TRANSAÇÕES

Sérgio Mergen

Versão modificada de Database System Concepts, 5th Ed.

Tópicos

- Conceito de Transação
- Estados de uma transação
- Definição de uma transação em SQL
- Conceito de Schedule
- Propriedades ACID

Conceito de Transação

- Uma **transação** é uma funcionalidade que acessa e possivelmente altera diversos registros de tabelas

- Ex.
  - em um sistema bancário, uma das funcionalidades existentes é a de transferência entre contas correntes
  - Para implementar essa funcionalidade, é necessário
    - acessar registros de contas e
    - gravar registros de contas

Conceito de Transação

- O exemplo abaixo mostra o script de uma transação em MySQL escrita em uma procedure
  - A transação transfere de 50 reais entre as contas 1 e 2

```
DECLARE s1 INT;
DECLARE s2 INT;

START TRANSACTION;

SELECT saldo INTO s1 FROM conta WHERE id = 1 FOR UPDATE;
SELECT saldo INTO s2 FROM conta WHERE id = 2 FOR UPDATE;

-- faz alguma lógica envolvendo s1 e s2 ...

-- atualiza os registros envolvidos
UPDATE conta SET saldo = saldo - 50 WHERE id = 2;
UPDATE conta SET saldo = saldo + 50 WHERE id = 1;

COMMIT;
```

Conceito de Transação

- Para simplificar, os componentes de uma transação serão abstraídos
  - Atualizações são representadas pela palavra **write**
  - Leituras são representadas pela palavra **read**
  - Letras simbolizam o conteúdo sendo lido/gravado

- Ex. a transação ao lado (T1) transfere R$50 da conta A para a conta B:
  - As operações aritméticas estão trocando o valor dos saldos das contas

T1
read (A) **A := A - 50** **write *(A)*** **read *(B)*** **B = B + 50** **write *(B)***

Tópicos

- Conceito de Transação
- Estados de uma transação
- Definição de uma transação em SQL
- Conceito de Schedule
- Propriedades ACID

Estados de uma Transação

- Uma transação que termina com sucesso terá a instrução de **commit** como a última instrução

T1
read (A) **A := A - 50** **write *(A)*** **read *(B)*** ***B := B + 50*** **write *(B)*** ***commit***

Estados de uma Transação

- Uma transação que falha terá a instrução de **abort** como a última instrução

T1
read (A) **A := A - 50** **write *(A)*** **read *(B)*** ***abort*** ***B := B + 50*** **write *(B)***

Estados de uma Transação

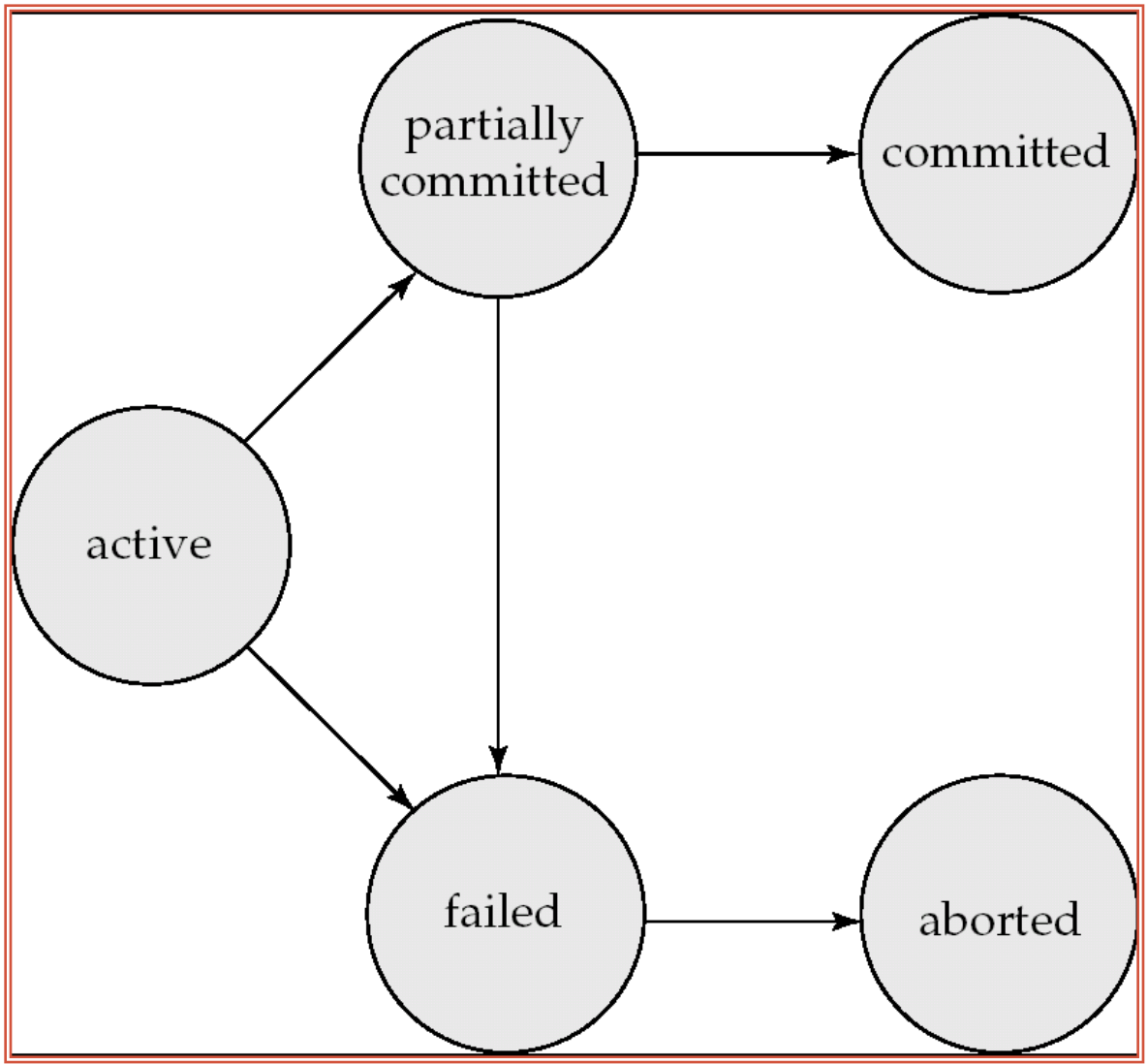
- **Active**
  - o estado inicial; a transação fica nesse estado enquanto estiver executando
- **Partially committed**
  - depois que o último comando for executado.
- **Failed**
  - depois da descoberta que a execução normal não pode prosseguir.

Estados de uma Transação

- **Aborted**
  - depois que a transação foi cancelada e o banco foi restaurado ao estado anterior a execução.
  - Existem duas formas de prosseguir:
    - Reiniciar a transação
      - faz sentido somente se o “abort” não foi devido a erros lógicos
    - Matar a transação

- **Committed**
  - depois da execução bem sucedida.

Estados de uma Transação



Tópicos

- Conceito de Transação
- Estados de uma transação
- Definição de uma transação em SQL
- Conceito de Schedule
- Propriedades ACID

Definição de Transação no SQL

- Em linguagens de programação
  - uma transação é iniciada de forma implícita.
  - O término é indicado por:
  - **Commit:** encerra transação atual e inicia uma nova.
  - **Rollback:** leva ao término mal sucedido da transação.

- Ex. em JDBC, são usadas as funções de objetos Connection
  - connection.commit();
  - connection.rollback();

Definição de Transação no SQL

- Em quase todos os SGBDs, por padrão, toda instrução isolada é comitada
  - Esse commit implícito pode ser desativado usando alguma diretiva

- Ex. em JDBC
  - connection.setAutoCommit(false);

Definição de Transação no SQL

- Scripts SQL executados diretamente no SGBD também podem ser tratados como transações
  - Usando START TRANSACTION e COMMIT/ROLLBACK.

- Se o START TRANSATION não for usado
  - Cada comando é tratado como uma transação

```
START TRANSACTION

SELECT ...

INSERT ...

Commit; -- ou rollback se for para abortar a transação
```

Tópicos

- Conceito de Transação
- Estados de uma transação
- Definição de uma transação em SQL
- Conceito de Schedule
- Propriedades ACID

Conceito de Schedule

- **Schedule** – uma sequência de instruções que especificam a ordem cronológica com que as instruções de transações concorrentes são executadas
  - Um schedule para um conjunto de transações deve
    - possuir todas as instruções dessas transações
    - preservar a ordem com que as instruções aparecem em cada transação.

n.	T1	T2
1	**read (A)**	
2	**A := A - 50**	
3	**write *(A)***	
4		**read(A)**
5		**read(B)**
6		**print(A + B)**
7	**read *(B)***	
8	***B := B + 50***	
9	**write *(B)***	

Conceito de Schedule

- Um schedule é dito concorrente se as instruções de suas transações são executadas de forma intercalada

- O schedule ao lado é concorrente:
  - São executadas instruções de T1
  - Depois T2
  - E depois T1 novamente

T1	T2
read (A)	
A := A - 50	
write *(A)*	
	read(A)
	read(B)
read *(B)*	
	print(A + B)
	commit
B := B + 50	
write *(B)*	
commit	

Conceito de Schedule

- Um schedule é dito serial se as transações são executadas uma após a outra
- Ou seja, não há intercalação de instruções
- O schedule ao lado é serial:
- Pode ser escrito da seguinte forma: <T1, T2>

T1	T2
read (A) **A := A - 50** **write *(A)*** **read *(B)*** ***B := B + 50*** **write *(B)*** ***commit***	**read(A)** **read(B)** **commit**

Tópicos

- Conceito de Transação
- Estados de uma transação
- Definição de uma transação em SQL
- Conceito de Schedule
- Propriedades ACID

Propriedades ACID

- Para preservar a integridade do banco de dados, um SGBD deve garantir que as transações satisfaçam as propriedades ACID:
  - Atomicidade
  - Consistência
  - Isolamento
  - Durabilidade

Propriedades ACID

- **Atomicidade**
  - Ou todas as operações de uma transação são refletidas no banco, ou nenhuma é.

- No caso ao lado
  - Suponha que a transação falhe depois do passo 3 e antes do passo 6
    - E o SGBD já gravou o **write(A**) no banco de dados
  - Nesse cenário, a transação não foi atômica
    - dinheiro será “perdido”, levando o banco de dados a um estado inconsistente

T1
1. read (A)
2. A := A - 50
3. write *(A)*
falha
4. read *(B)*
5. B = B + 50
6. write *(B)*
7. Commit

Propriedades ACID

- **Durabilidade**
  - uma vez que o usuário tenha sido notificado que a transação completou (ex., que a transferência foi bem sucedida)
    - as atualizações decorrentes da transação devem persistir mesmo que ocorram falhas posteriores.

- No caso ao lado
  - Suponha que o usuário recebeu a confirmação da transferência
  - E o SGBD falhou antes das mudanças terem sido persistidas no banco
  - Nesse cenário, a transação não teve efeitos duráveis

T1
read (A) **A := A - 50** **write *(A)*** **read *(B)*** **B = B + 50** **write *(B)*** **Commit**

Propriedades ACID

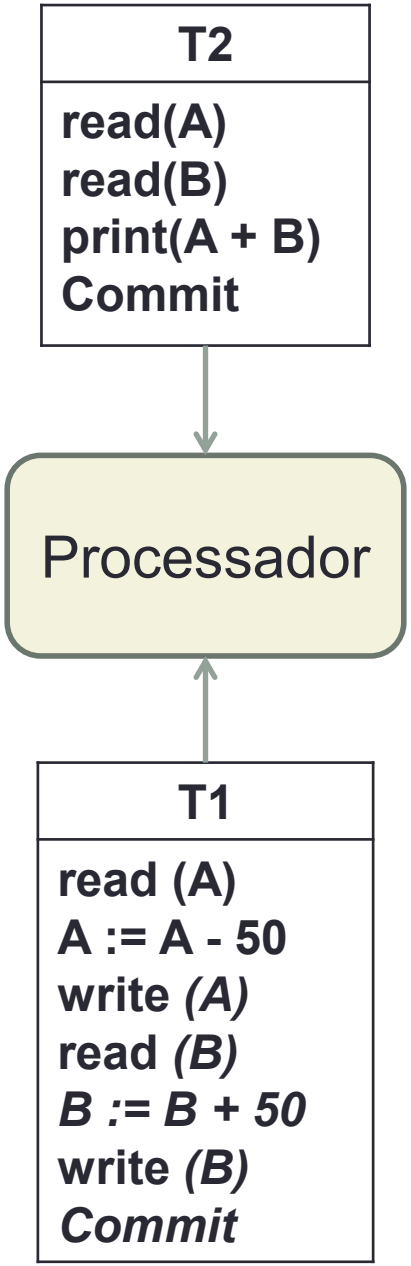
- **Consistência**:
  - A execução de uma transação deve deixar o banco em um estado correto

- No caso ao lado
  - Suponha que o SGBD tenha retirado 50 reais da conta A e adicionado 500 reais na conta B
  - Nesse cenário, a transação não gerou resultados consistentes
    - O valor retirado e o valor depositado não são iguais

T1
read (A) **A := A - 50** **write *(A)*** **read *(B)*** **B = B + 50** **write *(B)*** **Commit**

Propriedades ACID

- **Isolamento**
  - Requisito necessário quando transações diferentes disputam uso do processador
    - Transações concorrentes
  - Mesmo que as transações usem o processador de forma intercalada
    - A impressão é a de que elas são executadas uma após a outra



Propriedades ACID

- **Isolamento**

- No caso ao lado
  - Suponha que, entre os passos 3 e 6 em T1
    - outra transação T2 teve acesso ao banco parcialmente atualizado
  - Nesse cenário
    - T2 verá um banco inconsistente
    - a soma *A* + *B* será menor do que deveria ser.

T1	T2
1. read (A) **2.A := A - 50** **3. write *(A)***	
	1.read(A) **2.read(B)** **3.print(A + B)** **4.Commit**
4. read *(B)* **5. *B := B + 50*** **6. write *(B)*** ***7. Commit***	

Propriedades ACID

- **Isolamento**

- Pode ser garantido de forma simples pela execução **serial** das transações
  - Isto é, uma após a outra.

- Entretanto
  - executar múltiplas transações de forma concorrente traz benefícios, como veremos mais adiante

T1	T2
1. read (A) 2.A := A - 50 3.write *(A)* 4.read *(B)* 5.*B := B + 50* 6.write *(B)*	1.read(A) 2.read(B) 3.print(A + B)

Propriedades ACID

- **Atomicidade**. Ou todas as operações de uma transação são refletidas no banco, ou nenhuma é.
- **Consistência**. A execução de uma transação isolada preserva a consistência do banco.
- **Durabilidade**. Depois que uma transações finalizou, as mudanças feitas devem persistir mesmo que ocorram falhas posteriores.
- **Isolamento**. Mesmo que muitas transações executem concorrentemente, cada transação desconhece as demais execuções.