INSTITUTO FEDERAL
Rio Grande do Norte | Campus Parnamirim

Controle de Concorrência

Prof. Fábio Procópio

fabio.procopio@ifrn.edu.br

Tecnologia em Sistemas para Internet – Programação em Bancos de Dados

Relembrando...

- Na aula passada, discutimos sobre gerência de transações;
- Dando continuidade ao tema, nesta aula, falaremos sobre Controle de Concorrência. Vamos nessa?!

Introdução

- Concorrência ocorre quando muitas transações tentam acessar um mesmo recurso simultaneamente;
- O SGBD deve garantir algum mecanismo para assegurar que a transação A não interferirá na execução da transação B;
- Alguns problemas causados pela concorrência de transações são conhecidos como
  1. atualização perdida
  2. dependência sem COMMIT
  3. análise inconsistente
  4. leitura fantasma

Atualização perdida

Transação A	tempo	Transação B
SELECT t	t_1	
	t_2	SELECT t
UPDATE t	t_3	
	t_4	UPDATE t

Em t_4, a transação B atualiza a linha t, sem analisar a mudança que foi feita, no instante t_3, pela transação A.

Dependência sem commit – 1 de 2

Leitura

Transação A	tempo	Transação B
	t_1	UPDATE t
SELECT t	t_2	
	t_3	ROLLBACK

Ocorre quando uma transação A **lê** um registro que foi atualizado pela transação B, mas que **ainda não foi confirmada** por um COMMIT.

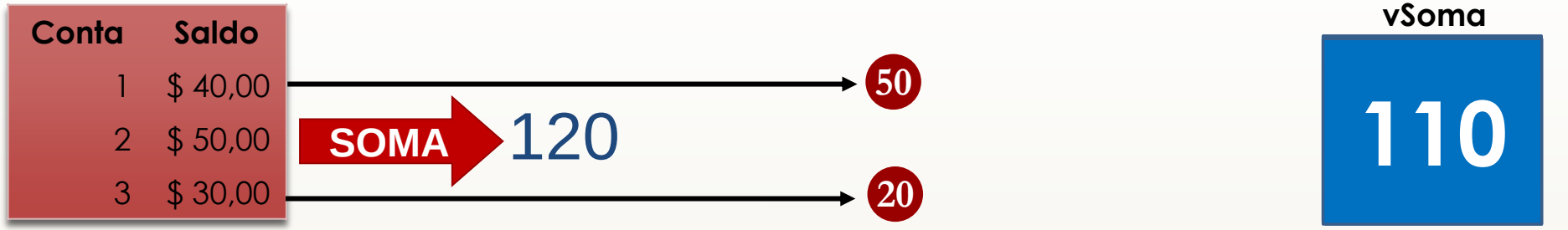
Dependência sem commit – 2 de 2

Escrita

Transação A	tempo	Transação B
	t_1	UPDATE t
UPDATE t	t_2	
	t_3	ROLLBACK

Ocorre quando uma transação A **atualiza** (pior caso) um registro que foi atualizado pela transação B, mas que **ainda não foi confirmada** por um COMMIT.

Análise inconsistente



	Transação A	Transação B
t_0	var vSoma := 0	
t_1	vSoma := vSoma + Saldo(Conta1) **$ 40,00**	
t_2	vSoma := vSoma + Saldo(Conta2) **$ 50,00**	
t_3		Saldo(Conta3) = Saldo(Conta3) - 10
t_4		Saldo(Conta1) = Saldo(Conta1) + 10
t_5		COMMIT
	vSoma := vSoma + Saldo(Conta3) **$ 20,00**	

Leitura fantasma

Transação A	tempo	Transação B
SELECT COUNT(linhas)	t_1	
	t_2	INSERT/DELETE linhas
SELECT COUNT(linhas)	t_3	

Bloqueio binário – 1 de 2

- Mecanismo utilizado para **controlar a concorrência** em um banco de dados;
- Enquanto uma transação acessa um item de dados nenhuma outra transação pode modificá-lo (essa afirmação se refere a qual propriedade ACID?);
- Existem vários formas de bloqueio, no entanto, aqui veremos apenas
  - exclusivo (X)

    se transação A bloquear um item

      requisições de bloqueios X e S por outras transações serão negadas
  - compartilhado (S)

    se transação A bloquear um item

      requisição de bloqueio **X** por outra transação será **negada**

      requisição de bloqueio **S** por outra transação será **concedida**

Bloqueio binário – 2 de 2

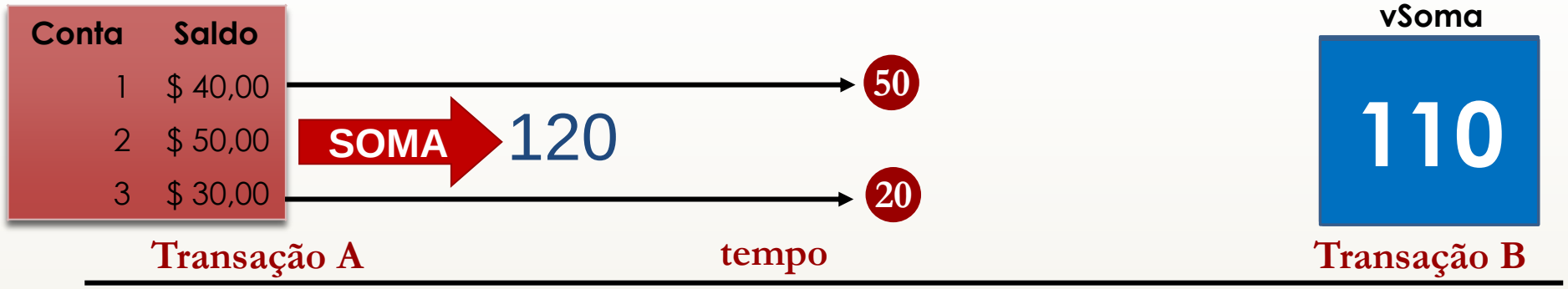
- Matriz de compatibilidade:

	X	S
X	Não	Não
S	Não	Sim

Atualização perdida: tratamento

Transação A	tempo	Transação B
SELECT t adquire um bloqueio S	t_1	
	t_2	SELECT t adquire bloqueio S
UPDATE t adquire bloqueio X espera espera espera espera espera espera espera espera espera espera	t_3 t_4	 UPDATE t bloqueio X espera espera espera espera espera espera

Análise inconsistente: tratamento



var vSoma := 0	t_0	
vSoma := vSoma + Saldo(Conta1) **$ 40,00**	t_1	**bloqueio S sobre a Conta1**
vSoma := vSoma + Saldo(Conta2) **$ 50,00**	t_2	**bloqueio S sobre a Conta2**
bloqueio X sobre a Conta3	t_3	Saldo(Conta3) = Saldo(Conta3) - 10
bloqueio X sobre a Conta1	t_4	Saldo(Conta1) = Saldo(Conta1) + 10
vSoma := vSoma + Saldo(Conta3) **$ 20,00**	t_5	**espera**
bloqueio S sobre a Conta3		**espera**
espera		**espera**

Impasse (*deadlock*) – 1 de 2

- Situação na qual uma ou mais transações esperam simultaneamente para que uma outra libere o recurso pelo qual ela espera;

- É necessário que o sistema detecte e interrompa o *deadlock*
  - a detecção é feita a partir de um Grafo de Espera

  - já a interrupção consiste em escolher uma das transações participantes como vítima para liberar o bloqueio.

- Nem todos os sistemas detectam impasse: alguns usam apenas um tempo de espera para identificar um *deadlock*.

Impasse (*deadlock*) – 2 de 2

Transação A	tempo	Transação B
LOCK EXCLUSIVO sobre r_1	t_1	
	t_2	LOCK EXCLUSIVO sobre r_2
LOCK EXCLUSIVO sobre r_2	t_3	
espera espera espera espera espera espera espera espera espera espera espera	t_4	LOCK EXCLUSIVO sobre r_1 espera espera espera espera espera espera

Tratamento para *deadlocks*

Algoritmo Wait-Die (Espera-Morte)

```
se (A é mais antiga que B) então
    transação A espera a conclusão da B
senão
    transação A é cancelada
fimse
```

Algoritmo Wound-Wait (Ferida-Espera)

```
se (A é mais recente que B) então
    transação A espera a conclusão da B
senão
    transação B é cancelada
fimse
```

Níveis de isolamento – 1 de 3

- Mecanismos que indicam ao SGBD o **grau de interferência** que uma transação aceitará das outras;
- Um nível baixo de isolamento disponibilizará recursos para muitos usuários
  - Porém um nível alto aumentam as chances de uma transação bloquear outras
- O SQL-92 define 4 níveis de isolamento:
  - *Read uncommited*
  - *Read commited*
  - *Repeatable read*
  - *Serializable*

Níveis de isolamento – 2 de 3

1. ***Read uncommited***
   - as transações podem fazer leituras de alterações realizadas por transações concorrentes que ainda não foram efetivadas;
2. ***Read commited***
   - as transações só podem fazem leituras de alterações realizadas depois que estas forem efetivadas no banco de dados;
   - Este é o nível de isolamento *default* do SQL Server;

Níveis de isolamento – 3 de 3

3. ***Repeatable read***
   - tem as mesmas características do isolamento *read commited* e;
   - nenhuma transação poderá alterar uma informação que foi lida pela transação atual até que esta seja encerrada;
4. ***Serializable***
   - oferece o nível de isolamento mais rigoroso;
   - emula a execução serial das transações como se todas fossem executadas uma após a outra e não simultaneamente.

Isolamento x Concorrência

Nível de isolamento	Dependência sem commit	Análise inconsistente	Leituras fantasmas
Read uncommited	Sim	Sim	Sim
Read commited	Não	Sim	Sim
Repeatable read	Não	Não	Sim
Serializable	Não	Não	Não

Referências

1. DATE, C J. **Introdução a Sistemas de Banco de Dados**. Rio de Janeiro: Elsevier, 2003.

2. SILBERSCHATZ, Abraham, KORTH, Henry F. e SUDARSHAN, S. **Sistema de Banco de Dados**. São Paulo: MAKRON Books, 1999.

3. POSTGRESQL. **Isolamento de transação**. Disponível em: http://pgdocptbr.sourceforge.net/pg80/transaction-iso.html. Acessado em: 13 out. 2019.