Banco de Dados

Material de Apoio

Parte XII – Gerenciamento de Transações

Prof. José Osvano da Silva, PMP, PSM I
joseosvano@unipac.br

1º sem / 2022

Sumário

- Conceitos iniciais: Transação
- Propriedades de uma transação
- Estados de uma transação
- Execução concorrente
- Subsistemas de trasações em um SGBD
- Referências

Conceitos iniciais: Transação

- É uma coleção de operações que executa uma função lógica única numa aplicação de banco de dados.

- Cada transação é uma unidade atômica

- Exemplos
  - Processar venda
  - Fazer transferência entre contas
  - Fechar balanço mensal

Propriedades de uma transação

- Para preservar a integridade dos dados, o SGBD tem de assegurar as seguintes propriedades:

1. **Atomicidade**
2. **Consistência / Seriabilidade**
3. **Isolamento**
4. **Durabilidade**

1. Atomicidade

- Todas as operações da transação são refletidas corretamente no banco de dados ou nenhuma delas
- Exemplos:
  - Processar venda
  - Totalizar e fechar venda
  - Baixar em estoque
  - Gerar conta a receber

2. Consistência/Seriabilidade

- A execução de uma transação isolada preserva a consistência do banco de dados, – ou seja, se o banco de dados estiver consistente antes da transação iniciar a sua execução, – deverá permanecer em um estado consistente após a sua execução.
- Exemplos de consistência a ser mantida:
  - Venda processada sem baixa em estoque
  - Retirada de uma conta sem processamento em outra conta

3. Isolamento

- Embora várias transações possam ser executadas ao mesmo tempo, – o sistema garante que não haverá anomalias no banco de dados, em função desta execução concorrente.

- Assim, cada transação não tem ciência da execução das outras no sistema.

4. Durabilidade

- Após uma transação ser completada com sucesso, as mudanças que ela fez no banco de dados persistem, mesmo que existam falhas no sistema.

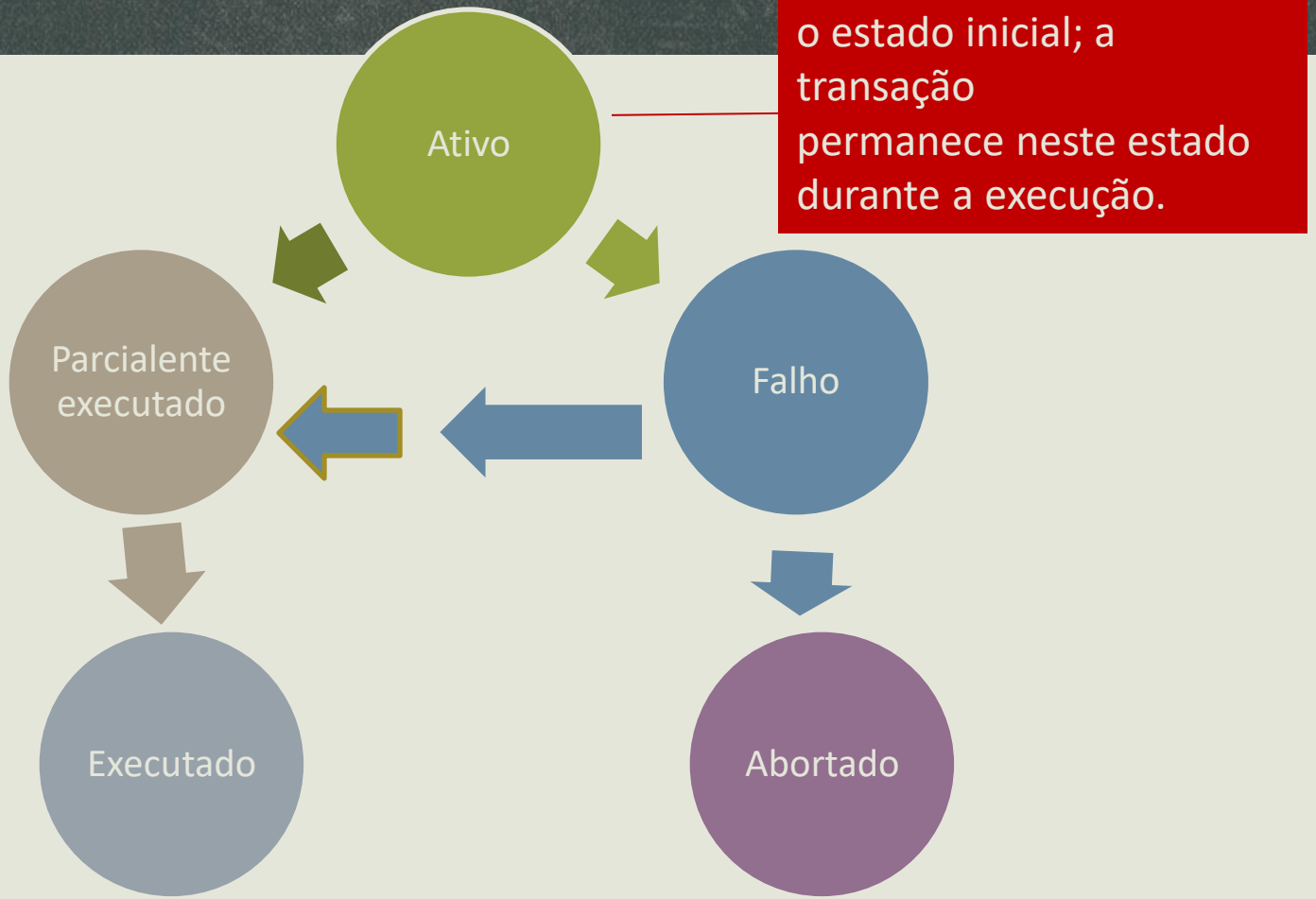
Exemplo: transação para transferir $50 da conta *A para* conta *B*

1. **read**(*A*)
2. *A* := *A* – 50
3. **write**(*A*)
4. **read**(*B*)
5. *B* := *B* + 50
6. **write**(*B*)

- Requisito da **Consistência** – a soma de *A e B não* pode mudar pela execução da transação.
- Requisito da **Atomicidade** — se a transação falha após o passo 3 ou antes do passo 6, o sistema deve assegurar que as atualizações não refletem no DB, senão, o DB ficará inconsistente.
- Requisito da **Durabilidade** — uma vez que o utilizador foi notificado que a transação terminou (i.e., a transferência de $50 teve lugar), as atualizações do DB feitas pela transação devem persistir apesar das falhas.
- Requisito do **Isolamento** — se entre os passos 3 e 6, outra transação é permitido o acesso a um DB parcialmente atualizado, ela encontrará um DB inconsistente (a soma *A + B será menor que o que* deve ser). Pode ser assegurado por correr transações em ***série, uma após a outra. Contudo,*** executar várias transações concorrentemente tem benefícios significativos.

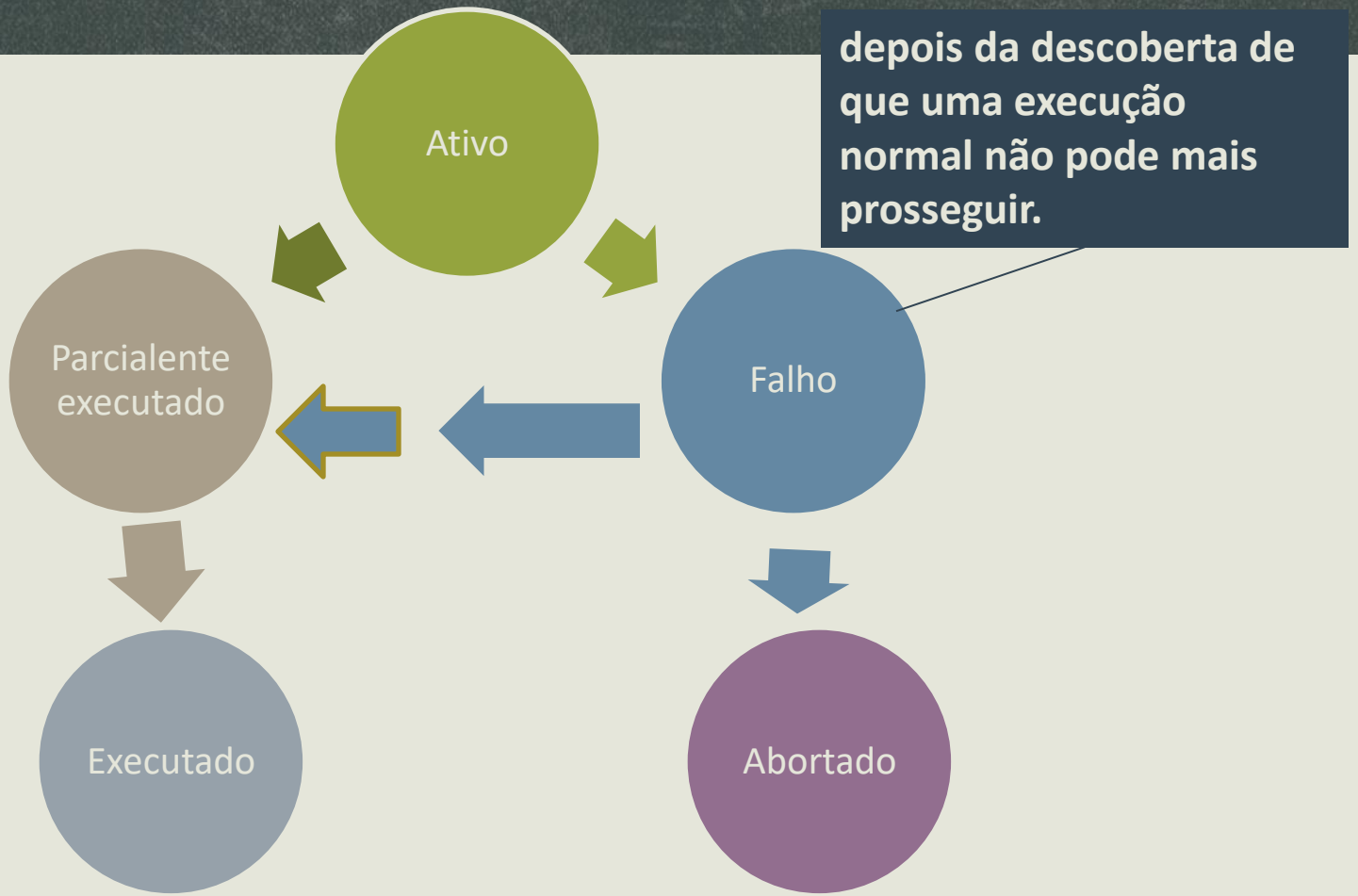
Estados de uma transação

Estados de uma transação

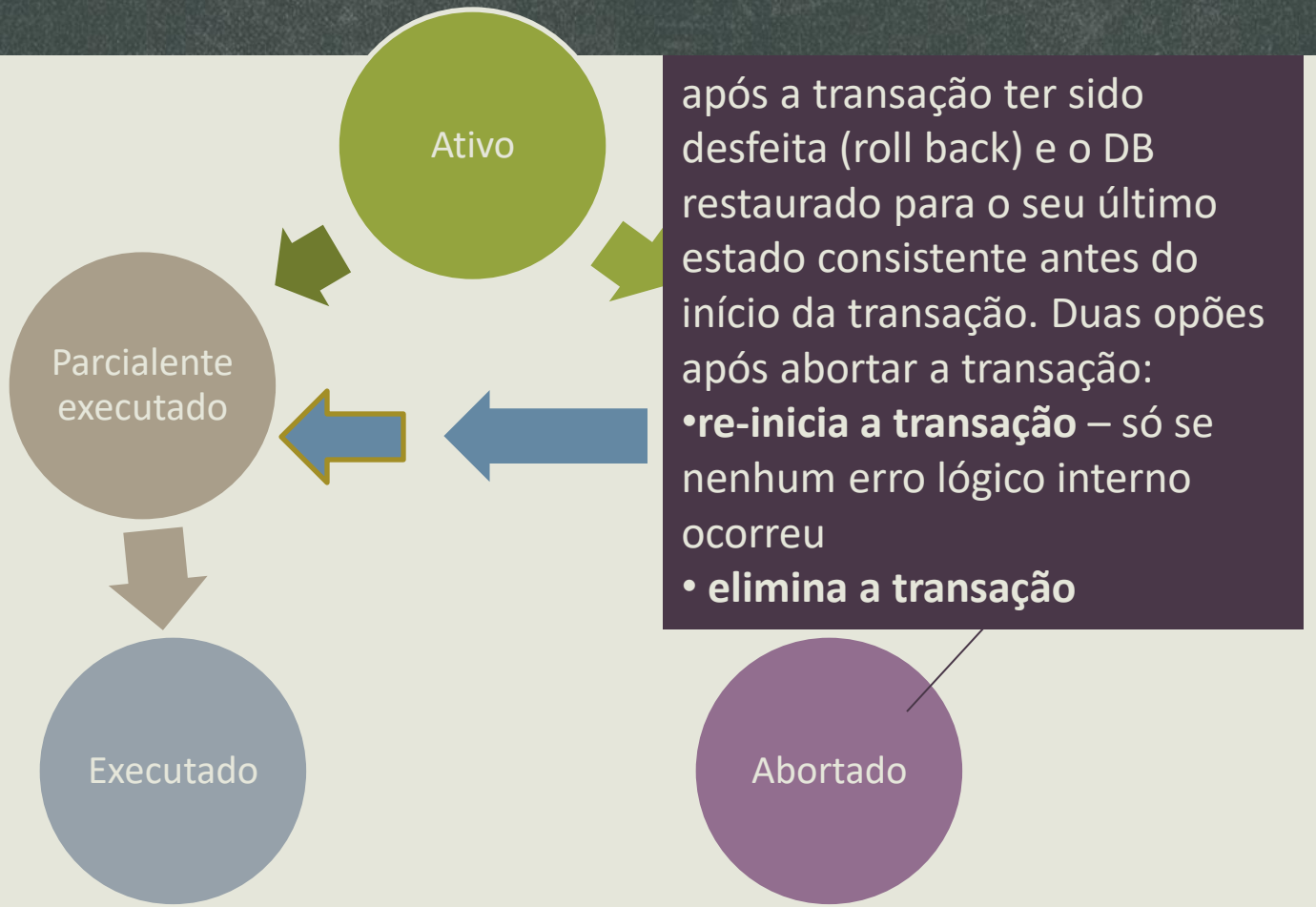


Estados de uma transação

Estados de uma transação



Estados de uma transação



Estados de uma transação

Execução Concorrente

- Os sistemas de processamento de transações normalmente permitem que várias transações sejam executadas ao mesmo tempo

- Essa atualização de dados simultânea causa várias complicações com a consistência dos dados o que gera um trabalho extra no gerenciamento dos mesmos.

- Vantagens
  - Melhor throughput e utilização dos recursos
  - Tempo de resposta reduzido

Execução Concorrente

- As seqüências de execução de transações em um SGBD são denominados ***schedules***.

- Eles determinam a ordem cronológica em que as instruções de uma transação são executados no sistema.

Execução Concorrente

- Schedule serial
  - Consiste em uma seqüência de instruções de várias transações, em que as instruções pertencentes a uma única transação aparecem juntas nesse schedule.

- Schedule não-serial
  - o sistema pode executar uma transação por um tempo, depois executar a segunda transação por algum tempo e depois voltar à primeira transação por algum tempo e, assim por diante, compartilhando o tempo de CPU para a execução das diversas transações.

Execução Concorrente

- A tarefa do SGBD é garantir que qualquer schedule executado deixe o banco de dados em um estado consistente.
- O componente de controle de acesso concorrente que possui esta tarefa.
- Para ilustrar o conceito de schedules seriais e não seriais, consideramos alguns exemplos:
  - Problema clássico do saque em conta
  - Transferência entre contas
    - T1 transfere R$50,00 da conta A para a conta B,
    - Enquanto T2 transfere 10% da conta A para a conta B.

Execução Concorrente

- Problema 2
- Imagine um saque em conta bancária, cuja transação é a seguinte:
  1. Leia Saldo Conta
  2. Diminua o Valor do Saque
  3. Salve Novo Saldo Conta

Schedule Serial

- Uma transação executa após a outra

T1	T2
read (Saldo, s) s = s - 400 **write (Saldo,s)**	
	read (Saldo, s) s = s - 400 **write (Saldo,s)**

- Qual o saldo final depois de dois saques de 400?

Schedule não serial

Imagine que as transações podem ser interrompidas “no meio”

- A primeira executa primeira linha
  - Lê o saldo de 500
  - Após isso ela deve parar pois chegou a vez de outro processo
- A segunda realiza por completo
  - Lê também um saldo de 500
  - Realiza o saque
- A primeira retoma
  - Com o saldo lido de 500
  - Também realiza o saque
- Conclusão
  - Com 500 em saldo
  - Foi permitido sacar 800

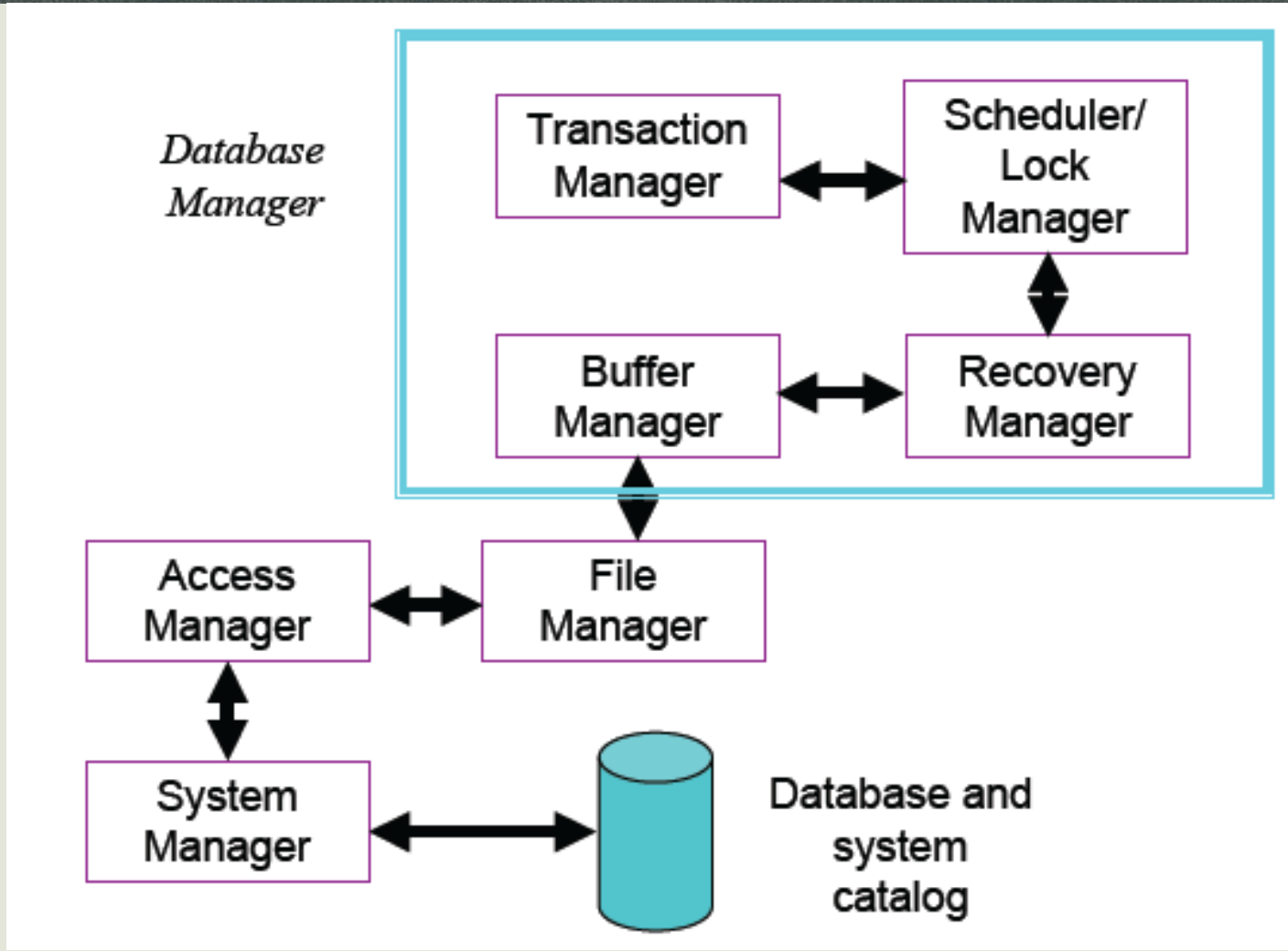
Schedule não serial

- Acompanhe visualmente

T1	T2
read (Saldo, s)	
	read (Saldo, s)
s = s - 400 **write (Saldo,s)**	
	s = s - 400 **write (Saldo,s)**

- Qual o problema nesta execução? Qual o saldo ao final depois de dois saques de 400?

Subsistemas de trasações em um SGBD



Subsistemas de trasações em um SGBD

- ***Transaction Manager****. Coordena as transações após a solicitação do* programa de aplicação. Comunica com o *scheduler.*
- ***Scheduler*** *implementa a estratégia para o controle de* concorrência.
- Se ocorre qualquer falha, o ***recovery manager*** encarrega-se de tratá-la.
- ***Buffer manager*** *tem a tarefa de transferir dados* entre o disco e a RAM.
- ***File manager*** *manipula os ficheiros subjacentes e* gere a alocação de espaço em disco.
  - *File manager não manipula diretamente a* entrada/saída física dos dados; em vez disso, passa os pedidos para o *access manager.*
- Um método de acesso apropriado é usado para ler ou escrever dados para o ***system manager****.*

Bibliografia

1. *R. Ramakrishnan and J. Gehrke. "Database Management Systems". Addison-Wesley, 2003 (cap.16).*
2. *Aula do professor* Otacílio José Pereira

Dúvidas

José Osvano da Silva
joseosvano@unipac.br