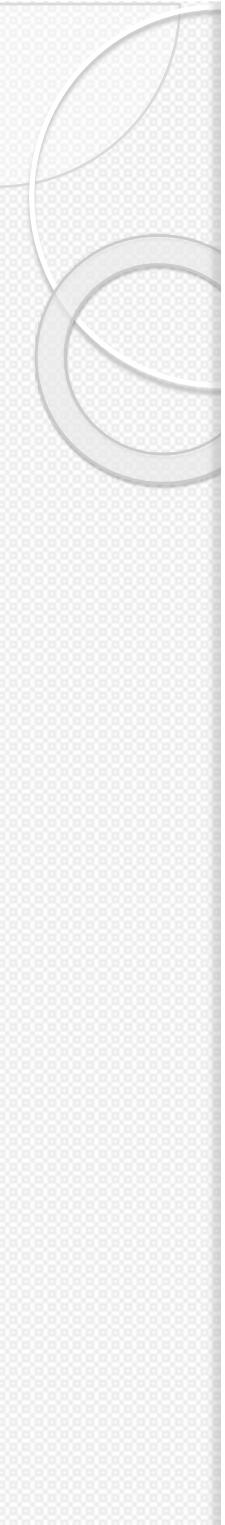


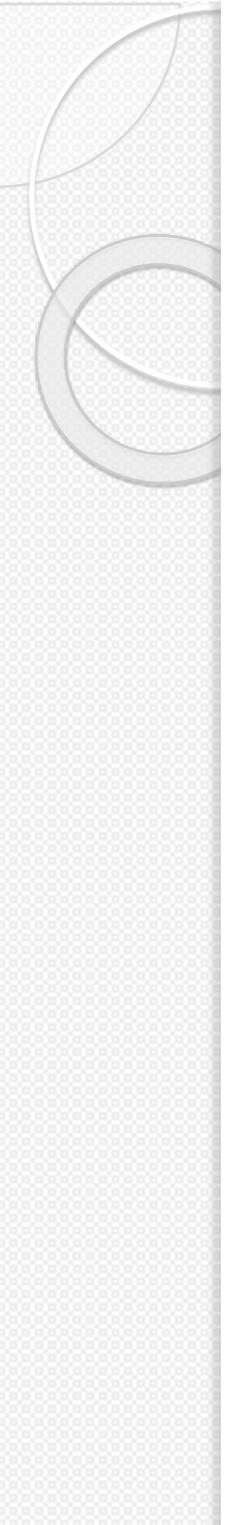
SGBD

Controle de Concorrência de Transações



Gerenciamento de Transações

- Gerenciamento de Transações
 - Transação
 - Escalonamento
 - Protocolos de controle de concorrência

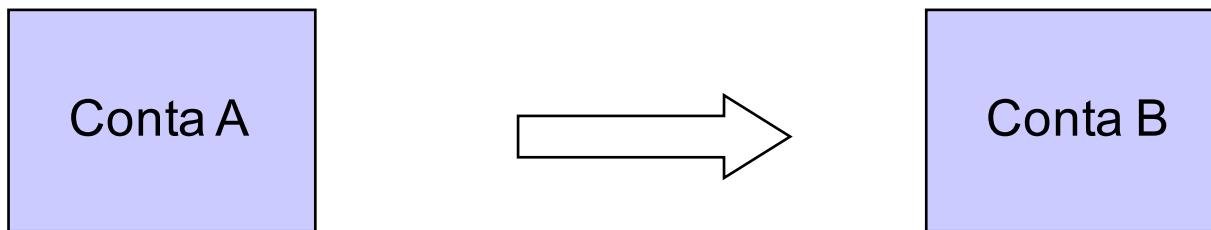


Transação (I)

- Transação (*Eswaram et al*)
 - Abstração que representa a seqüência de operações de bancos de dados resultantes da execução de um programa
- Exemplo: Transferência Bancária
 - Retirar dinheiro de uma conta origem
 - Verificar saldo
 - Subtrair o valor do depósito da conta de origem
 - Atualizar o saldo
 - Depositar na conta destino
 - Ler saldo
 - Somar o valor do saldo na conta destino
 - Atualizar o saldo

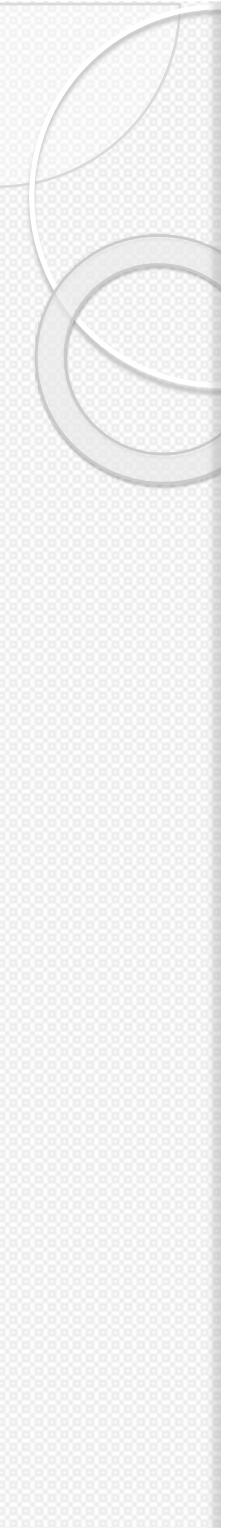
Transação (2)

Transferir: 100 reais



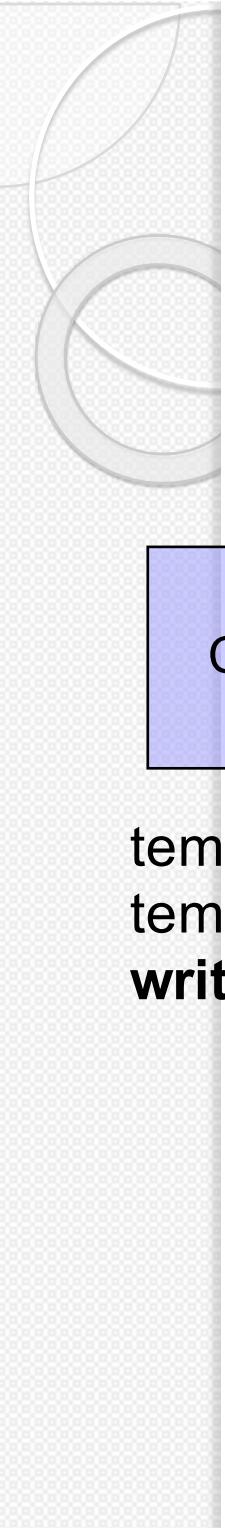
```
temp:=read(saldoA)  
temp: temp - 100  
write(temp)
```

```
temp:=read(saldoB)  
temp: temp + 100  
write(temp)
```



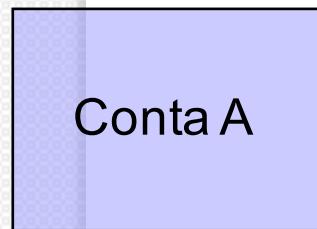
Transação (3)

- Seqüência de operações de *read* e *write*.
 - Exemplo de notação: $T_1 = r_1(x) \; w_1(y) \; c_1$
- Propriedades ACID
 - Atomicidade
 - Consistência
 - Isolamento
 - Durabilidade



Transação (4)

Transferir: 100 reais



Transferir: 100 reais

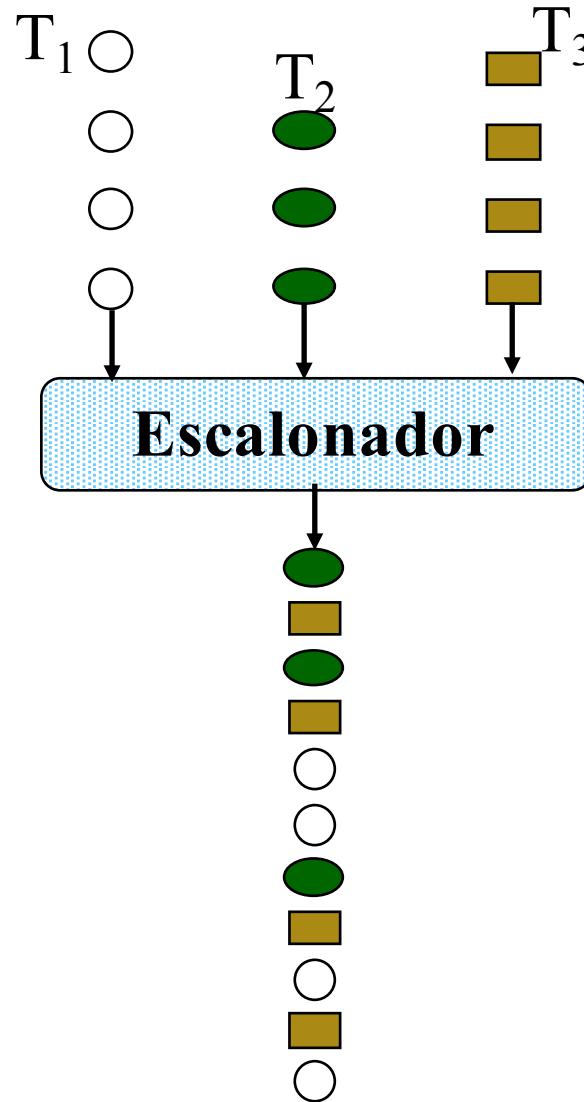


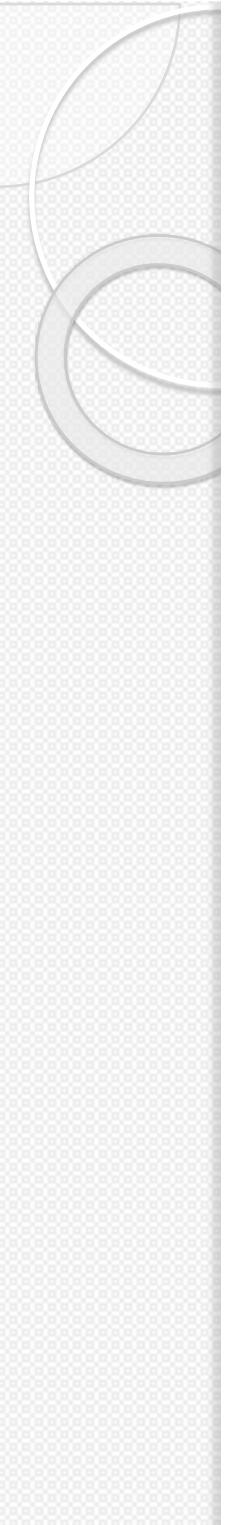
```
temp:=read(saldoA)  
temp: temp - 100  
write(temp)
```

```
temp:=read(saldoC)  
temp: temp - 100  
write(temp)
```



Escalonador (Scheduler)





Escalonamento (I)

- Múltiplas Transações
- Entrelaçar as operações de diferentes transações
- Escalonamento (*schedule*) :

$$T_1 = r_1(x)w_1(y)c_1$$

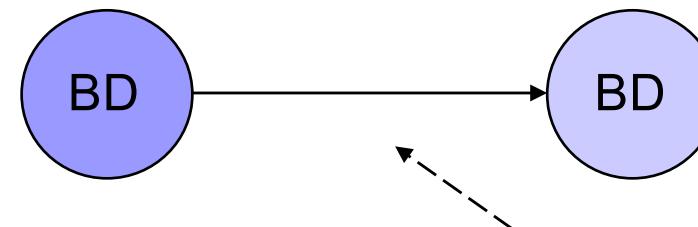
$$T_2 = r_2(u)r_2(y)w_2(x)c_2$$

$$S = r_1(x)r_2(u)w_1(y)r_2(y)w_2(x) c_1c_2$$

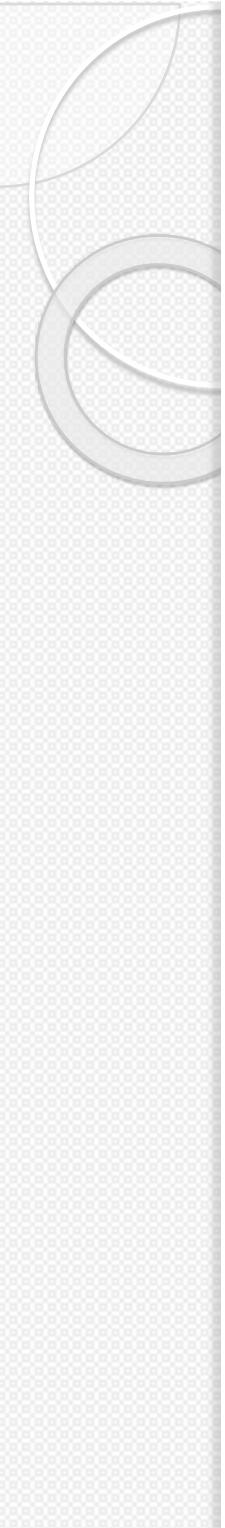
Escalonamento (2)

■ Escalonamento Correto

Estado Consistente

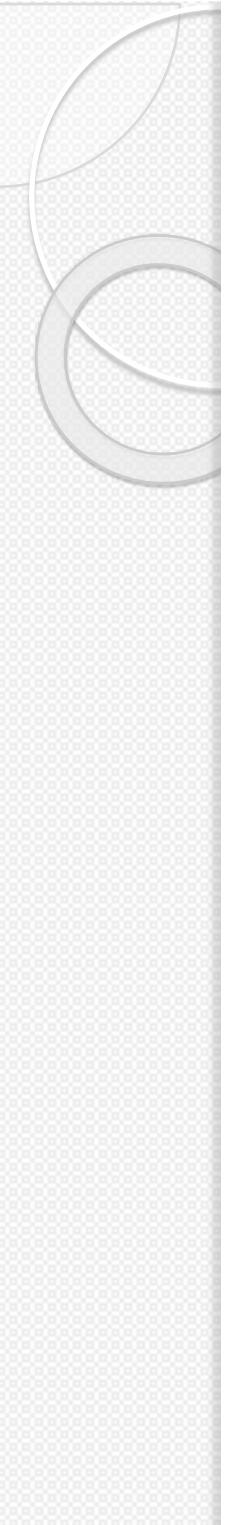


Escalonamento Correto



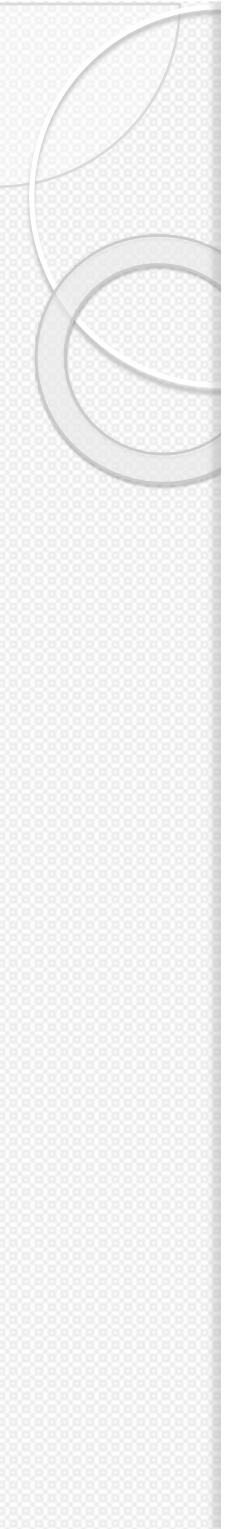
Escalonamento Correto

- **Premissa**
 - A execução de uma transação é correta, se executada isoladamente
 - Produz sempre um estado consistente
- **Teorema**
 - Toda execução serial de transações é correta
 - $S = T_1 T_2 T_3 \dots T_{n-1} T_n$
- **Escalonamento Serial (correto)**



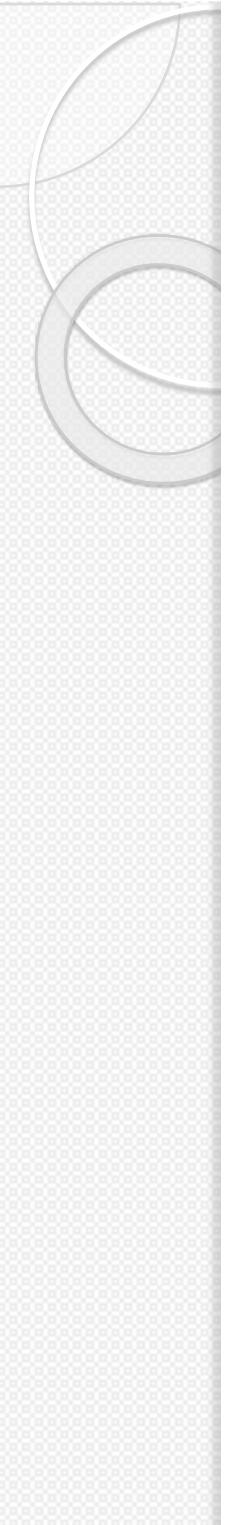
Escalonamento (3)

- Escalonamentos:
 - Escalonamento serial
 - Outros escalonamentos (aumentar o paralelismo)
- Seja um schedule S
 - Se S produz um estado de banco de dados igual ao produzido por alguma execução serial do mesmo conjunto de transações, então a execução S é correta



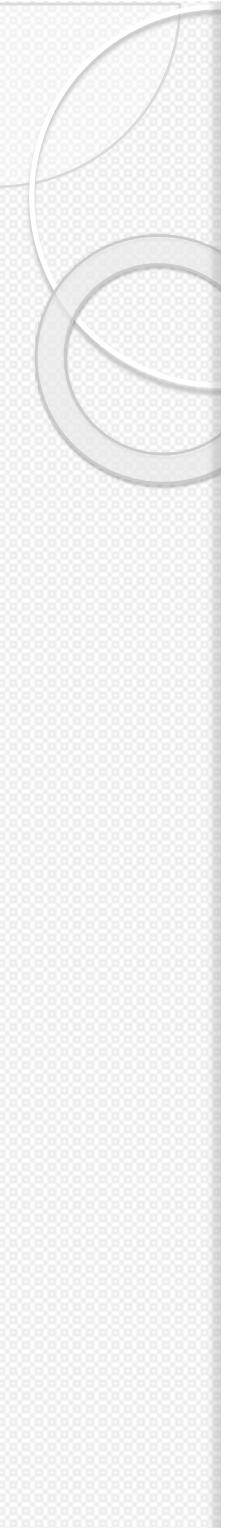
Escalonamentos

- Equivalência de escalonamentos:
 - Dado um conjunto $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$
 - Encontrar escalamentos cuja execuções produzam o mesmo estado no BD que a execução de algum escalonamento serial sobre \mathcal{T} .
- Critério de corretude de serialibilidade.
 - Escalonamento S é equivalente a um escalonador serial.



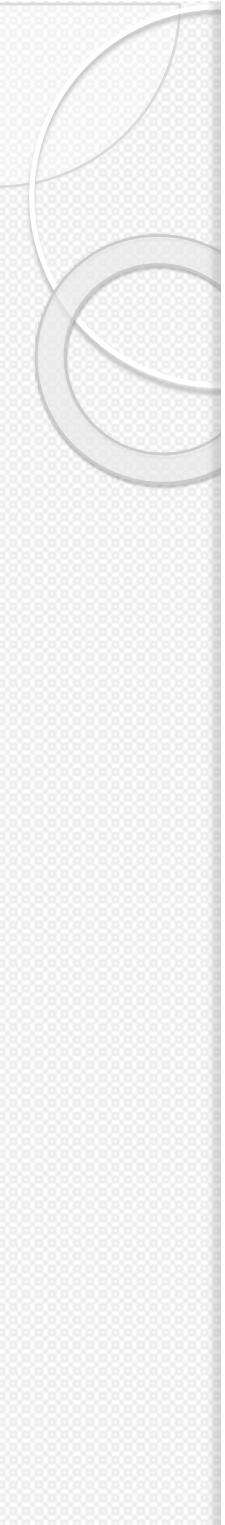
Escalonamento (4)

- Serialização:
 - Identificação de escalonamentos corretos
 - Vários tipos de serialização
- Serialização por conflito
 - Operações Conflitantes :
 - Duas operações são ditas conflitantes, quando agem no mesmo item de dados, pertencem a transações diferentes e pelo menos uma das operações é de escrita.



Equivalência por conflito

- Equivalência de conflito
 - Dois escalonamentos S e S' sobre um conjunto $\mathfrak{S} = \{T_1, T_2, \dots, T_n\}$ são equivalentes de conflito ($S \approx_C S'$) se, e somente se,
 - (i) têm as mesmas operações pertencentes às transações de \mathfrak{S} e
 - (ii) A ordem das operações conflitantes são as mesmas
- Para quaisquer duas operações em conflito $p_i \in OP(T_i)$ e $q_j \in OP(T_j)$, com $i \neq j$, se $p_i <_S q_j$, então $p_i <_{S'} q_j$



Exemplo

$$T_1 = r_1(y)r_1(x)w_1(y) \quad T_2 = r_2(x)r_2(y)w_2(x)$$

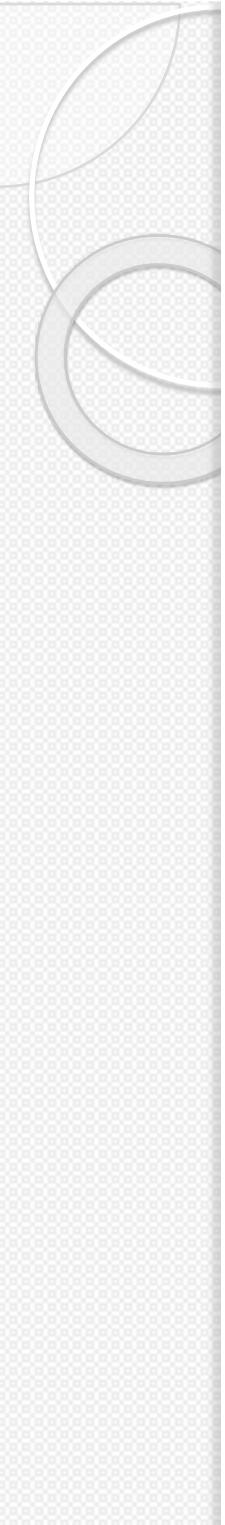
$$S = r_1(y)r_1(x)r_2(x)w_1(y)r_2(y)w_2(x)$$

$$S' = r_1(y)r_1(x)w_1(y)r_2(x)r_2(y)w_2(x)$$

Serializabilidade Por Conflito

- Serializabilidade por conflito
 - S é serializável por conflito, se S é equivalente por conflito a algum *schedule* serial S_S sobre o mesmo conjunto de transações
- Exemplo

$$T_1 = r_1(y) r_1(x) w_1(y)$$
$$T_2 = r_2(x) r_2(y) w_2(x)$$
$$T_3 = r_3(y) r_3(x) w_3(z)$$
$$S = r_3(y) r_1(y) r_1(x) r_2(x) w_1(y) r_2(y) \textcolor{red}{w_2(x)} \textcolor{red}{r_3(x)} w_3(z)$$
$$S_S = \textcolor{brown}{r_3(y)} r_3(x) w_3(z) \textcolor{green}{r_1(y)} r_1(x) w_1(y) r_2(x) r_2(y) \textcolor{red}{w_2(x)}$$



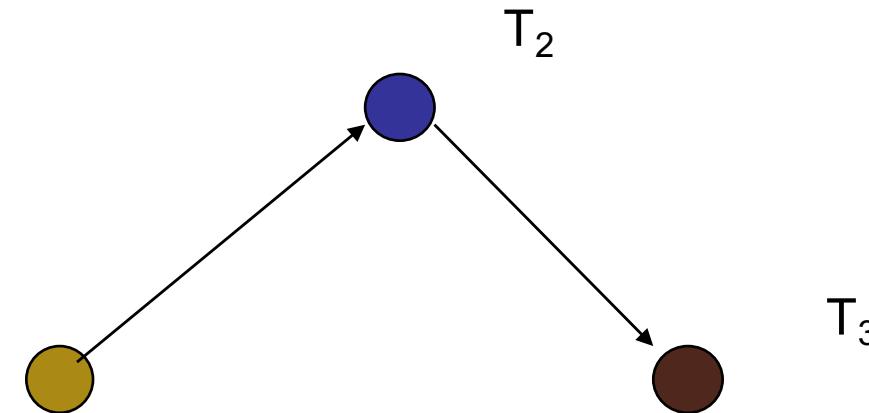
Escalonamento (5)

- Grafo de serialização de um schedule S
 - Definido para um conjunto $T = \{T_1, T_2, \dots, T_n\}$
 - Grafo direcionado
 - $G(N, A)$, onde
 - N são as transações
 - $T_i \rightarrow T_j \in A$
 $\exists p \in OP(T_i), q \in OP(T_j)$, tal que **p conflita com q** e $p <_S q$

Escalonamento (6)

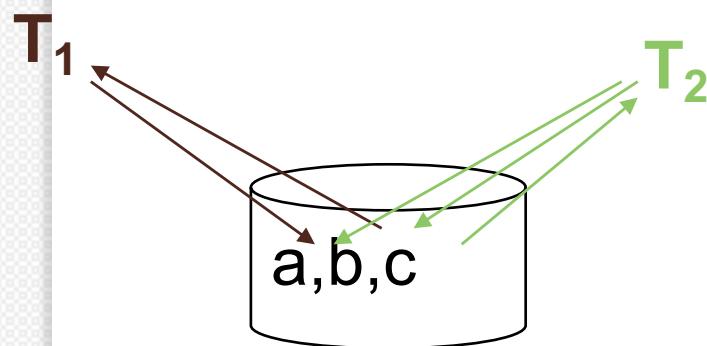
$S = r_2(A) \ r_1(B) \ w_2(A) \ r_3(C) \ w_1(B) \ w_3(A) \ r_2(B) \ w_2(B)$

$T = \{T_1, T_2, T_3\}$



Grafo acíclico S é serializável por conflito

Escalonamentos Corretos



$$T_1 = w_1(a) \ r_1(b)$$

$$T_2 = w_2(a) \ w_2(b) \ r_2(c)$$

$$S_{serial} = T_1 * T_2$$

$$S_{serial} = w_1(a) \ r_1(b) \ w_2(a) \ w_2(b) \ r_2(c)$$

$$S_{serial} = w_2(a) \ w_2(b) \ r_2(c) \ w_1(a) \ r_1(b)$$

$$S' = w_1(a) \ w_2(a) \ r_1(b) \ w_2(b) \ r_2(c)$$

$$S'' = w_1(a) \ w_2(a) \ w_2(b) \ r_1(b) \ r_2(c)$$

**Quais
escalonamentos são
corretos ?**

Serializabilidade

Escalonamentos corretos

$$S_{serial} = T_1 * T_2$$

$$S_{serial} = w_1(a) \ r_1(b) \ w_2(a) \ w_2(b) \ r_2(c)$$

$$S_{serial} = w_2(a) \ w_2(b) \ r_2(c) \ w_1(a) \ r_1(b)$$



$$S' = w_1(a) \ w_2(a) \ r_1(b) \ w_2(b) \ r_2(c)$$



Escalonamento correto!

Serializabilidade

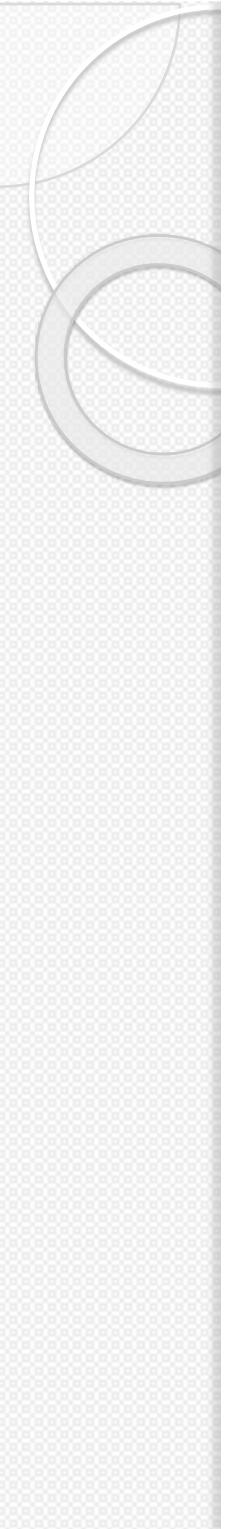
Escalonamentos corretos

$T_1 \xrightarrow{} T_2$

$S'' = w_1(a) w_2(a) w_2(b) r_1(b) r_2(c)$

$T_2 \xrightarrow{} T_1$

Escalonamento não é correto!



Serializabilidade

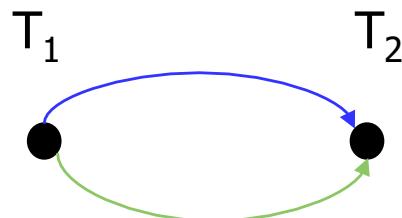
Grafo de Serializabilidade

- Grafo de serialização de um escalonamento S
 - Definido para um conjunto $T = \{T_1, T_2, \dots, T_n\}$
 - Grafo direcionado
 - $G(N, A)$, onde
 - N são as transações
 - $T_i \rightarrow T_j \in A$
 $\exists p \in OP(T_i), q \in OP(T_j)$, tal que **p conflita com q** e $p <_S q$

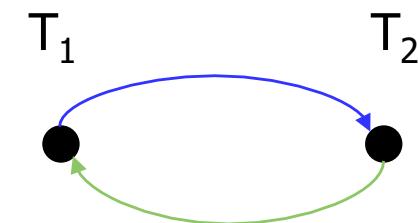
Serializabilidade

Grafo de Serializabilidade

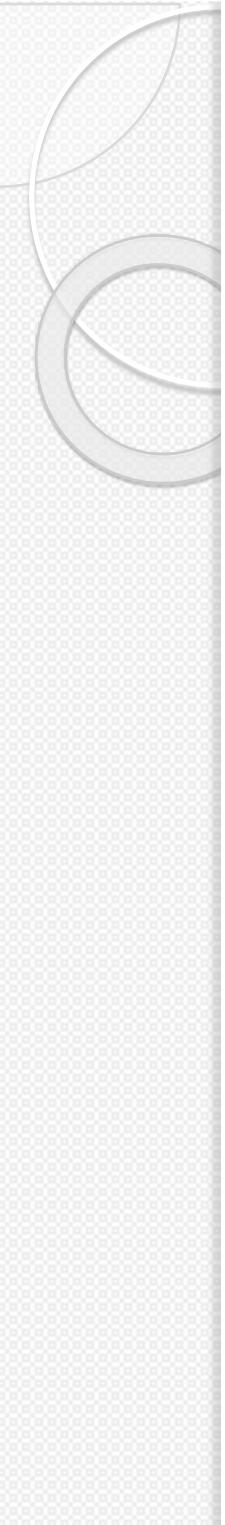
$S' = w_1(a) \ w_2(a) \ r_1(b) \ w_2(b) \ r_2(c)$ $S'' = w_1(a) \ w_2(a) \ w_2(b) \ r_1(b) \ r_2(c)$



**Grafo acíclico
Escalonamento correto !**



**Grafo com ciclo.
Escalonamento NÃO é correto !**

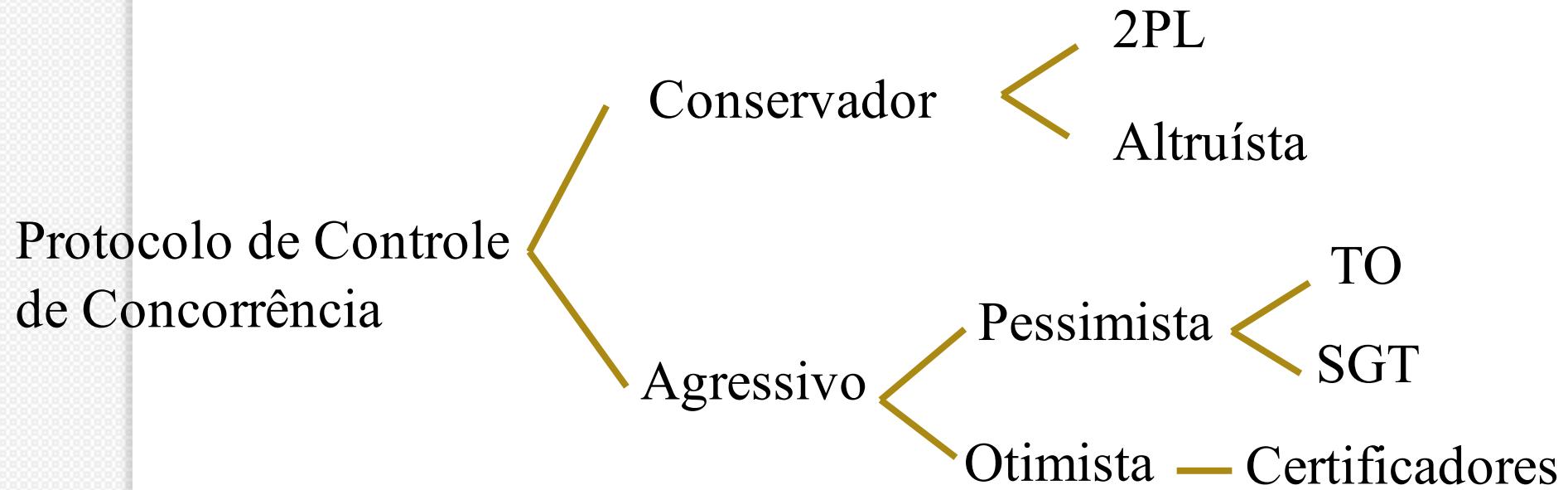


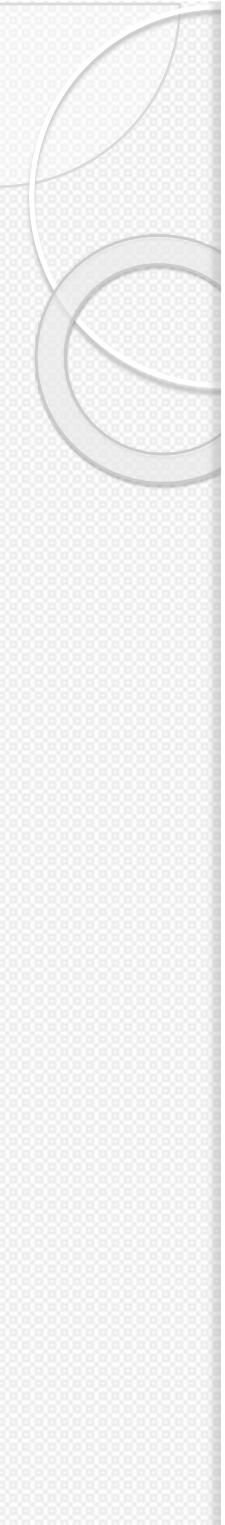
Exercício

S é serializável por conflito?

S=r₃(y)r₁(y)r₁(x)r₂(x)w₁(y)r₂(y)w₂(x)r₃(x)w₃(z)

Controle de Concorrência





Protocolos para o Controle de Concorrência

- **Conservadores**
 - Baseados em mecanismos de bloqueio
 - Atrasam a execução de operações para sincroniza-las corretamente
- **Agressivos**
 - Sincronizam operações imediatamente
 - ➡ Pessimistas
 - ⇒ Decidem se aceitam ou rejeitam a operação
 - ⇒ Se rejeitam, abortam a transação
 - ➡ Otimista
 - ⇒ Aceitam a operação imediatamente
 - ⇒ Periodicamente verificam a corretude do schedule já produzido

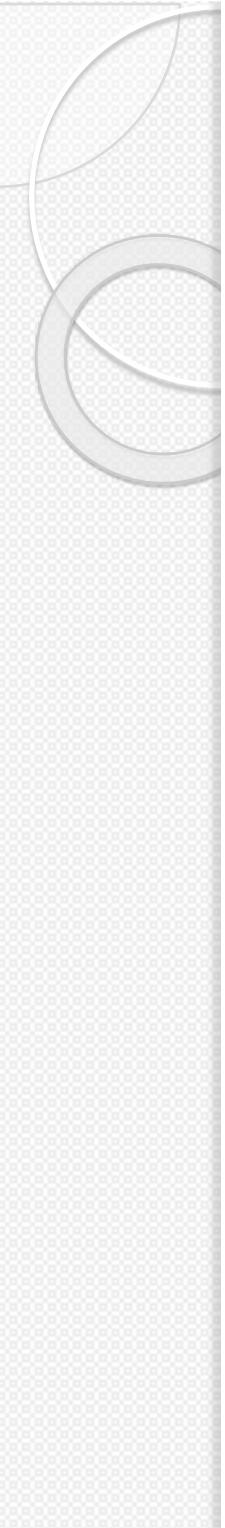
2PL (two-Phase Locking)

- Bloqueio → garantia de algum tipo de restrição ao acesso a um item de dado.
- Os dois tipos básicos de bloqueio de um item de dado são :
 - Compartilhado (rl - *read lock*)
 - Exclusivo (wl - *write lock*)
- Matriz de compatibilidade de bloqueio:

	rl	wl
rl	+	-
wl	-	-

Transação *i*
(solicitante)

Transação *j*
(retém o bloqueio)



2PL – Regras Básicas

- **Protocolo Básico**

1. Seja $p_i \in \{r_i, w_i\}$. Para conceder um bloqueio do tipo $pl_i(x)$:
O scheduler verifica se existe um **bloqueio incompatível** com $pl_i(x)$.
Se existir, então
o scheduler **atrasa** a execução da operação $p_i(x)$,
até que $pl_i(x)$ seja concedido.

2. Uma vez uma transação T_i libere algum bloqueio, T_i não pode obter mais bloqueios sobre qualquer objeto do DB

- **As duas fases do 2PL:**

- **Expansão (ou crescimento):** obtém os bloqueios.
- **Encolhimento:** liberam os seus bloqueios.

Obedece ao 2PL?

$$T_1 = r_1(x) \; w_1(y)$$

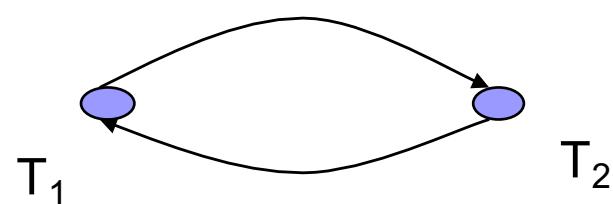
$$T_2 = w_2(x) \; w_2(y)$$

$$S = rl_1(x) \; r_1(x) \; ru_1(x) \; wl_2(x) \; w_2(x) \; wl_2(y) \; w_2(y) \; wu_2(x) \\ wu_2(y) \; wl_1(y) \; w_1(y) \; wu_1(y)$$

Obedece o 2PL ?

Não, uma vez que $ru_1(x) < wl_1(y)$

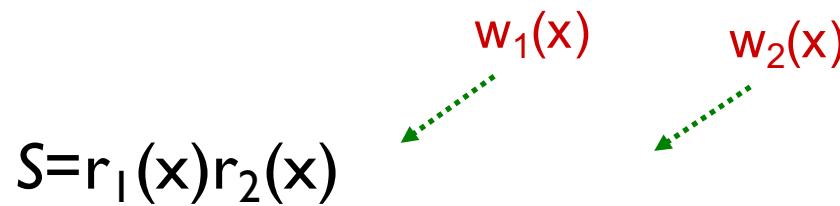
SG



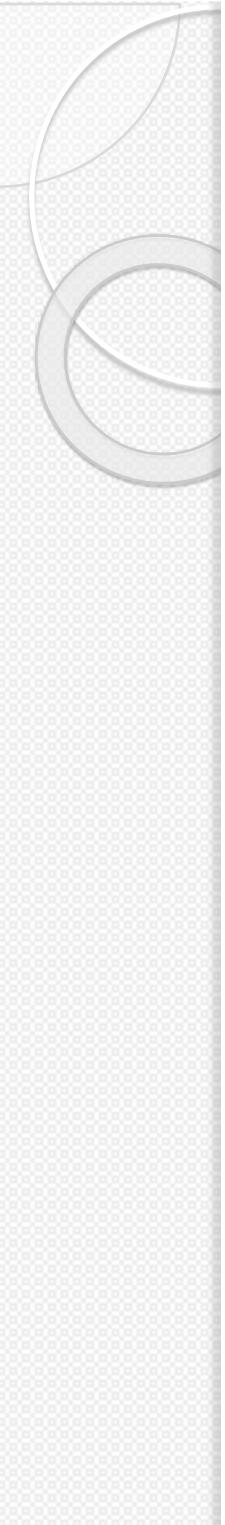
SG tem ciclo!

Protocolos Baseado em Bloqueio

$$T_1 = r_1(x) w_1(x) c_1 \quad T_2 = r_2(x) w_2(x) c_2$$

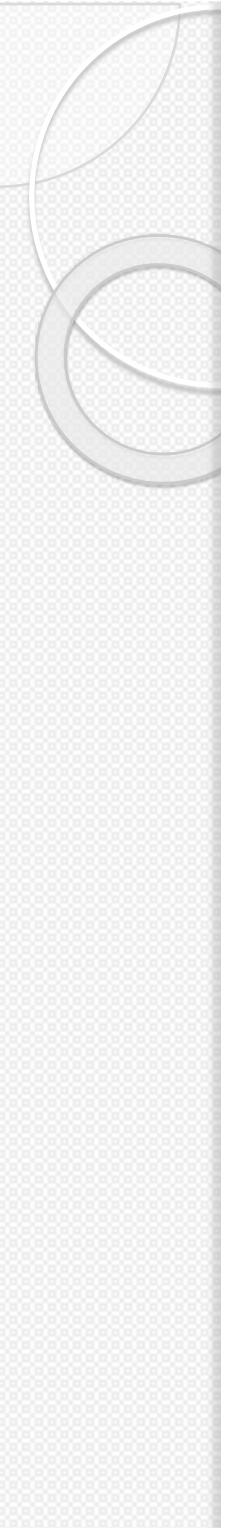


- ⇒ Transação T_1 espera que T_2 libere o bloqueio de leitura sobre x
- ⇒ Transação T_2 espera que T_1 libere o bloqueio de leitura sobre x
- ⇒ **Deadlock**



2PL - *Deadlock*

- **Timeout**
- **WFG (Wait-For-Graph)**
 - Os nós são as transações e
 - Uma aresta $Ti \rightarrow Tj$ estará presente, se e somente se, a transação Ti estiver esperando uma liberação de bloqueio de uma transação Tj
 - Um ciclo nesse grafo identifica um *deadlock*.
- **Descobrir um *deadlock* → abortar a transação**



Gerenciamento de Transações

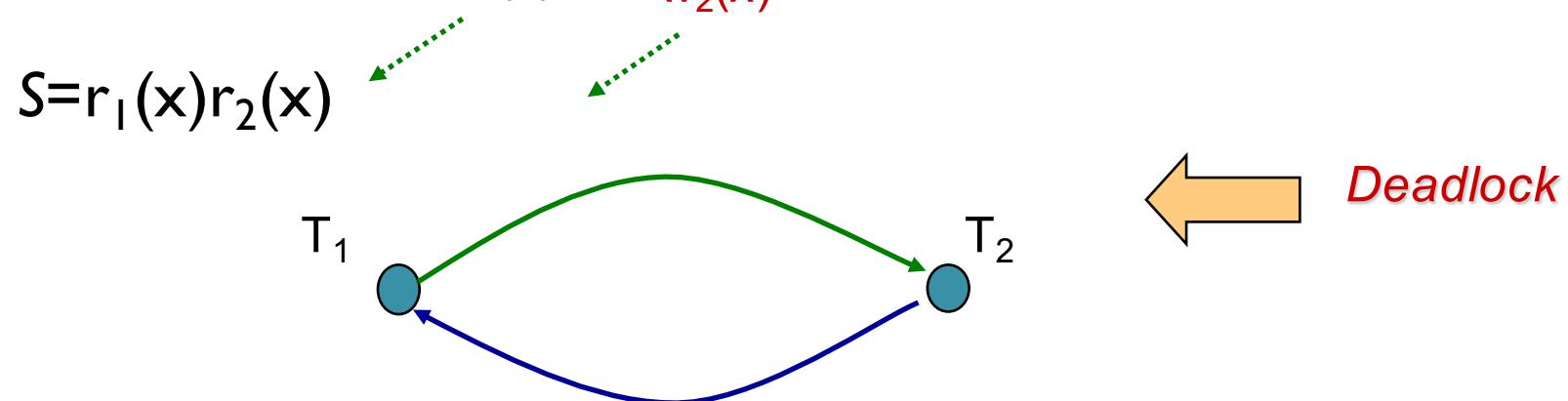
2PL - *Deadlock*

- A escolha da transação vítima envolve a análise dos seguintes aspectos:
 - A quantidade de esforço que já tenha sido gasto pela transação.
 - O número de atualizações que esta transação já realizou.
 - A quantidade de esforço para finalizar uma transação.
 - O número de ciclos que contém a transação.

Grafo de Espera

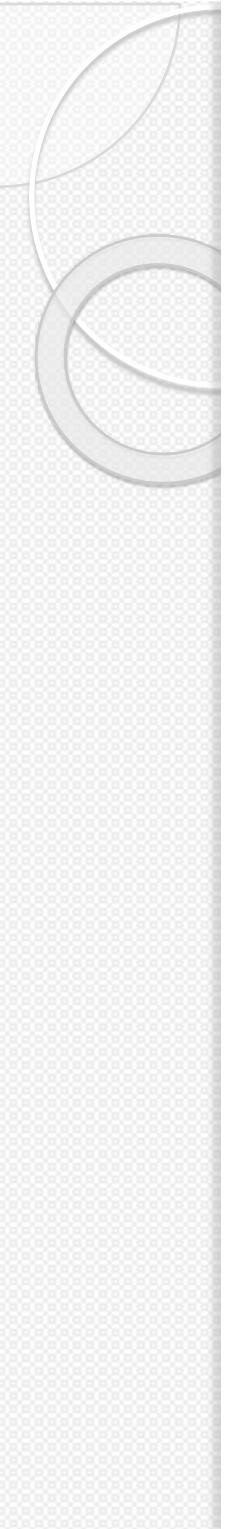
- Detecção de *Deadlocks*
 - ➡ *Grafos de Espera* (cont.)

$$T_1 = r_1(x)w_1(x)c_1 \quad T_2 = r_2(x)w_2(x)c_2$$



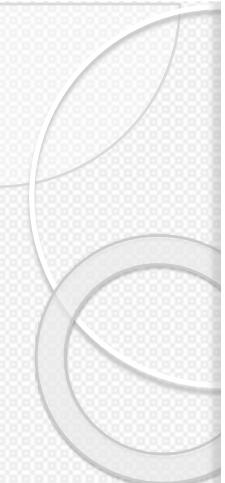
Resolução

➡ T_2 (mais recente) deve ser abortada (**vítima**)



Protocolo de ordenação por marca de tempo TO (Timestamp Ordering)

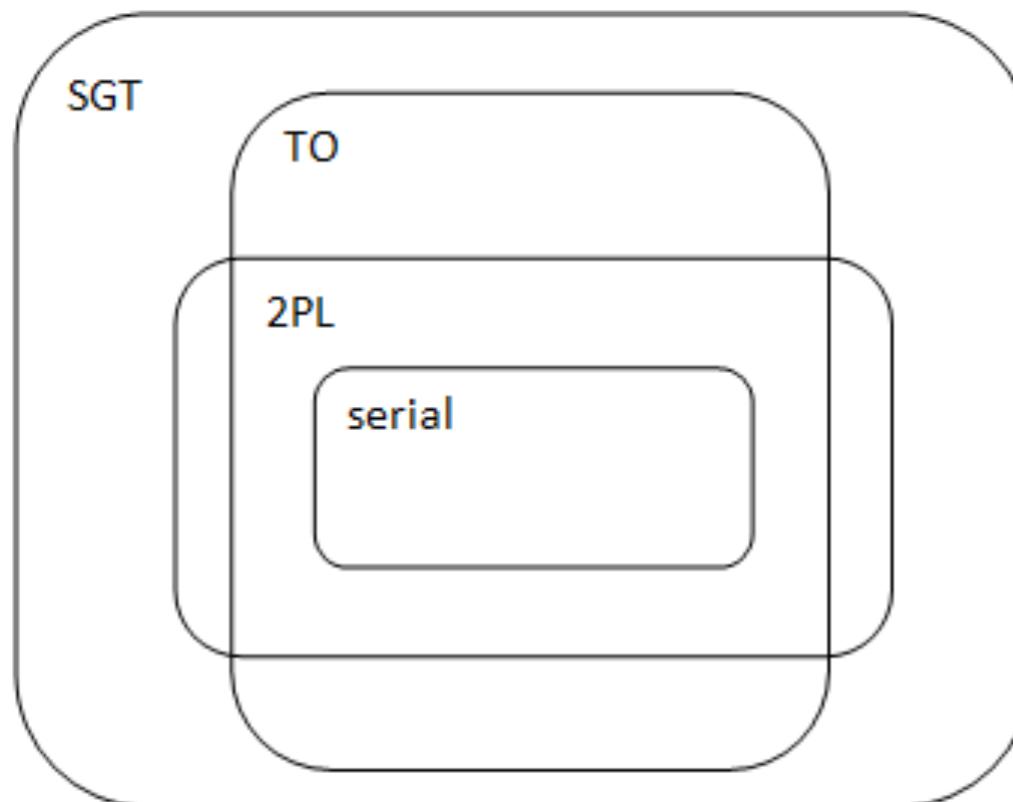
- Não usa bloqueio → não tem *deadlocks*.
- Marca de tempo para cada transação T_i , e as operações de leitura e escrita para cada item de dado.
- A regra básica do TO é a seguinte:
 - Se $pi(x)$ e $qj(x)$ são operações conflitantes, então o gerenciador de dados processa $pi(x) <_s qj(x)$ se e somente se $ts(Ti) < ts(Tj)$.

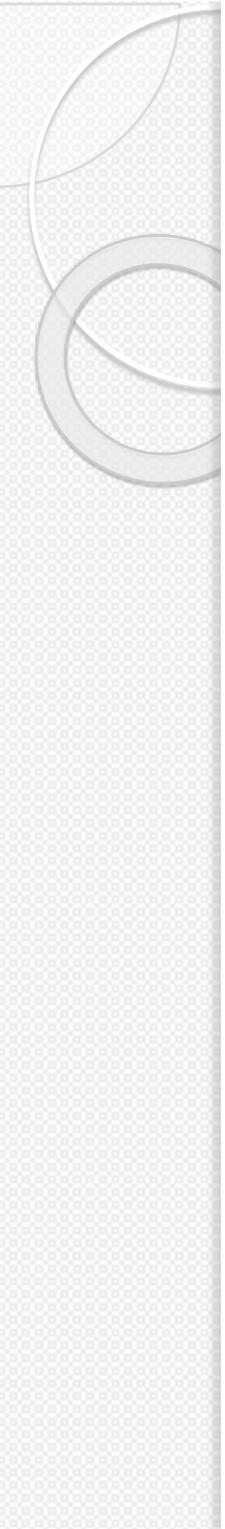


Teste de Grafo de Serialização SGT (*Serialization Graph Testing*)

- Grafo de serialização (SG), representando as transações em execução.
 - O nó representa T_i .
 - Uma aresta é adicionada de T_j para T_i , quando uma operação $q_j(x)$ conflita com $p_i(x)$.

Diagrama Venn

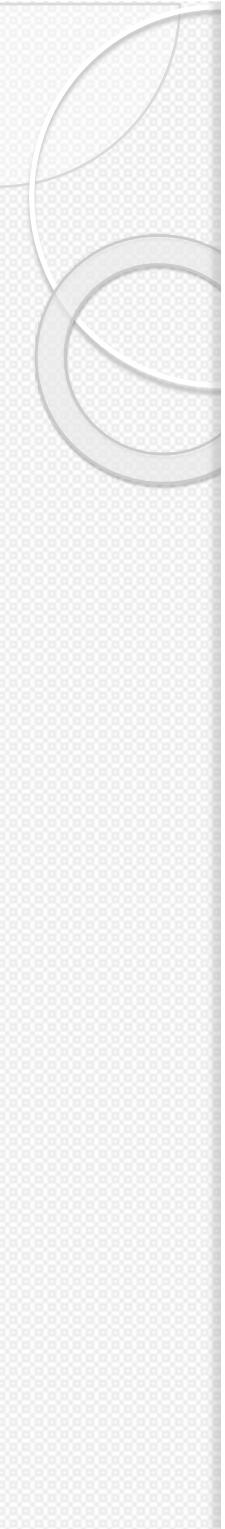




Gerenciamento de Transações

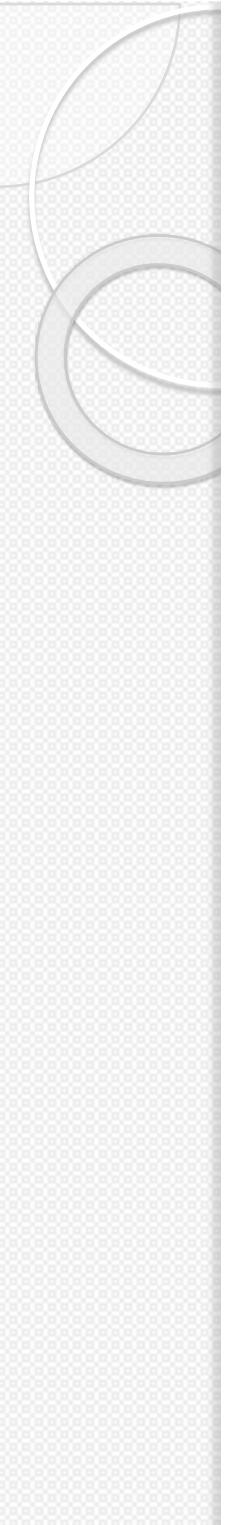
Protocolos Otimistas

- Também conhecidas por técnicas de validação ou certificação, nenhuma verificação é feita enquanto a transação está sendo executada.
- Três fases distintas podem ser especificadas nesse modelo :
 1. **Fase de leitura:** nessa fase se inicia, a execução da transação T_i . Variáveis locais temporárias.
 2. **Fase de validação:** A transação T_i processa um teste de validação.
 3. **Fase de escrita:** Se a transação T_i obtém sucesso na validação (passo 2), então a atualização é aplicada de fato ao banco de dados. Caso contrário, T_i é desfeita



CheckList

- Transações
- Escalonamentos
 - Corretos, Serial e Serializáveis
 - Grafo de Serializabilidade
 - Equivalência de Escalonamento
- Controle de Concorrência
 - 2PL, TO e SGT
 - Protocolos Otimistas
- Diagrama de Venn



Bibliografia

- Capítulos 17 e 18 do livro texto
ELMASRI, R., NAVATHE, S. B., Sistemas de Banco de Dados, Quarta Edição, 2005, Editora Addison Wesley.