BD2 - BCC 2023/2

Concorrência - **parte 2** - 05/09/2023 - Hylson Elmasri 6a edição (português)

22.1.1 Outros protocolos que impedem *deadlock*

- no waiting (sem espera)
- cautions waiting (espera cuidadosa)

22.1.1 Outros protocolos que impedem *deadlock*

- no waiting (sem espera)
  - se não conseguir um bloqueio, aborta e reinicia após certo tempo

22.1.1 Outros protocolos que impedem *deadlock*

- no waiting (sem espera)
  - se não conseguir um bloqueio, aborta e reinicia após certo tempo
  - **pode provocar abortos e reinícios desnecessários**

22.1.1 Outros protocolos que impedem *deadlock*

- cautions waiting (espera cuidadosa)
  - Ti tenta bloquear X, mas não consegue por causa de Tj

22.1.1 Outros protocolos que impedem *deadlock*

- cautions waiting (espera cuidadosa)
  - Ti tenta bloquear X, mas não consegue por causa de Tj
  - Se Tj não estiver bloqueada (esperando algum item), então Ti pode esperar
    - Caso contrário, aborte Ti

22.1.1 Outros protocolos que impedem *deadlock*

- detecção de deadlock
  - bom em caso de pouca concorrência

22.1.1 Outros protocolos que impedem *deadlock*

- detecção de deadlock
  - bom em caso de pouca concorrência
    - transações pequenas
    - poucos bloqueios

22.1.1 Outros protocolos que impedem *deadlock*

- mecanismo:
  - grafo de espera

22.1.1 Outros protocolos que impedem *deadlock*

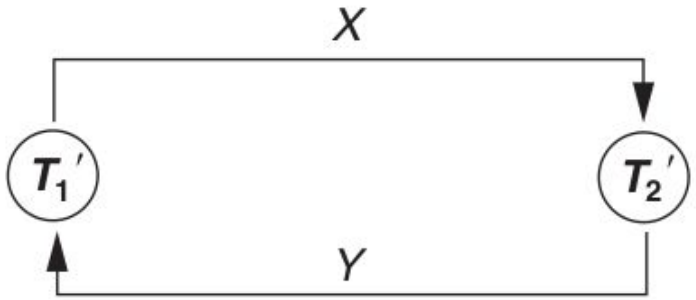
- grafo de espera

(a)

T_1'	T_2'
read_lock(Y);	
read_item(Y);	
	read_lock(X);
	read_item(X);
write_lock(X);	
	write_lock(Y);

Tempo

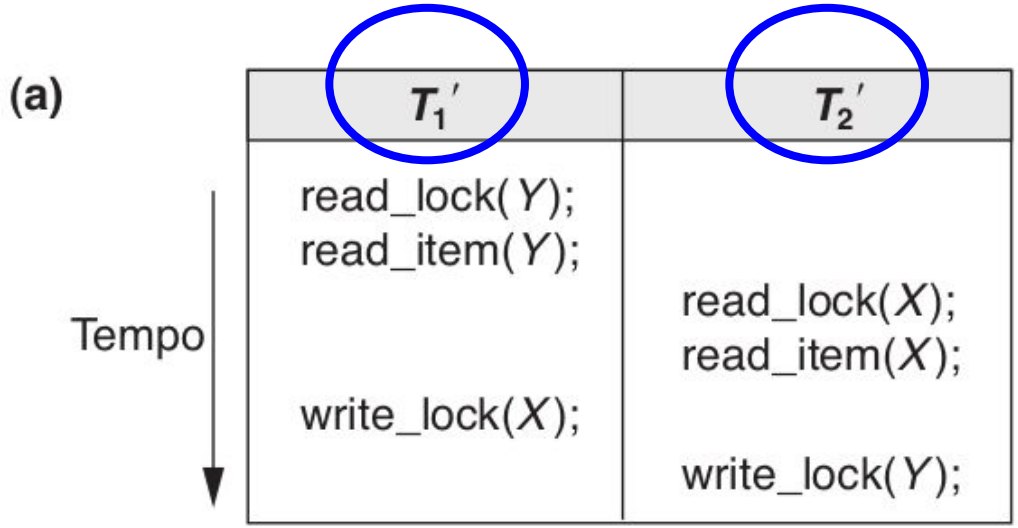
(b)

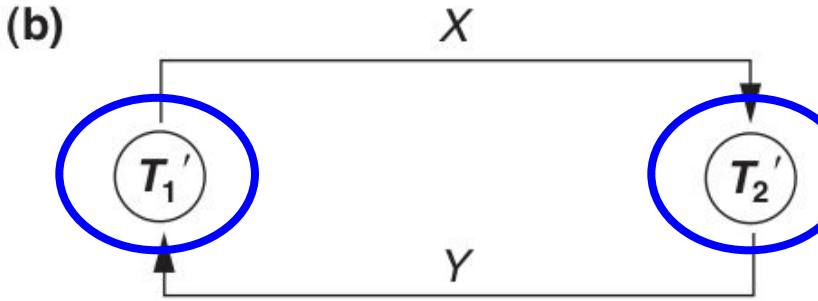


22.1.1 Outros protocolos que impedem *deadlock*

- grafo de espera

um nó para cada transação executada





22.1.1 Outros protocolos que impedem *deadlock*

- grafo de espera

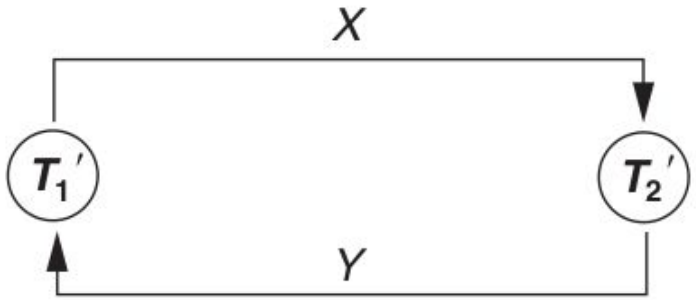
Ti esperando bloquear X que está bloqueado por Tj: criar ***aresta Ti -> Tj***

(a)

T_1'	T_2'
read_lock(Y);	
read_item(Y);	
	read_lock(X);
	read_item(X);
write_lock(X);	
	write_lock(Y);

Tempo

(b)



22.1.1 Outros protocolos que impedem *deadlock*

- grafo de espera

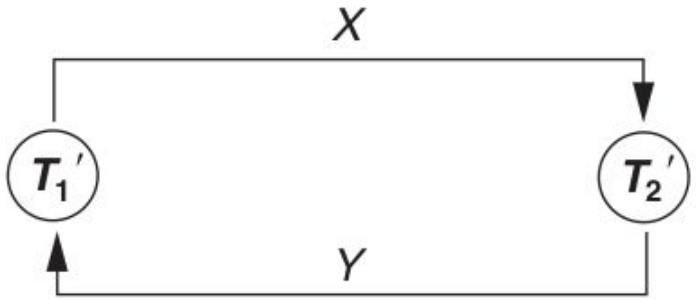
Ti esperando bloquear X que está bloqueado por Tj: criar ***aresta Ti -> Tj***

(a)

T_1'	T_2'
read_lock(Y);	
read_item(Y);	
	read_lock(X);
	read_item(X);
write_lock(X);	
	write_lock(Y);

Tempo

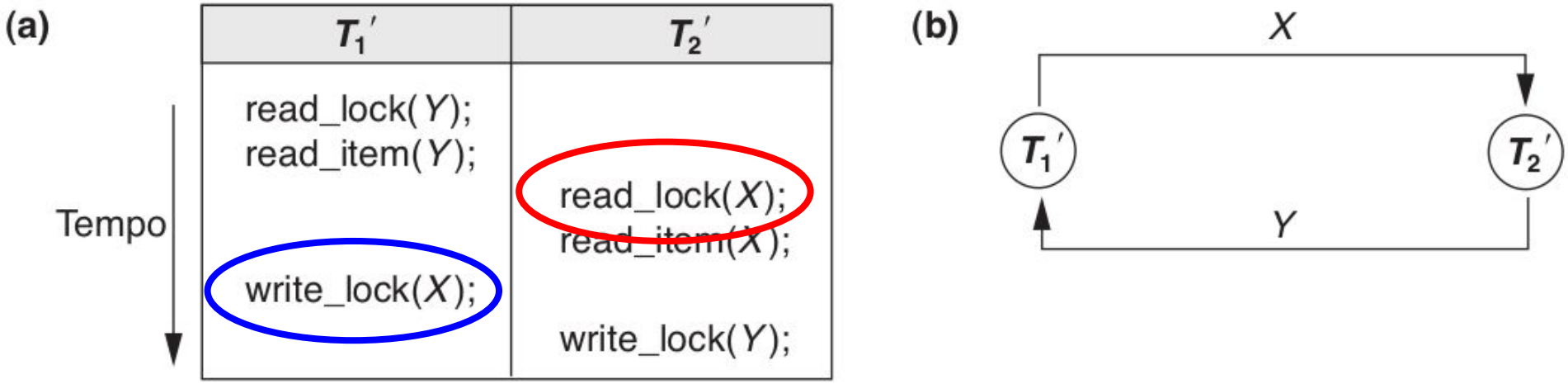
(b)



22.1.1 Outros protocolos que impedem *deadlock*

- grafo de espera

Ti esperando bloquear X que está bloqueado por Tj: criar ***aresta Ti -> Tj***

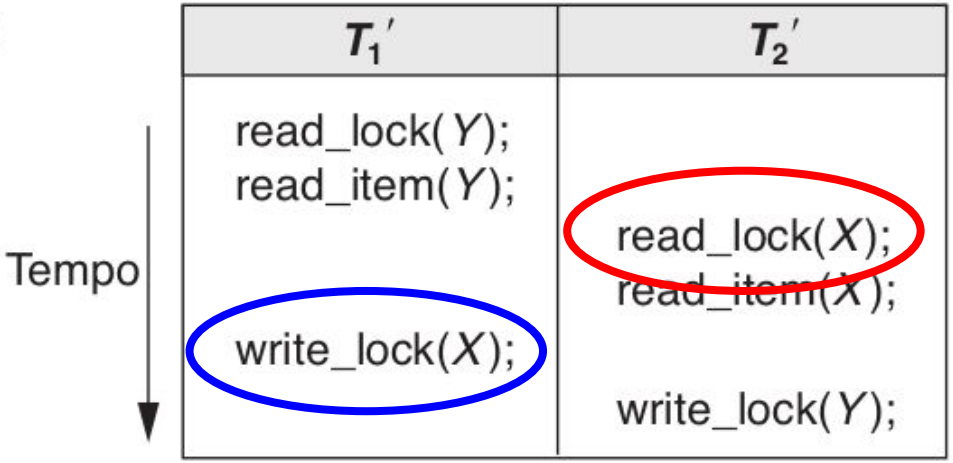


22.1.1 Outros protocolos que impedem *deadlock*

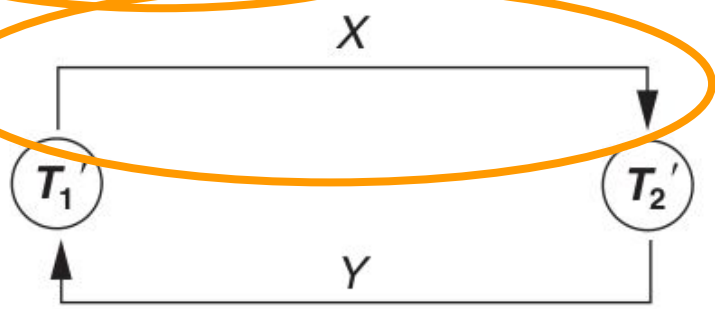
- grafo de espera

Ti esperando bloquear X que está bloqueado por Tj: criar ***aresta Ti -> Tj***

(a)



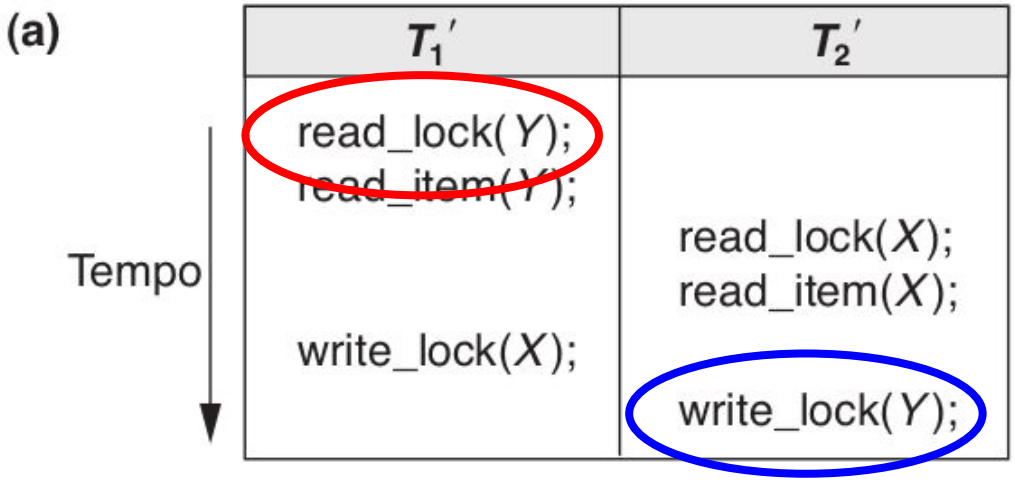
(b)

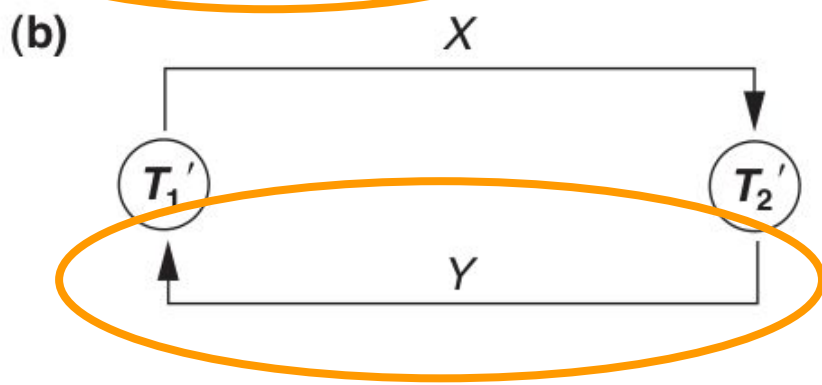


22.1.1 Outros protocolos que impedem *deadlock*

- grafo de espera

Ti esperando bloquear X que está bloqueado por Tj: criar ***aresta Ti -> Tj***

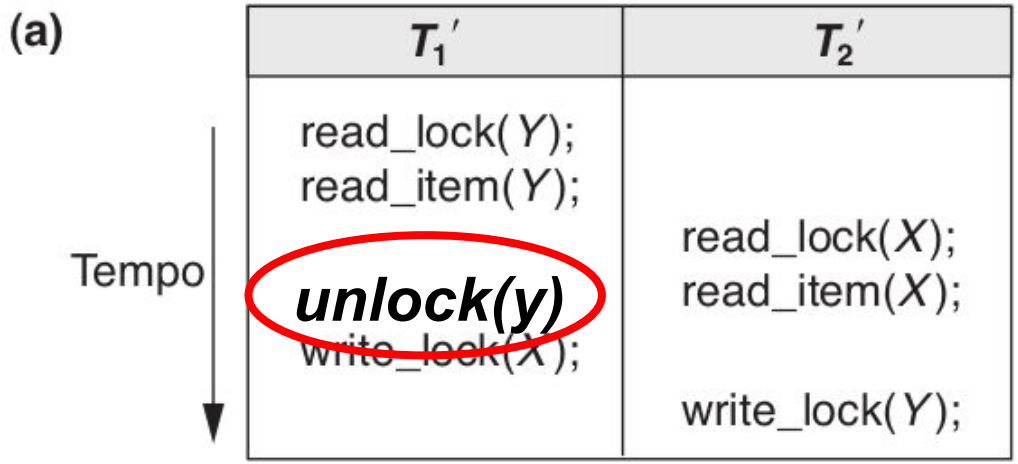


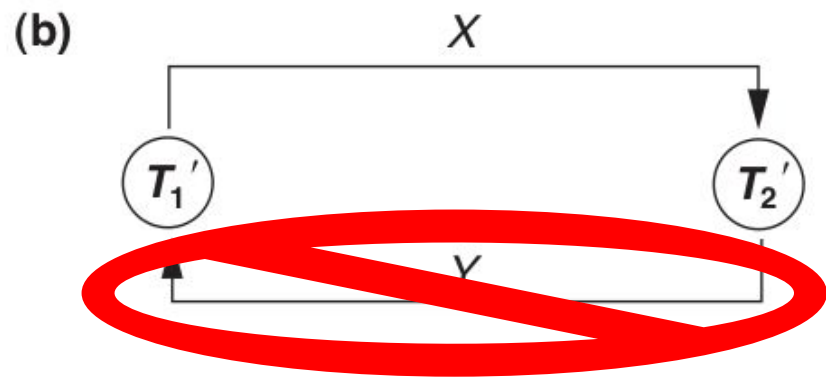


22.1.1 Outros protocolos que impedem *deadlock*

- grafo de espera

quando Tj libera o bloqueio
=> aresta é removida





22.1.1 Outros protocolos que impedem *deadlock*

- grafo de espera

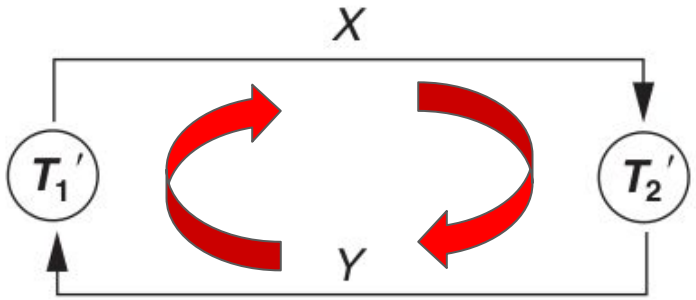
detecção de deadlock:
existência de ***CICLO***

(a)

T_1'	T_2'
read_lock(Y);	
read_item(Y);	
	read_lock(X);
	read_item(X);
write_lock(X);	
	write_lock(Y);

Tempo

(b)

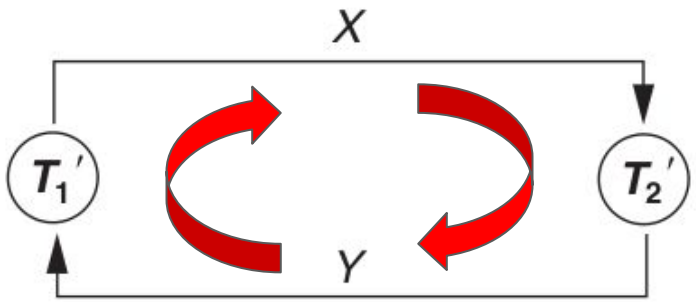


22.1.1 Outros protocolos que impedem *deadlock*

- grafo de espera

problema: quando realizar essa verificação de existência de ciclo?

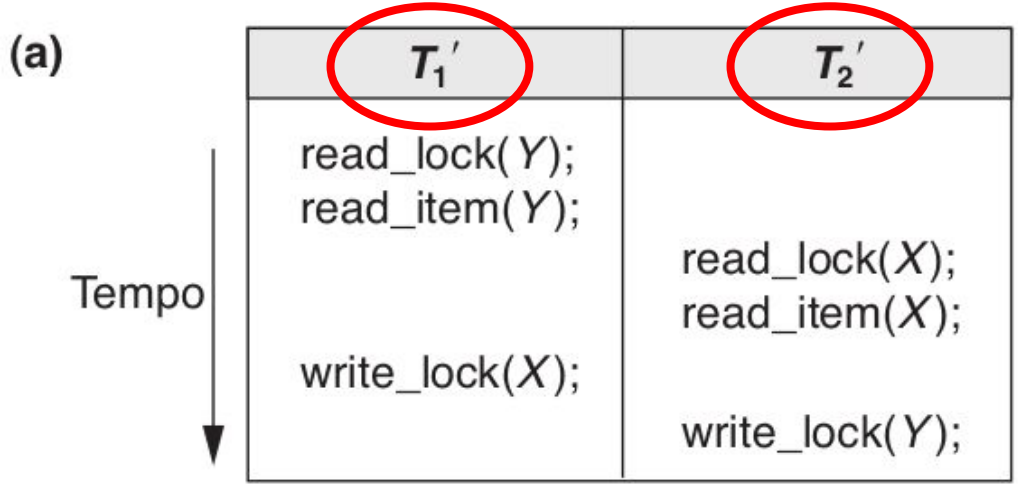
- *quando surgir aresta no grafo?*
- *quando houver muitas (n=?) transações executando?*
- *quando tempo de espera for grande (maior que x=?)*



22.1.1 Outros protocolos que impedem *deadlock*

- grafo de espera

*havendo deadlock: escolher uma **vítima** para ser abortada*



22.1.1 Outros protocolos que impedem *deadlock*

- timeout

mecanismo prático:
espera pelo bloqueio,
se não conseguir…

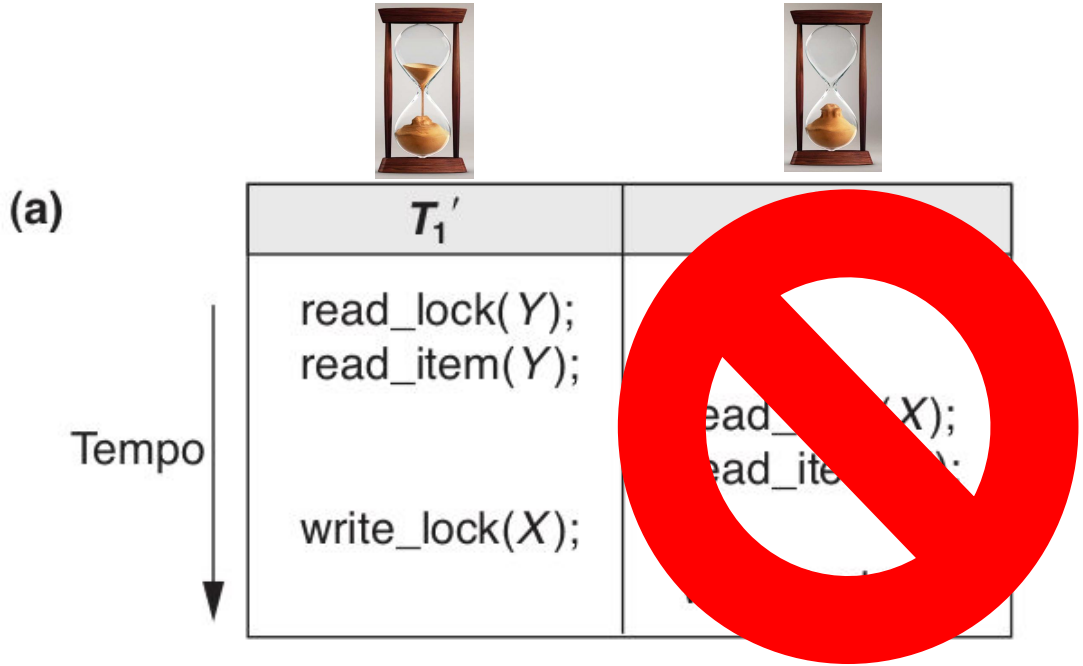
(a)

T_1'	T_2'
read_lock(Y);	
read_item(Y);	
	read_lock(X);
	read_item(X);
write_lock(X);	
	write_lock(Y);

Tempo

22.1.1 Outros protocolos que impedem *deadlock*

- timeout



mecanismo prático:
espera pelo bloqueio,
se não conseguir,
aborta!

22.1.1 Outros protocolos que impedem *deadlock*

- inanição (starvation)

  T não consegue prosseguir por tempo indefinido… enquanto outras transações continuam!

22.1.1 Outros protocolos que impedem *deadlock*

- inanição (starvation)

solução: esquema justo de espera

FILA

MAIOR PRIORIDADE CONFORME TEMPO DE ESPERA

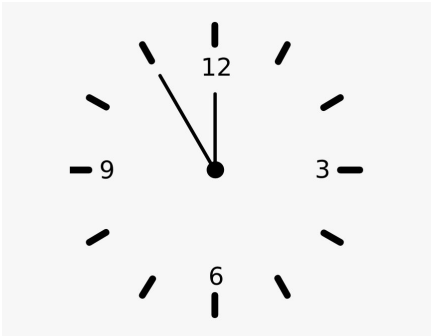
- wait and die
- wound-wait

22.2 - Controle de concorrência baseado na ordenação de rótulo de tempo (***timestamp***)

22.2 - timestamp

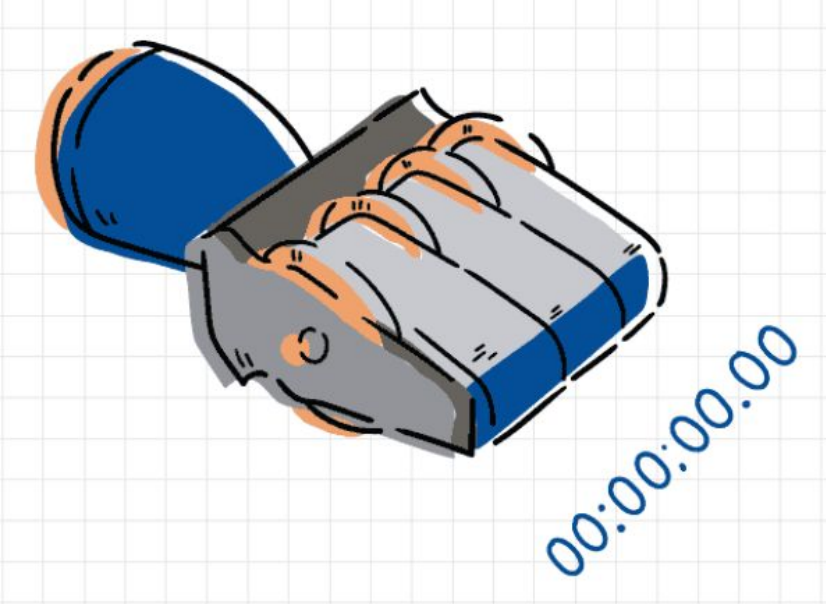
bloqueios + 2PL:
garante serialização
via ordem de
bloqueios

outra estratégia:
ordem **temporal**



22.2.1 timestamp

identificador exclusivo criado pelo SGBD para identificar uma transação



22.2.1 timestamp

atribuído na ordem em que as transações são submetidas

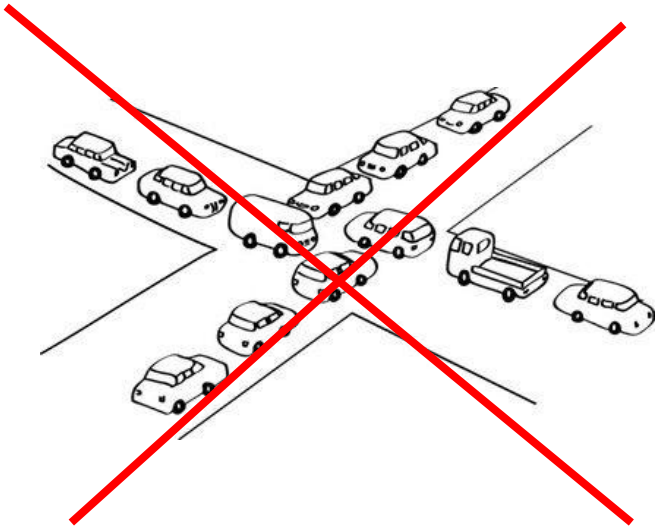
TS(T)

22.2.1 timestamp

atribuído na ordem em que as transações são submetidas

TS(T)

não usam bloqueios :-)



sem deadlocks!

22.2.1 timestamp

geração de timestamp:

contador incremental

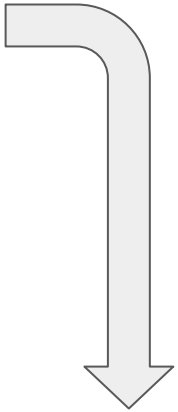
data/hora do sistema (batida do clock)

23-02-2012 9:18:12

Unix Timestamp 1329968892

22.2.1 timestamp

ordenação com
base no *timestamp*



implica em…

único schedule serial equivalente
permitido: ordem dos timestamps (TO)

22.2.1 timestamp

valores utilizados:

read_TS(X): maior TS das transações que leram X

write_TS(X): maior TS das T’s que gravaram em X

22.2.1 timestamp

ordenação básica TO:

1. T tenta executar write_item(X)

22.2.1 timestamp

ordenação básica TO:

1. T tenta executar write_item(X)

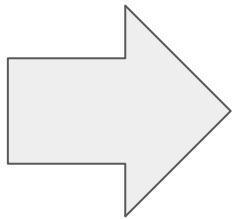
a) se read_TS(X) > TS(T)
   ou
   write_TS(X) > TS(T)

22.2.1 timestamp

ordenação básica TO:

1. T tenta executar write_item(X)

a) se read_TS(X) > TS(T)
ou
write_TS(X) > TS(T)



aborte e reverta T

rejeite a operação

(alguma T mais recente leu ou gravou X)

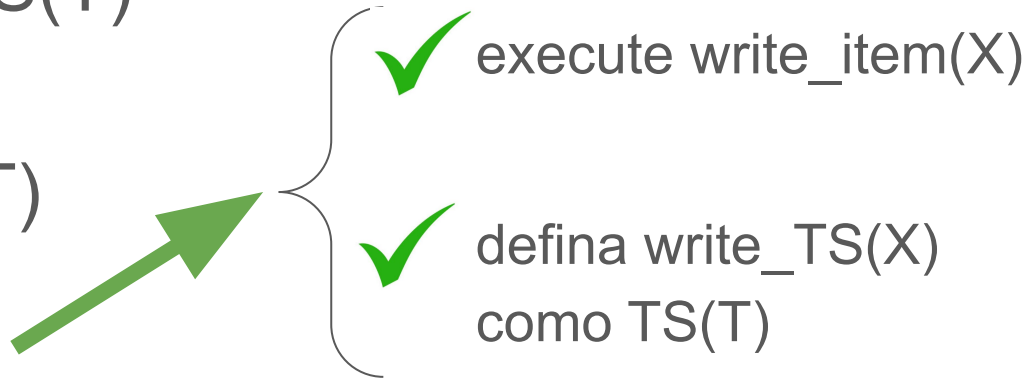
22.2.1 timestamp

ordenação básica TO:

1. T tenta executar write_item(X)

a) se read_TS(X) > TS(T)
ou
write_TS(X) > TS(T)

b) caso contrário…



22.2.1 timestamp

ordenação básica TO:

2) T tenta executar read_item(X)

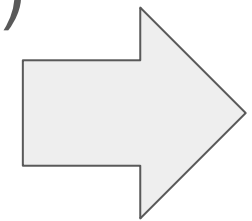
a) se write_TS(X) > TS(T)

22.2.1 timestamp

ordenação básica TO:

2) T tenta executar read_item(X)

a) se write_TS(X) > TS(T)



aborte e reverta T

rejeite a operação

(alguma T' mais recente gravou X antes de T ler)

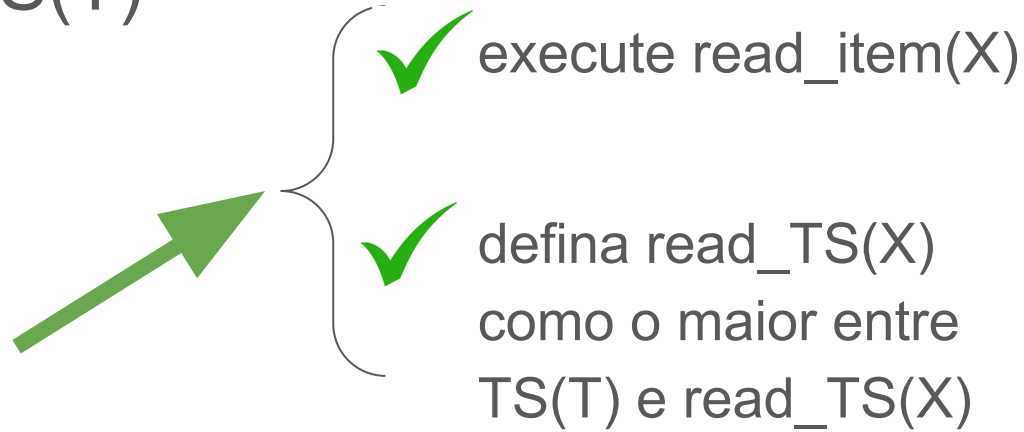
22.2.1 timestamp

ordenação básica TO:

2) T tenta executar read_item(X)

a) se write_TS(X) > TS(T)

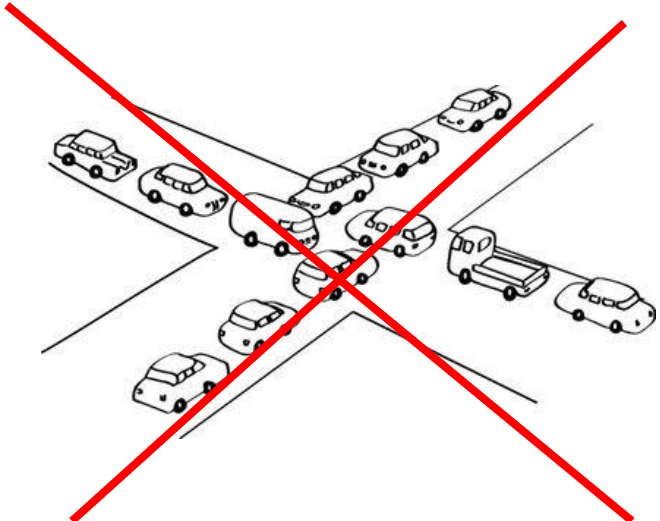
b) caso contrário…



22.2.1 timestamp

ordenação básica TO:

não ocorrem deadlocks



pode ocorrer inanição

22.2.1 timestamp

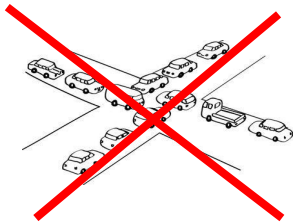
ordenação TO estrita

T emite read_item(X) ou write_item(X)

e

TS(T) > write_TS(X)

22.2.1 timestamp



ordenação TO estrita

não causa deadlock :-)
pois T só espera T' se…

T emite read_item(X) ou write_item(X)

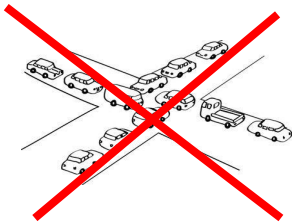
e

TS(T) > write_TS(X)

adia T até que T' (que gravou X) confirme ou aborte

(simular bloqueio de X até T' confirmar ou abortar)

22.2.1 timestamp



ordenação TO estrita

não causa deadlock :-)
pois T só espera T' se…

T emite read_item(X) ou write_item(X)

e

TS(T) > write_TS(X)

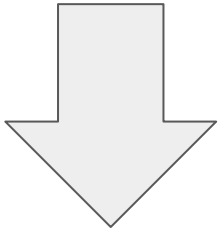
adia T até que T' (que gravou X) confirme ou aborte

(simular bloqueio de X até T' confirmar ou abortar)

T espera T', mas T' não espera T

22.3 Técnicas de CC multiversão

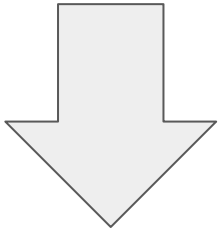
*mantém valores **antigos** quando dados são atualizados*



ao solicitar acesso, uma versão “apropriada” é escolhida para manter a serialização do schedule

22.3 Técnicas de CC multiversão

*mantém valores **antigos** quando dados são atualizados*



ao solicitar acesso, uma versão “apropriada” é escolhida para manter a serialização do schedule

=> *requer mais espaço de armazenamento*
=> *similar a um tempo de dados temporal*

22.4 Técnicas otimistas (validação)

nenhuma verificação é feita antes ou depois




22.4 Técnicas otimistas (validação)

nenhuma verificação é feita antes ou depois

atualizações são feitas a cópia locais

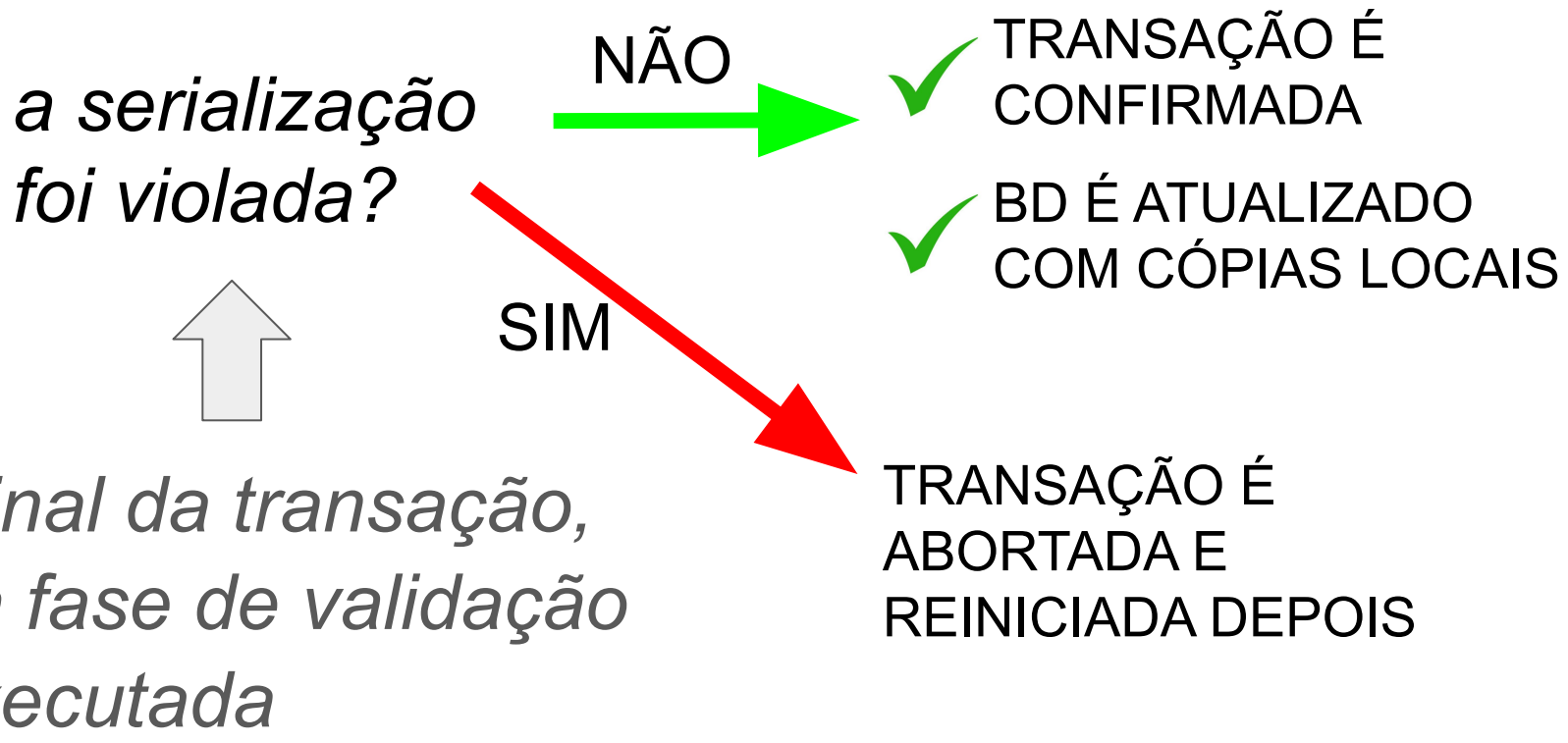
22.4 Técnicas otimistas (validação)

nenhuma verificação é feita antes ou depois

atualizações são feitas a cópia locais

ao final da transação, uma fase de validação é executada

22.4 Técnicas otimistas (validação)



22.4 Técnicas otimistas (validação)

FASES:

1. leitura: ler valores dos itens de T, atualizar em cópias locais
2. validação
3. gravação

22.4 Técnicas otimistas (validação)

FASES:

1. leitura: ler valores dos itens de T, atualizar em cópias locais
2. validação
3. gravação

pouca interferência: transações validadas com sucesso

muita interferência: muitos abortos e reinícios

22.4 Técnicas otimistas (validação)

***Validação de Ti**: para cada Tj confirmada ou em validação, uma condição abaixo deve ser mantida:*

1. Tj completa a gravação antes de Ti iniciar a leitura

22.4 Técnicas otimistas (validação)

***Validação de Ti**: para cada Tj confirmada ou em validação, uma condição abaixo deve ser mantida:*

1. Ti inicia a gravação depois que Tj completa a gravação, **e** read-set de Ti não tem itens em comum com write-set de Tj
2.

22.4 Técnicas otimistas (validação)

***Validação de Ti**: para cada Tj confirmada ou em validação, uma condição abaixo deve ser mantida:*

1. read-set e write-set de Ti não tem itens em comum com write-set de Tj, **e** Tj completa leitura antes de Ti
2.
3.

22.4 Técnicas otimistas (validação)

***Validação de Ti**: para cada Tj confirmada ou em validação, uma condição abaixo deve ser mantida:*

1. *se nenhuma das 3*
2. *condições for mantida, Ti é*
3. *abortada e reiniciada depois*

22.5 granularidade

Item pode ser…

um registro
um valor de campo de um registro
um bloco de disco
um arquivo inteiro
um BD inteiro

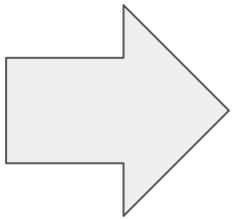
22.5 granularidade

Granularidade fina:
itens pequenos

Granularidade grossa:
itens grandes

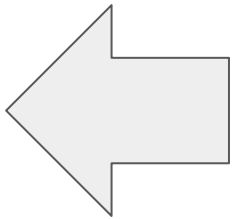
22.5 granularidade

Granularidade fina: itens pequenos



maior o número de itens, maior o número de bloqueios a controlar, maior *overhead*

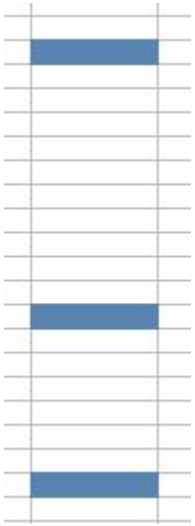
menor o nível de concorrência permitido



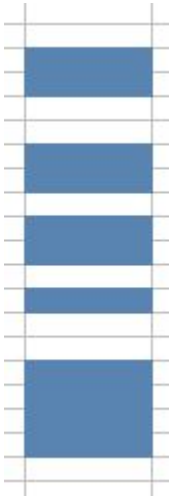
Granularidade grossa: itens grandes

22.5 granularidade

Tamanho ideal de cada item? Depende



acessa **poucos** registros? vantagem: granularidade como registro



acessa **muitos** registros? vantagem: granularidade como bloco ou arquivo

22.7.2 transações interativas

Transações lêem e gravam valores de um dispositivo interativo, como uma tela de monitor, antes de confirmar

22.7.2 transações interativas

Transações lêem e gravam valores de um dispositivo interativo, como uma tela de monitor, antes de confirmar

***problema**: usuário pode fornecer valores em T com base em valor escrito na tela por T' que ainda não foi confirmado*

22.7.2 transações interativas

Transações lêem e gravam valores de um dispositivo interativo, como uma tela de monitor, antes de confirmar

mitigação: adiar exibição na tela até obter valores confirmados

***problema**: usuário pode fornecer valores em T com base em valor escrito na tela por T' que ainda não foi confirmado*

FIM “concorrência”