Introdução a Transações

1

Transações

- **Transação** é a unidade lógica de processamento do banco de dados;
- Ela inclui um ou mais comandos SQL que fazem escrita e leitura de dados:
  - Leitura: SELECT;
  - Escrita: INSERT, UPDATE e DELETE.
- Uma transação é delimitada por um início e fim:
  - BEGIN TRANSACTION;
  - END TRANSACTION.
- Por padrão, todo comando SQL é implicitamente uma transação.

2

2

Operações de Leitura e Escrita

- As operações em um banco de dados têm por finalidade ler ou escrever um item de dado que pode ser:
  - Um campo de uma tabela, vários campos de uma ou mais tabelas ou uma tabela inteira.
- Entretanto, a unidade básica de dado transferida do disco para a memória principal do computador é um bloco;
- Operações básicas de **leitura** e **escrita**
  - **read_item(X**): Lê um item do banco de dados nomeado X para uma variável de programa. Para simplificar, assume-se que a variável de programa é também nomeada X.
  - **write_item(X**): Escreve o valor da variável de programa X em um item do banco de dados nomeado X.

Operações de Leitura

- O comando **read_item(X)** inclui os seguintes passos:
  1. Localiza o endereço do bloco de disco que contém o item X;
  2. Copia o bloco de disco para um buffer na memória principal (se o bloco já não está em algum buffer da memória);
  3. Copia o item X do buffer para a variável de programa chamada X.

Operações de Escrita

- O comando **write_item(X**) inclui os seguintes passos:
  1. Localiza o endereço do bloco de disco que contém o item X.
  2. Copia o bloco de disco para um buffer na memória principal (se o bloco já não está em algum buffer da memória).
  3. Copia o item X da variável de programa chamada X em seu local correto no buffer.
  4. Armazena o bloco atualizado do buffer de volta para o disco (imediatamente ou posteriormente).

5

5

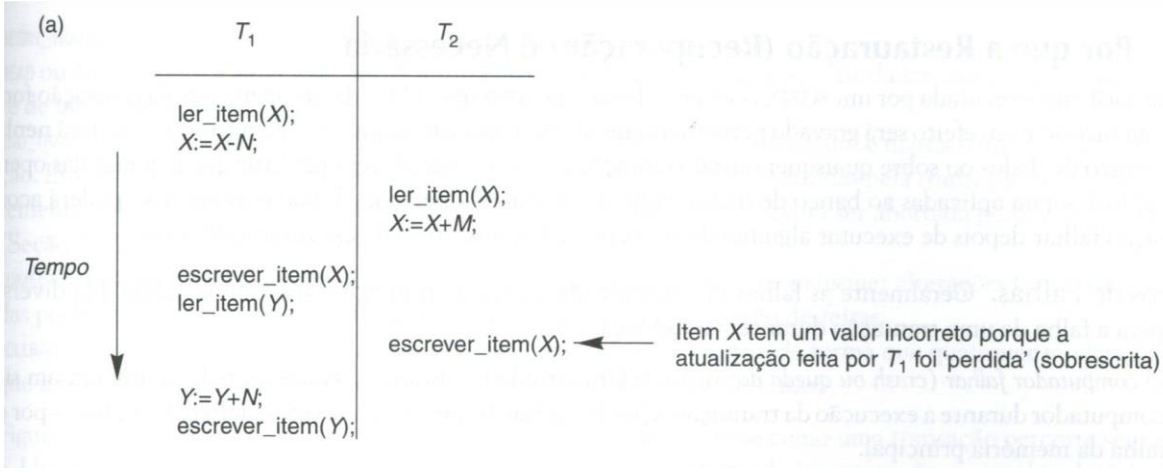
Interferências entre transações

- **O problema da Atualização Perdida**
  - Ocorre quando duas transações que acessam os mesmos itens do banco de dados têm suas operações intercaladas de forma que faça com que os valores de alguns itens fiquem incorretos.
- **O problema da Atualização Temporária (Leitura suja)**
  - Ocorre quando uma transação atualiza um item do banco de dados e então a transação falha por alguma razão.
  - O valor do item atualizado é acessado por outra transação antes que ele retorne ao seu valor original.
- **O problema do Sumário Incorreto**
  - Se uma transação aplicar uma função de agregação para um número de registros enquanto outras transações estiverem atualizando alguns desses registros, a função de agregação deverá calcular alguns valores antes deles serem atualizados e outros depois de feita a atualização.

6

6

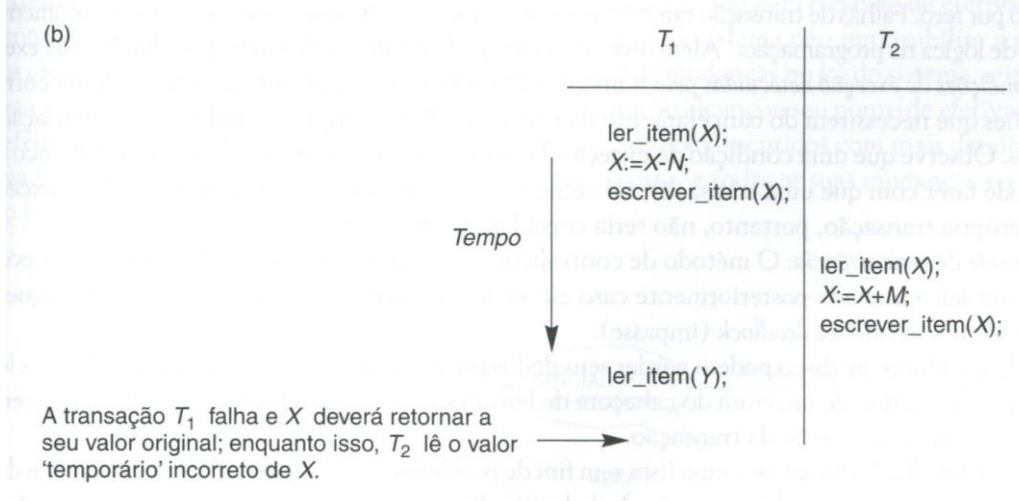
O problema da Atualização Perdida



7

7

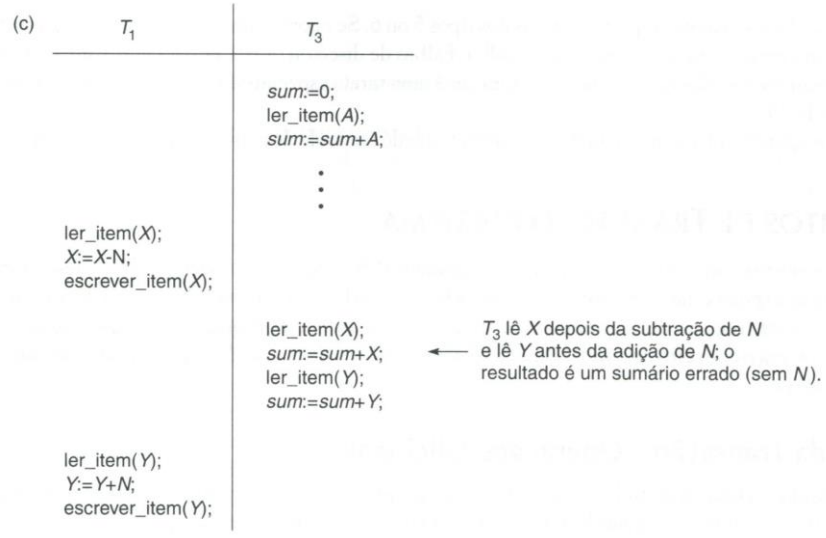
O problema da Atualização Temporária



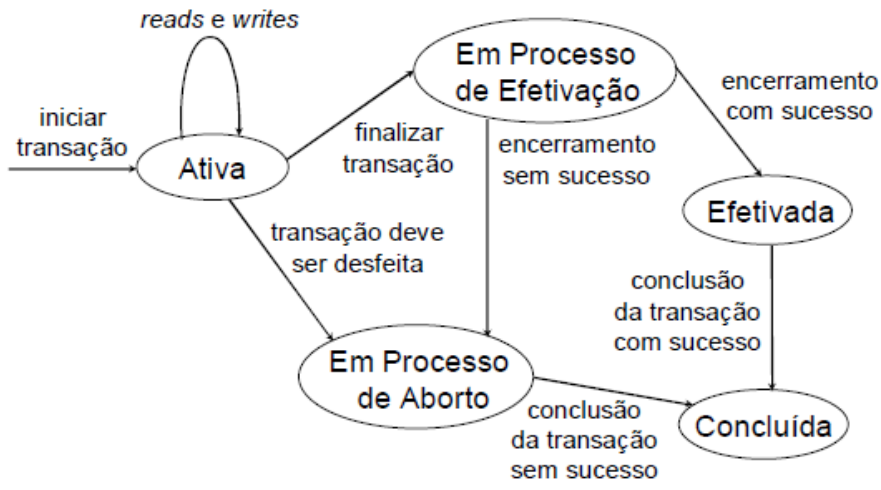
8

8

O problema do Sumário Incorreto



Ciclo de vida de um transação



Ciclo de vida de um transação

- Ativa:
  - Estado inicial de toda transação selecionada para executar;
  - Enquanto ativa, uma transação executa uma ou mais operações read e write.
- Em Processo de Efetivação:
  - Entra nesse estado após executar sua última operação (solicitação de COMMIT);
  - Neste momento, o BD precisa garantir que as suas atualizações sejam efetivadas com sucesso.

11

11

Ciclo de vida de um transação

- Efetivada:
  - Entra nesse estado após o SGBD confirmar que todas as modificações da transação estão garantidas no BD (COMMIT OK).
    - Exemplos: gravação em Log, descarga de todos os buffers em disco.
- Em Processo de Aborto:
  - Entra nesse estado se não puder prosseguir a sua execução;
  - Pode passar para esse estado enquanto:
    - Ativa: exemplo, violação de RI;
    - Em processo de efetivação: exemplo pane no S.O.
  - Suas ações já realizadas devem ser desfeitas (ROLLBACK).

12

12

Ciclo de vida de um transação

- Concluída:
  - Estado final de uma transação;
  - Indica uma transação que deixa o sistema:
    - As informações da transação mantidas em catálogo podem ser excluídas;
    - Se a transação não concluiu com sucesso, ela pode ser reiniciada automaticamente.

13

13

Propriedades ACID

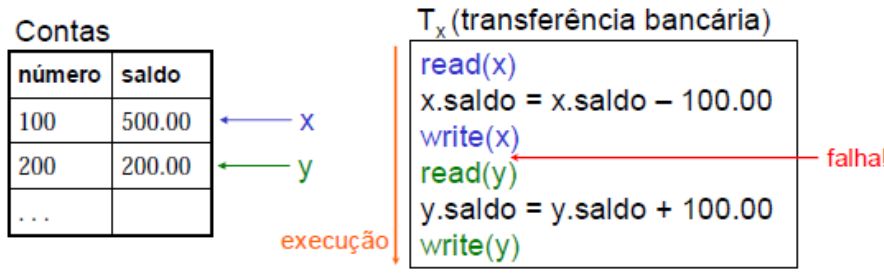
- **A**tomicidade:
  - Todas as operações da transação devem ser efetivadas com sucesso no BD ou nenhuma das operações será efetivada;
- **C**onsistência:
  - Uma transação sempre deve conduzir o BD de um estado consistente para outro estado também consistente;
- **I**solamento:
  - Em um conjunto de transações concorrentes, cada transação deve ser executada no BD como se ela fosse a única;
- **D**urabilidade:
  - O BD deve garantir que as modificações realizadas por uma transação que concluiu com sucesso sejam persistidas.

14

14

Atomicidade - Exemplo

- Uma transação pode manter o BD em um estado inconsistente durante a sua execução.



15

15

Isolamento – Exemplo

T_1	T_2
read(A)	
A = A – 50	
write(A)	
	read(A)
	A = A+A*0.1
	write(A)
read(B)	
B = B + 50	
write(B)	
	read(B)
	B = B - A
	write(B)

escalonamento válido

T_1	T_2
read(A)	
A = A – 50	
	read(A)
	A = A+A*0.1
	write(A)
	read(B)
write(A) ← T_1 interfere em T_2	
read(B)	
B = B + 50	
write(B)	
	B = B - A
	write(B) ← T_2 interfere em T_1

escalonamento inválido

16

16

Sintaxe da transação em SQL

- Uma transação SQL é definida pelo seguinte comando:

```
BEGIN [ WORK | TRANSACTION ] [ transaction_mode [, ...] ]

where transaction_mode is one of:

   ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }
   READ WRITE | READ ONLY
   [ NOT ] DEFERRABLE
END [ WORK | TRANSACTION ];
```

17

17

Características das Transações em SQL

Características especificadas por uma cláusula TRANSACTION:

- READ ONLY ou READ WRITE.
  - Por padrão a transação é READ WRITE, a menos que o nível de isolamento seja READ UNCOMITTED, nesta caso é assumido READ ONLY.
- DEFERRABLE
  - Só é permitida em transações com isolamento SERIALIZABLE;
  - Permite que a transação bloqueie todos os dados de uma só vez e execute sem risco de ser cancelada por uma falha de serialização.

18

18

Problemas associados aos Níveis de Isolamento

Potenciais problemas com níveis de isolamento baixo:

- **Leitura Suja**:
  - Uma transação T_1 pode ler uma atualização ainda não efetivada de uma transação T_2. Se T_2 falhar e for abortada, então T_1 lerá um valor que não existe e está incorreto.
- **Leitura Não-Repetível** :
  - Uma transação T_1 pode ler um valor em uma tabela. Se depois uma outra transação T_2 atualizar esse valor e T_1 lê-lo novamente, T_1 enxergará um valor diferente.

19

19

Problemas associados aos Níveis de Isolamento

Potenciais problemas com níveis de isolamento baixo:

- **Fantasmas**:
  - Novas linhas sendo lidas usando o mesmo READ como condição.
    - Uma transação T_1 pode ler um conjunto de linhas de uma tabela, provavelmente baseada em alguma condição especificada na cláusula WHERE.
    - Suponha, agora, que uma transação T_2 insira uma nova linha que também satisfaça a condição da cláusula WHERE usada em T_1, dentro da tabela usada por T_1.
    - Se T_1 for repetida, então verá um fantasma, uma linha que não existia anteriormente, chamada fantasma.

20

20

Níveis de Isolamento

- O nível de isolamento de uma transação determina quais dados a transação pode ver quando outras transações estão sendo executadas simultaneamente;
- O SQL define 4 níveis de isolamento de transação:
  - READ UNCOMMITTED;
  - READ COMMITTED;
  - REPEATABLE READ;
  - SERIALIZABLE.

21

21

Níveis de Isolamento

- READ UNCOMMITTED:
  - Ele permite a leitura de dados de transações não efetivadas;
  - Ele permite a ocorrência de leitura suja, leitura não repetível e fantasmas.
  - O PostgreSQL não implementa esse tipo de isolamento.
- READ COMMITTED:
  - É o nível padrão do PostgresSQL;
  - Só permite a leitura de dados efetivados antes do início da execução do commando SQL;
  - Entretanto, não garante que dois comandos SQL sucessivos enxerguem os mesmos dados, mesmo estando dentro da mesma transação;
  - Portanto, não suporta a leitura repetível.

22

22

Níveis de Isolamento

- REPEATABLE READ:
  - Garante que comandos SQL consecutivos dentro de uma única transação veem os mesmos dados;
  - Ou seja, eles não veem alterações feitas por outras transações que foram confirmadas após o início da transação.
  - Porém, não garante a leitura sem o surgimento de novos registros (fantasmas).
- SERIALIZABLE:
  - Fornece o isolamento mais rigoroso entre as transações;
  - Emula a execução serial das transações, como se todas as transações fossem executadas uma após a outra;
  - Entretanto, um ponto negativo é que ele causa muitas falhas de serialização.

23

23

Níveis de Isolamento

		Tipos de Problemas		
		Leitura Suja	**Leitura Não-Repetível**	**Fantasmas**
Níveis de Isolamento	**READ UNCOMMITED**	Sim	Sim	Sim
	READ COMMITED	Não	Sim	Sim
	REPEATABLE READ	Não	Não	Sim
	SERIALIZABLE	Não	Não	Não

24

24

COMMIT, ROLLBACK e SAVEPOINT

- COMMIT
  - Ele confirma a transação atual;
  - Todas as alterações feitas pela transação tornam-se visíveis e têm sua durabilidade garantida em caso uma falha.
- ROLLBACK
  - Ele reverte a transação atual e faz com que todas as alterações feitas por ela sejam desfeitas;
- SAVEPOINT
  - Ele permite descartar seletivamente partes da transação, enquanto confirma o restante dela.

25

25

Transação em SQL – Exemplo

Exemplo de uma transação bancária.

```
BEGIN ISOLATION LEVEL READ COMMITTED;
    UPDATE cliente SET saldo = saldo - 100 WHERE nome = 'Paulo';
    SAVEPOINT savepoint_1;
    UPDATE cliente SET saldo = saldo + 100 WHERE nome = 'Lucia';
    -- Opa... Lucia não é a cliente correta!!
    ROLLBACK TO savepoint_1;
    UPDATE cliente SET saldo = saldo + 100 WHERE nome = 'Luiz';
END;
```

26

26