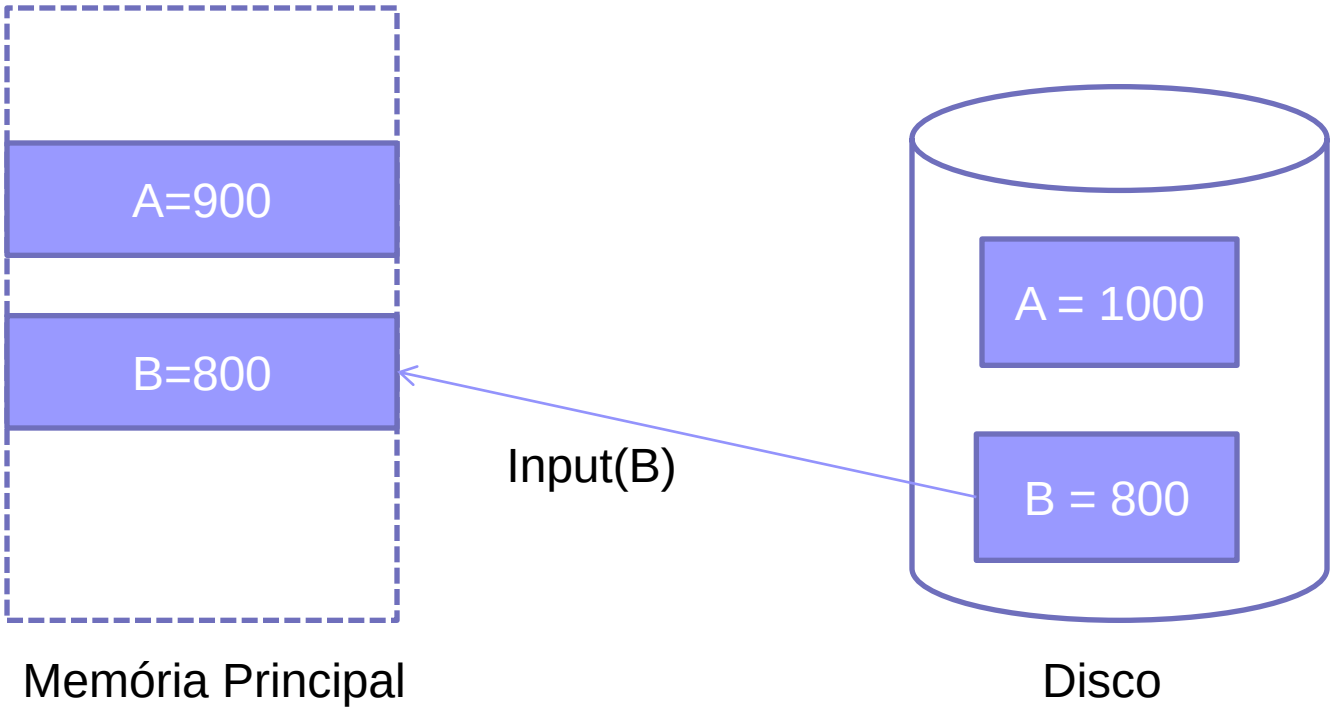
1. **read**(A,a1)
2. a1:= a1 – 100
3. **write** (A,a1)



Disco

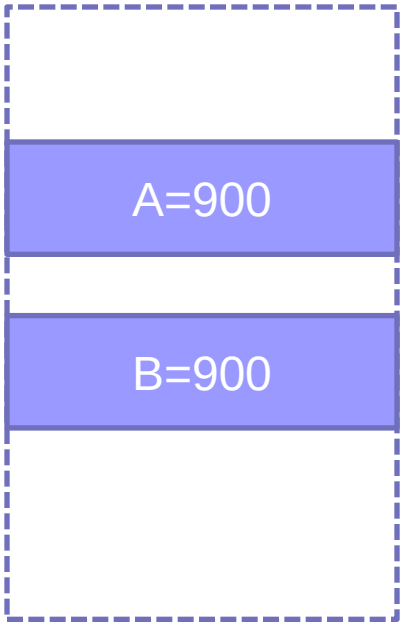
Exemplo

- Memória e disco antes do passo 4 da transação T



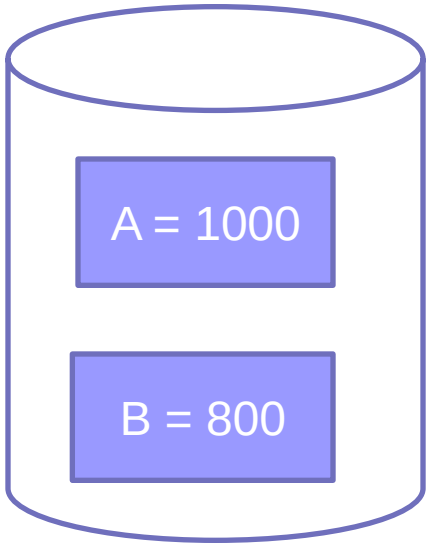
Exemplo

- Atualizar Bob



Memória Principal

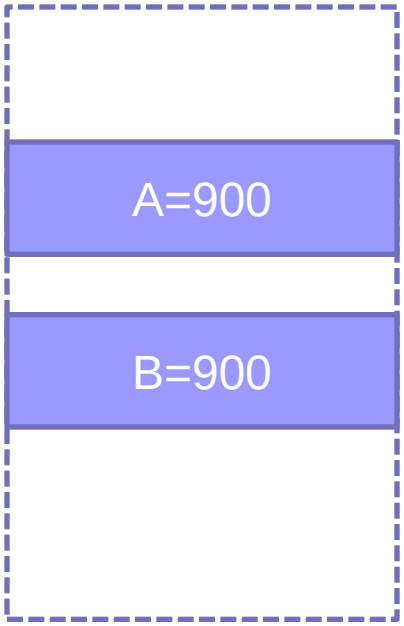
4. **read**(B,b1)
5. b1:=b1+ 100
6. **write**(B,b1)



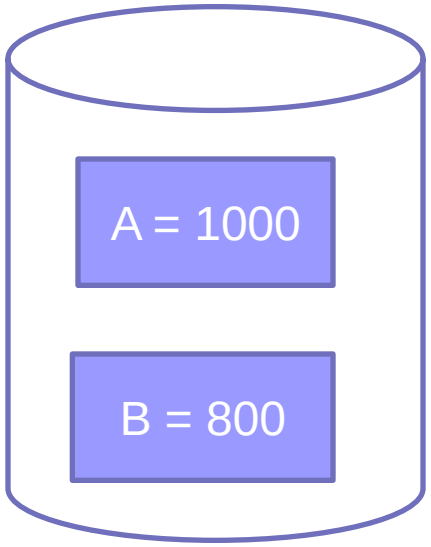
Disco

Exemplo

- Memória e disco depois do passo 6



Memória Principal

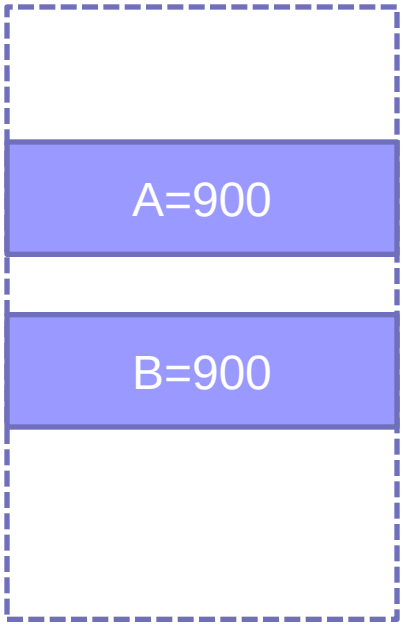


Disco

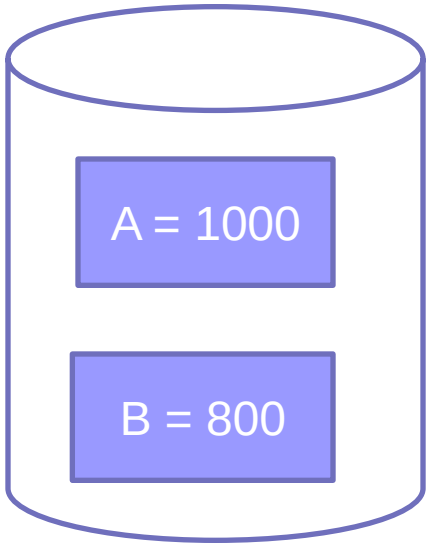
Exemplo

- Memória e disco depois do passo 6

Estado da Transação : Parcialmente Compromissada



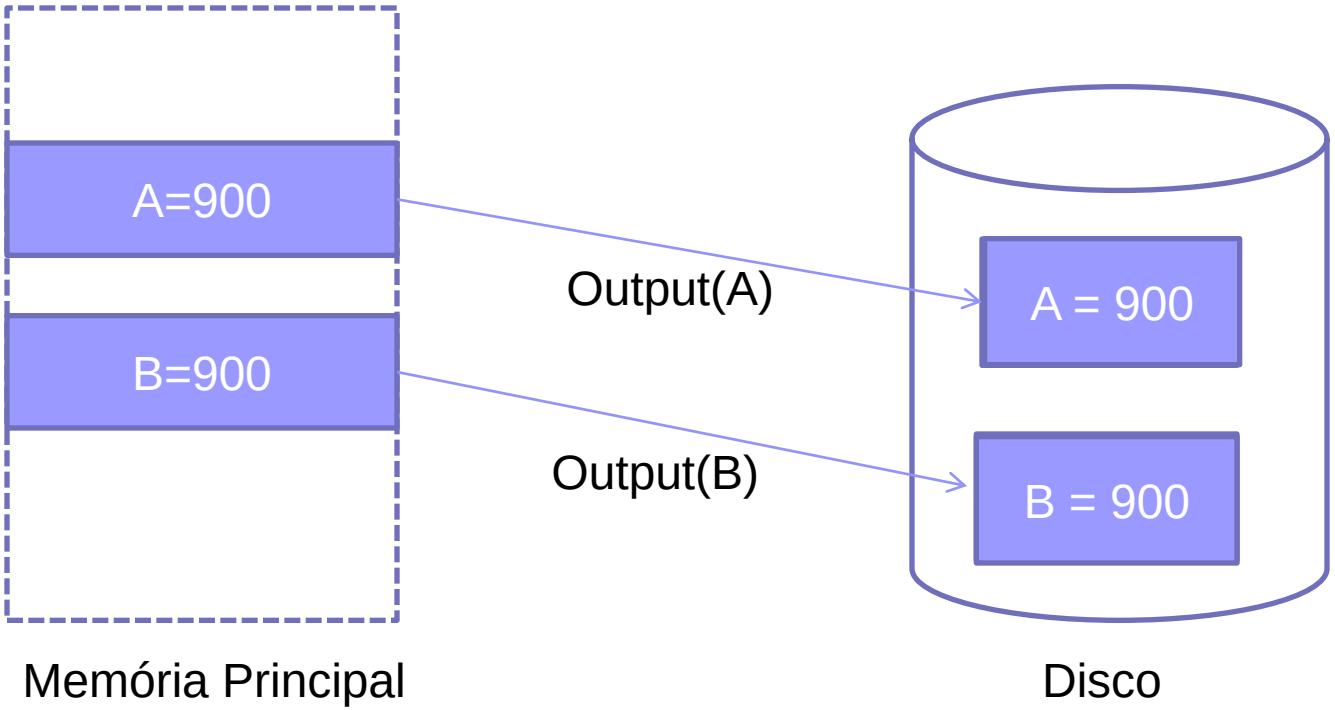
Memória Principal



Disco

Exemplo

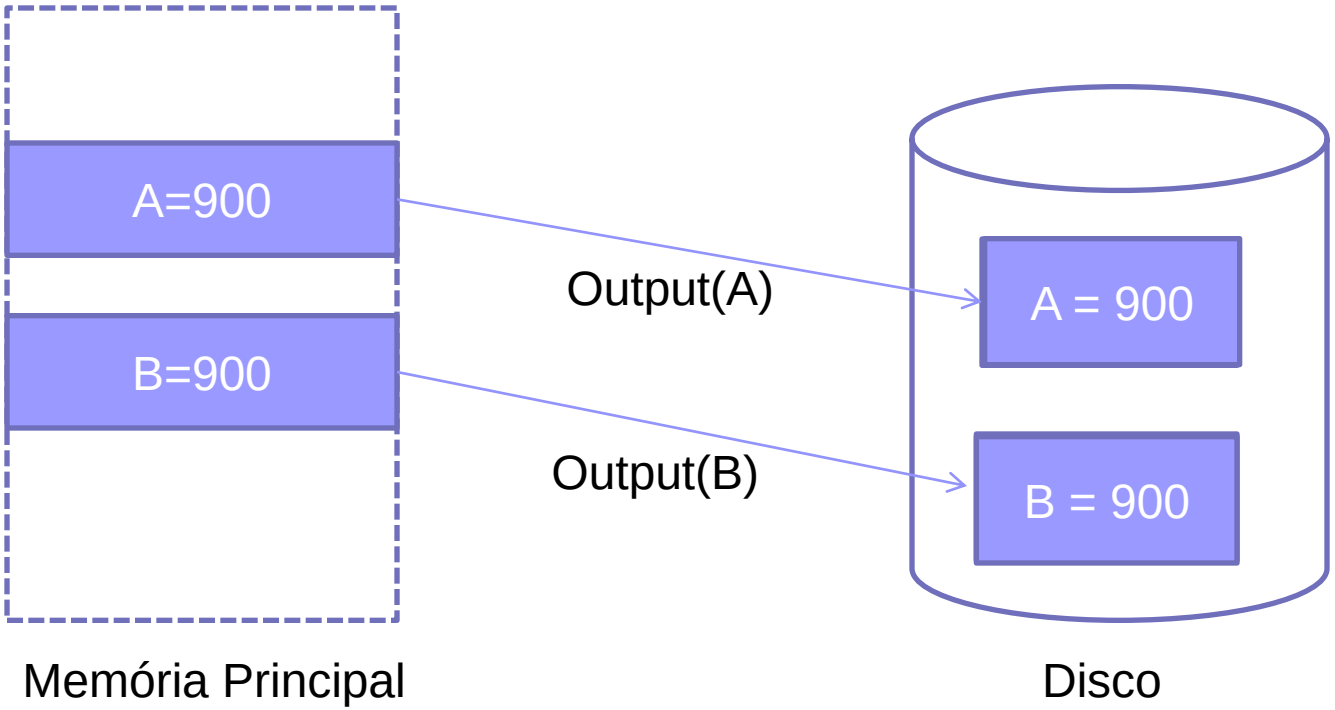
- É preciso validar as alterações no disco



Exemplo

- É preciso validar as alterações no disco

Estado da Transação : Commited



Falhas

- Uma transação entra no estado falhado depois de ser determinado que a transação não pode mais prosseguir com sua execução normal (por exemplo, devido a erros de hardware ou a erros lógicos).

- Como se recuperar de uma falha na transação?
  - **subsistema de recuperação contra falhas** (*recovery*) do SGBD
    - Desfazer as ações de transações parcialmente executadas.

Recuperação de Falhas x Restauração

- O subsistema de recuperação contra falhas (recovery) do SGBD age quando o sistema ‘cai’ e, ao voltar, o banco de dados continua íntegro.
  - Ou seja, falha APENAS no conteúdo de armazenamento volátil

- Se houver danos no armazenamento não volátil será necessário restaurar o banco utilizando um *backup*
  - Tarefa do DBA

UNIVERSIDADE FEDERAL DE ITAJUBÁ

TÉCNICAS DE RECUPERAÇÃO DE BANCO DE DADOS

Recuperação

- A recuperação de falhas de uma **transação** significa que o BD é ***restaurado*** ao estado consistente mais recente antes do momento de falha.

- Para fazer isso o sistema deve manter informações sobre as mudanças que foram aplicadas aos itens de dados pelas diversas transações.

- Em geral, essas informações são mantidas em **arquivos de log**.

Recuperação

- O SGBD utiliza o **log de transações** para rastrear todas as transações que **atualizam** o banco de dados.

- A recuperação de transações de bancos de dados utiliza os dados do **log de transações** para recuperar o banco de dados de um estado inconsistente para um consistente.

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
- <T_i abort>
  - A transação T_i foi abortada

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>

Log de Transações

- O log é um arquivo sequencial, apenas para inserção, que é mantido no **disco**, de modo que não é afetado por qualquer tipo de falha, exceto por falha no disco ou falhas catastróficas.

- Normalmente, um (ou mais) buffers de memória mantêm a última parte do arquivo de log, de modo que as entradas no log são primeiro acrescentadas ao buffer de memória principal.

- Além disso, o arquivo de log do disco é periodicamente copiado para arquivamento em memória estável.

Recuperação

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

Recuperação

- **Steal :** uma página do cache atualizada pode ser gravada em disco ANTES do COMMIT da transação.
  - **No Steal**

- **Force :** todas as páginas atualizadas por uma transação são IMEDIATAMENTE gravadas em disco quando a transação atinge seu ponto de confirmação.
  - **No Force**

Recuperação

- Existem três esquemas diferentes para assegurar a atomicidade e durabilidade de uma transação:
  - Log com modificações adiadas
    - No Steal/Force
  - Log com modificações imediatas
    - Steal/Force
    - Steal/No Force
  - Paginação com imagem

UNIVERSIDADE FEDERAL DE ITAJUBÁ

RECUPERAÇÃO BASEADA EM LOG

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

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>

Garantia que o Log esteja em armazenamento Estável

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

Transação Ativa

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

LOG

<T start>

<T, A, 1000, 900>

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

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(A)

Output (B)

Parcialmente compromissado

Compromissado

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)

Parcialmente compromissado

LOG

<T start>

<T, A, 1000, 900>

<T, B, 800, 900>

<T commits>

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 – 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**

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

- O algoritmo visto anteriormente GARANTE que as páginas do disco são escritas antes do final da transação.
  - Algoritmo UNDO/NO REDO
- Essa estratégia acaba fazendo muitos acessos ao disco.
- Existem estratégias de recuperação que otimizam os acessos ao disco
  - Algoritmo UNDO/REDO

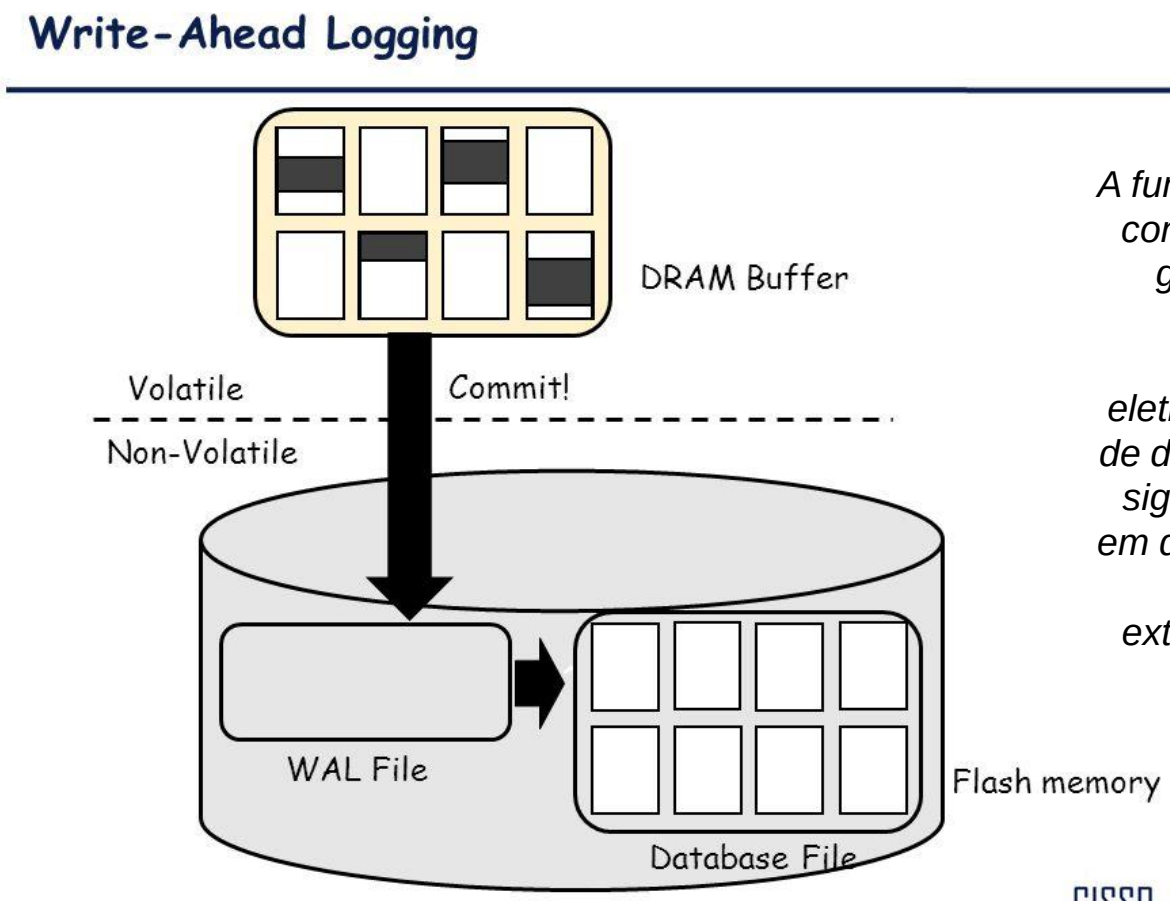
UNIVERSIDADE FEDERAL DE ITAJUBÁ

LOGGING WRITE-AHEAD WAL

Logging Write-Ahead – WAL

- Esta estratégia garante que no final da transação, todas as modificações tenham sido escritas no **arquivo de log**, mas não necessariamente no **disco**.
  - Performance!
  - https://www.postgresql.org/docs/9.1/static/wal-intro.html
  - https://www.postgresql.org/docs/9.4/static/wal-internals.html

Logging Write-Ahead – WAL



A função da área de WAL é garantir a consistência e segurança quanto à gravação dos dados em disco, principalmente em situações adversas, como queda de eletricidade, e propiciar um aumento de desempenho por meio da redução significativa do número de escritas em disco, já que esta representa uma operação de alto custo e extremamente onerosa ao sistema operacional.

Logging Write-Ahead – WAL

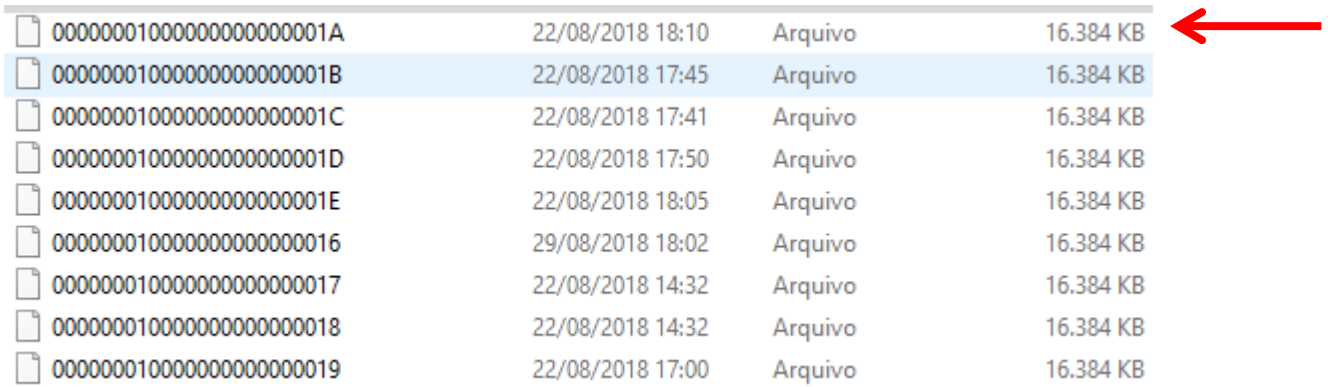
- A maioria dos SGBDs atuais utilizam a estratégia Steal/No Force
- E, para isso, precisam de um suporte maior e mais complexo dos algoritmos de recuperação e dos arquivos de log
  - Lista de transações ativas
  - Lista de redo
  - Lista de undo
- Ideia: utilizar um backup do sistema de arquivos junto com o backup do WAL
  - Backup incremental

Logging Write-Ahead – WAL

- No caso de perdas no armazenamento não volátil, o sistema pode usar o backup juntamente com o WAL para restaurar o banco ao seu último estado consistente.
  - habilitar o arquivamento de log das transações (WAL);
  - realizar a cópia de segurança com identificador de base inicial;
  - e armazenar os arquivos de log das transações realizadas (WAL) após o backup inicial.

Modificação imediata do banco de dados

- Pasta pg_wal



00000001000000000000001A	22/08/2018 18:10	Arquivo	16.384 KB
00000001000000000000001B	22/08/2018 17:45	Arquivo	16.384 KB
00000001000000000000001C	22/08/2018 17:41	Arquivo	16.384 KB
00000001000000000000001D	22/08/2018 17:50	Arquivo	16.384 KB
00000001000000000000001E	22/08/2018 18:05	Arquivo	16.384 KB
000000010000000000000016	29/08/2018 18:02	Arquivo	16.384 KB
000000010000000000000017	22/08/2018 14:32	Arquivo	16.384 KB
000000010000000000000018	22/08/2018 14:32	Arquivo	16.384 KB
000000010000000000000019	22/08/2018 17:00	Arquivo	16.384 KB

UNIVERSIDADE FEDERAL DE ITAJUBÁ

CHECKPOINTS

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, T1>

<T1, D, 20, 50>

<start_transaction, T4>

<T4, B, 15, 19>

<CHECKPOINT>

<T4, A, 20, 22>

<Commit, T4>

<start_transaction, T2>

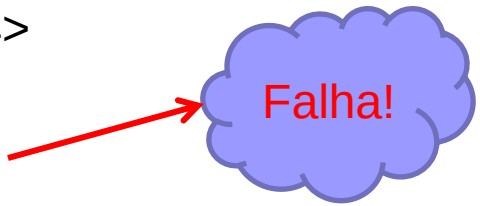
<T2, B, 12, 59>

<start_transaction, T3>

<T3, A, 30, 12>

<T2, D, 25, 44>

O quê acontecerá quando o sistema voltar?



Exemplo

LSN	TRX NUM	PTR ANT	PTR SEG	OPERAÇÃO	TABELA	ID DE LINHA	ATRIBUTO	VALOR ANTES	VALOR DEPOIS
341	101	Nulo	352	START	***inicio transação				
352	101	341	363	UPDATE	PRODUCT	54778	PROD_QOH	45	43
363	101	352	365	UPDATE	CUSTOMER	10011	CUST_BAL	615,73	675,62
365	101	363	Nulo	COMMIT	***fim transação				
397	106	Nulo	405	START	***inicio transação				
405	106	397	415	INSERT	FATURA	1009			1009,10016,...
415	106	405	419	INSERT	LINE	1009			1009,1, 89, DS
419	106	415	427	UPDATE	PRODUCT	89	PROD_QOH	12	11
423	CHECKPOINT								
427	106	419	431	UPDATE	CUSTOMER	10016	CUST_BAL	0,00	277,55
457	106	427	Nulo	COMMIT	***fim transação				
521	155	Nulo	525	START	***inicio transação				
525	155	521	528	UPDATE	PRODUCT	2232	PROD_QOH	6	26
528	155	525	Nulo	COMMIT	***fim transação				
*****FALHA********									

FONTE : Rob & Coronel

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.

Para Casa

- Livro : Silberschatz, Korth & Sudarshan
  - 6ª Edição
  - Editora Campus

- Estudar o Capítulo 16 – Sistema de Recuperação