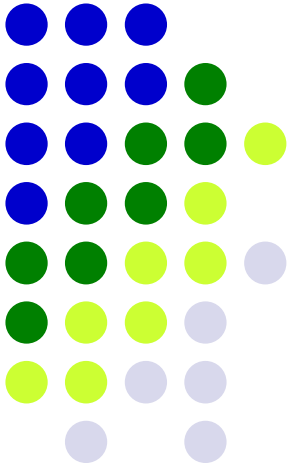
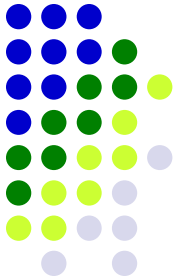
Controle de Concorrência

Sistemas de Banco de Dados

Prof. Antonio Guardado

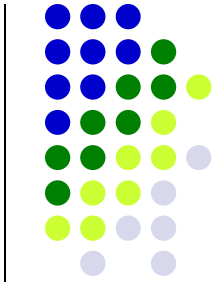


1. Problema



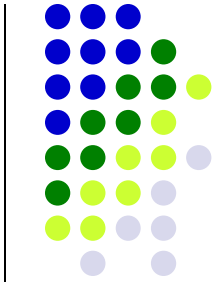
- Escalas não serializáveis podem violar a consistência do BD.
- Forçar um comportamento sequencial fazendo uma transação esperar que a outra execute determinadas operações.
  - Como ? Bloqueando o acesso aos dados que uma transação utiliza para as demais transações concorrentes.
  - A transação solicitará ao Subsistema de Controle de Concorrência o bloqueio dos dados.

1.1 - Protocolos com base em bloqueios (lock)



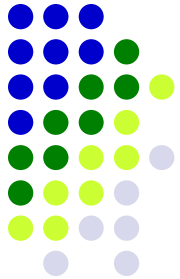
- Obrigar que o acesso a dados seja feito de maneira mutuamente exclusiva preservando o ISOLAMENTO
- Enquanto uma transação acessa um item de dados, nenhuma outra pode modificá-lo.
- As transações devem solicitar o bloqueio de modo apropriado, dependendo do tipo de operação realizada.
- O gerenciador de controle de concorrência concede (*grants*) o bloqueio para a transação (ela pode ter que esperar).

1.2 -Tipos de Bloqueios



- Compartilhado: Se uma transação Ti obteve um bloqueio compartilhado (denotado por S de *share*) sobre o item Q, então T*i* pode ler, mas não escrever Q
- Exclusivo: Se uma transação T*i* obteve um bloqueio exclusivo (denotado por X de *eXclusive*) sobre o item Q, então T*i* pode tanto ler como escrever Q

1.3 -Função de compatibilidade

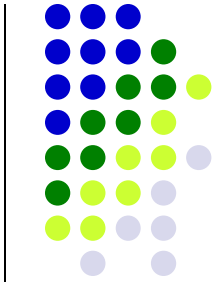


- Um elemento comp(A,B) da matriz possui valor verdadeiro se, e somente se, o modo A é compatível com o modo B

	S	*X*
S	verdade	falso
X	falso	falso

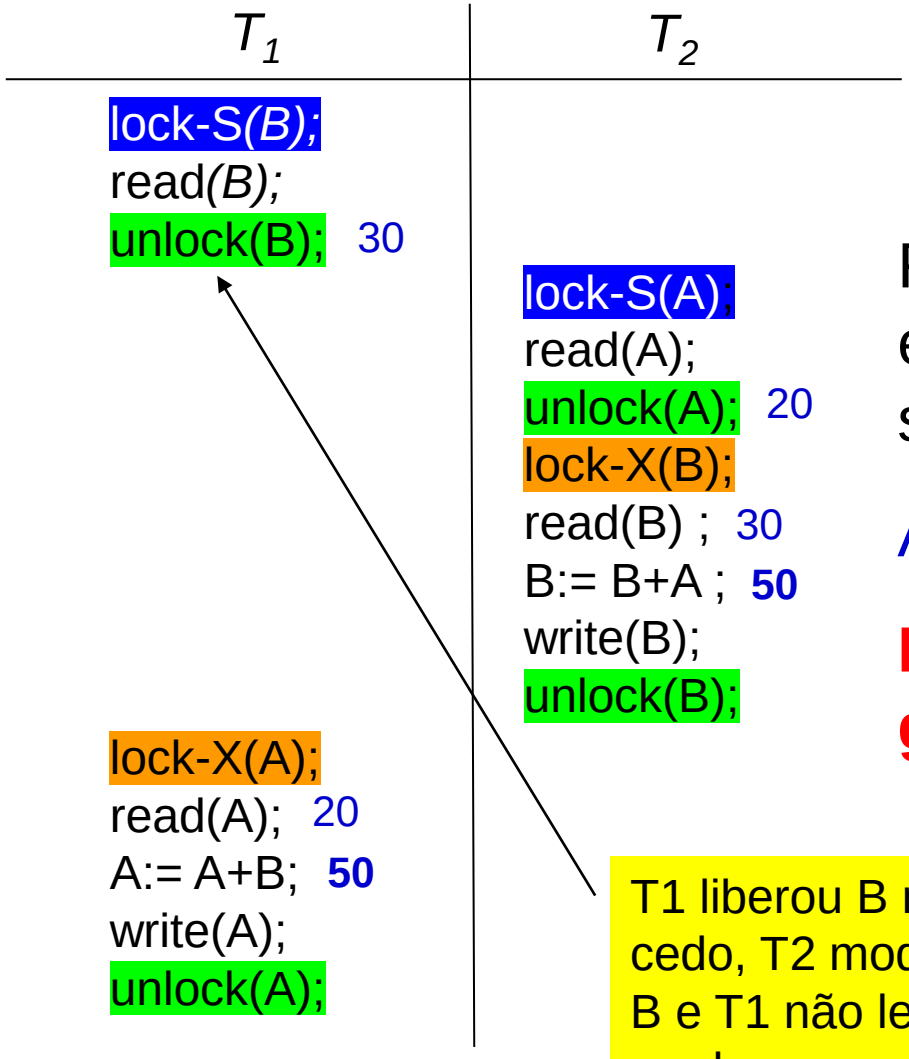
S = Share
X - EXclusive

1.4 -Exemplos de locks



- Instruções:
  - lock-S(Q) : bloqueio no modo compartilhado
  - lock-X(Q) : bloqueio no modo exclusivo
  - unlock(Q) : remove todos os bloqueios
- Para manter o acesso a um item de dado, a transação precisa primeiro bloqueá-lo. Se já estiver incompativelmente bloqueado, o gerenciador não concederá o bloqueio até que todos os bloqueios incompatíveis sejam desfeitos

1.4 - Exemplos

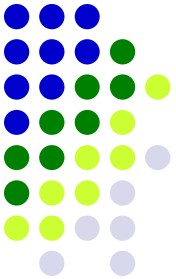


Resultado deste escalonamento não-serializável

A=50 e B=50

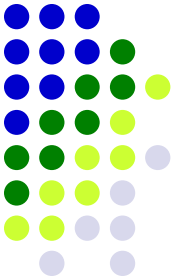
Bloqueios por si só não garantem a serialização

2 - Protocolo de bloqueio



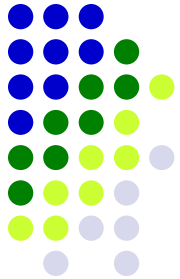
- Conjunto de regras que determina quando uma transação pode ou não bloquear ou desbloquear um item de dados
- Restringe o número de escalas possíveis (serializadas)

2.1 - Protocolo de bloqueio em duas fases (Two-Phase Locking ou 2PL)



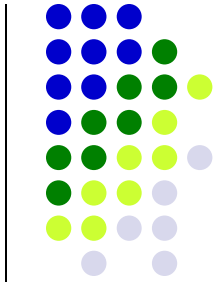
- Garante a serialização
- Exige que cada transação emita suas solicitações de bloqueio e desbloqueio em duas fases:
  1. Fase de expansão
  2. Fase de encolhimento

2.1.1 - Fases do 2PL



- Fase de expansão ou crescimento: uma transação pode obter bloqueios, mas não pode liberar nenhum
- Fase de encolhimento ou retrocesso: uma transação pode liberar bloqueios, mas não consegue obter nenhum bloqueio novo
- Não garante a ausência de deadlock
- Pode ocorrer o *rollback* em cascata

2.1.2 - Exemplo do 2PL



$T_{1'}$	$T_{2'}$
lock-S(*B*);	lock-S(*A*);
read(*B*);	read(*A*);
lock-X(*A*);	lock-X(*B*);
unlock(*B*);	unlock(*A*);
read(*A*);	read(*B*) ;
A:= A+B;	B:= B+A ;
write(*A*);	write(*B*);
unlock(*A*);	unlock(*B*);

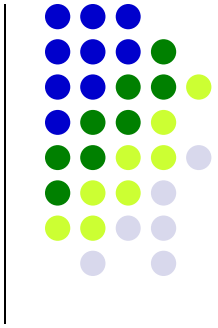
Expansão

Obtenção dos bloqueios

Encolhimento

Liberação dos bloqueios

2.1.2- Exemplo do 2PL



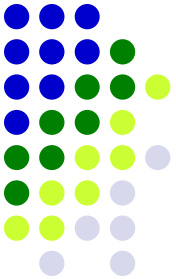
$T_{1'}$	$T_{2'}$
lock-S(*B*);	
read(*B*); 30	
lock-X(*A*);	
unlock(*B*);	
	lock-S(*A*);
	read(*A*); 50
	lock-X(*B*);
	unlock(*A*);
	read(*B*) ; 30
	B:= B+A ; 80
	write(*B*);
	unlock(*B*);
read(*A*); 20	
A:= A+B; 50	
write(*A*);	
unlock(*A*);	

2PL garante a serialização, pois T2' tem que esperar a liberação de A em T1' para executar

Resultado deste escalonamento serializável

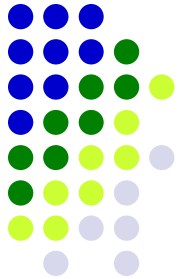
A=50 e B=80

2.1.3 – Variações do 2PC



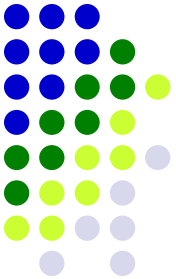
- Bloqueio em duas fases **severo**: exige que todos os bloqueios de modo exclusivo tomados por uma transação sejam mantidos até que a transação seja efetivada (commit)
  - Não permite rollback em cascata
- Bloqueio em duas fases **rigoroso**: exige que todos os bloqueios tomados por uma transação sejam mantidos até que a transação seja efetivada (commit)
  - As transações podem ser serializadas na ordem de efetivação

2.2 - Protocolo de *commit* em duas fases Distribuído



- *Two-phase commit protocol:* **2PC**
- A ação de *commit* deve ser “instantânea” e indivisível.
- Pode ser necessária a cooperação de muitos processos, em máquinas distintas, cada qual com um conjunto de objetos envolvidos na transação.
- Um dos processos é designado como **coordenador** (normalmente o próprio cliente que inicia a transação).
- Os demais processos são designados como **participantes.**
- Toda ação é registrada em *log,* armazenado em *memória estável*, para o caso de falha durante o protocolo.

2.2.1 - Fases do 2PC Distribuído



- **Fase 1: Votação**
  - O coordenador envia mensagem VOTE_REQUEST para todos os participantes e aguarda as respostas.
  - Cada participante responde VOTE_COMMIT ou VOTE_ABORT para o coordenador.
- **Fase 2: Decisão**
  - Se todos os participarem tiverem respondido VOTE_COMMIT, o coordenador envia para todos os participantes um GLOBAL_COMMIT senão envia um GLOBAL_ABORT.
  - Cada participante confirma ou aborta a sua transação local, conforme receba GLOBAL_COMMIT ou GLOBAL_ABORT, respectivamente.

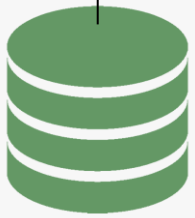
Fase 1 - Votação

Vote_Commit

Brasília

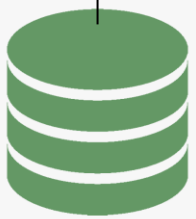
Empregados de Brasília,
Empregados de São Paulo

Projetos de Brasília



Vote_Commit

São Paulo



Empregados de São Paulo,
Empregados de Brasília

Projetos de São Paulo,
Projetos de Brasília

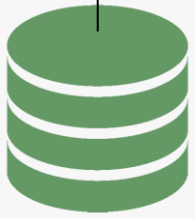
Projetos de Porto Alegre

Rede de Comunicação

Porto Alegre

Empregados de Porto Alegre

Projetos de Porto Alegre,
Projetos do Rio de Janeiro



Coordenador

Rio de Janeiro

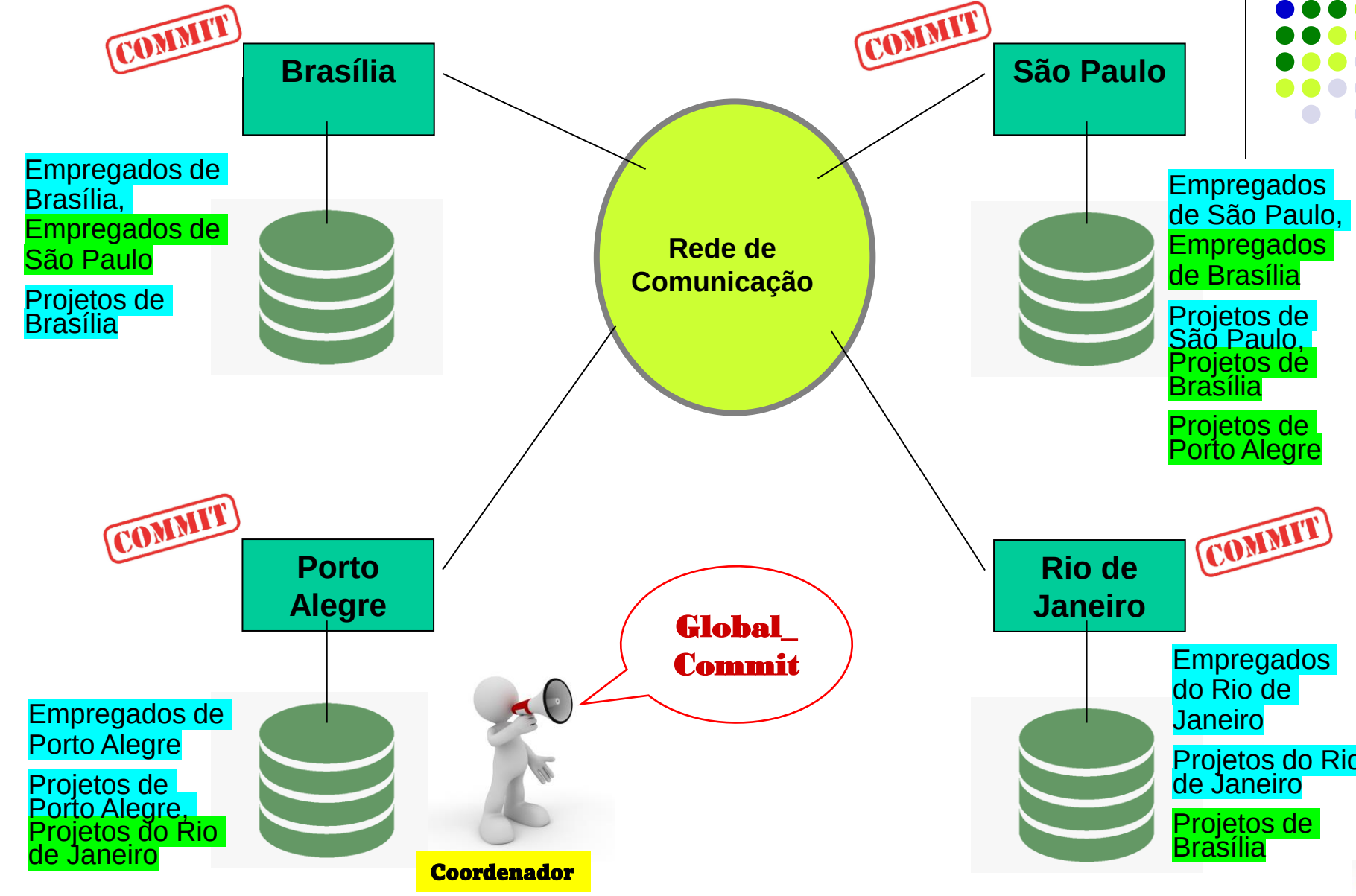
Vote_Commit

Empregados do Rio de Janeiro

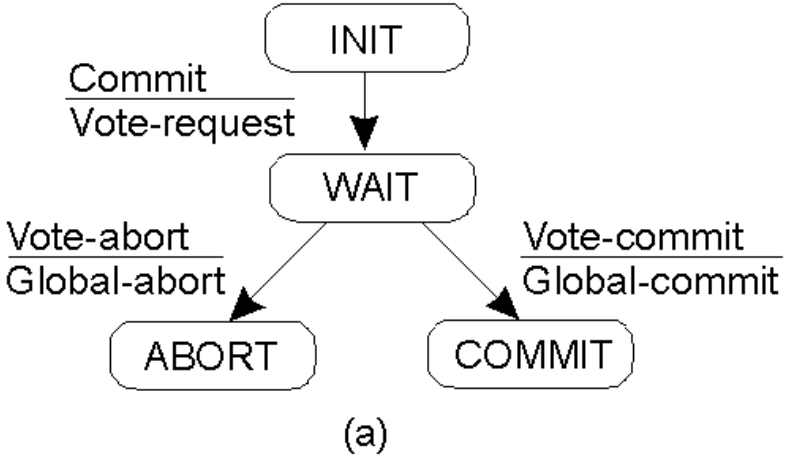
Projetos do Rio de Janeiro

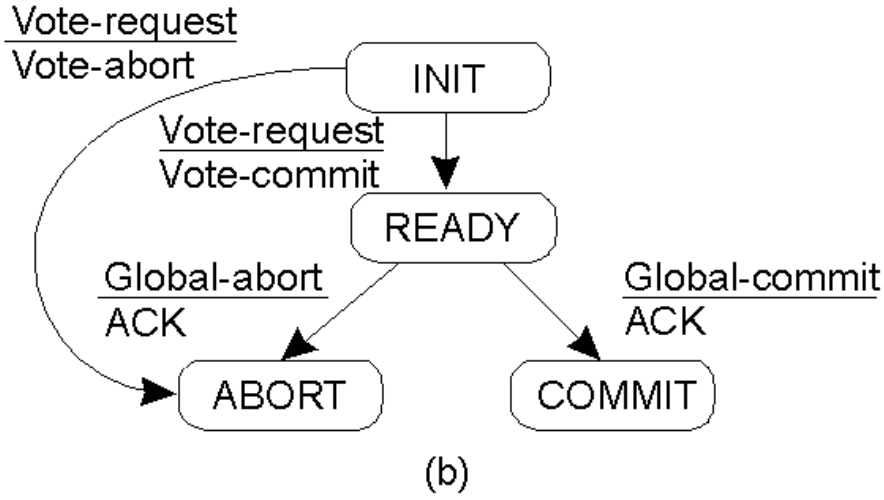
Projetos de Brasília

Fase 2 - Decisão



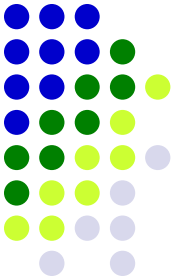
2.2.1 - Fases do 2PC Distribuído





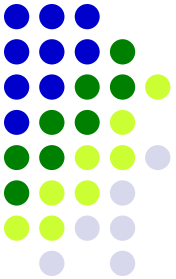
a) Máquina de estados do coordenador
b) Máquina de estados de cada participante

2.2.2 - Falhas durante o 2PC Distribuído



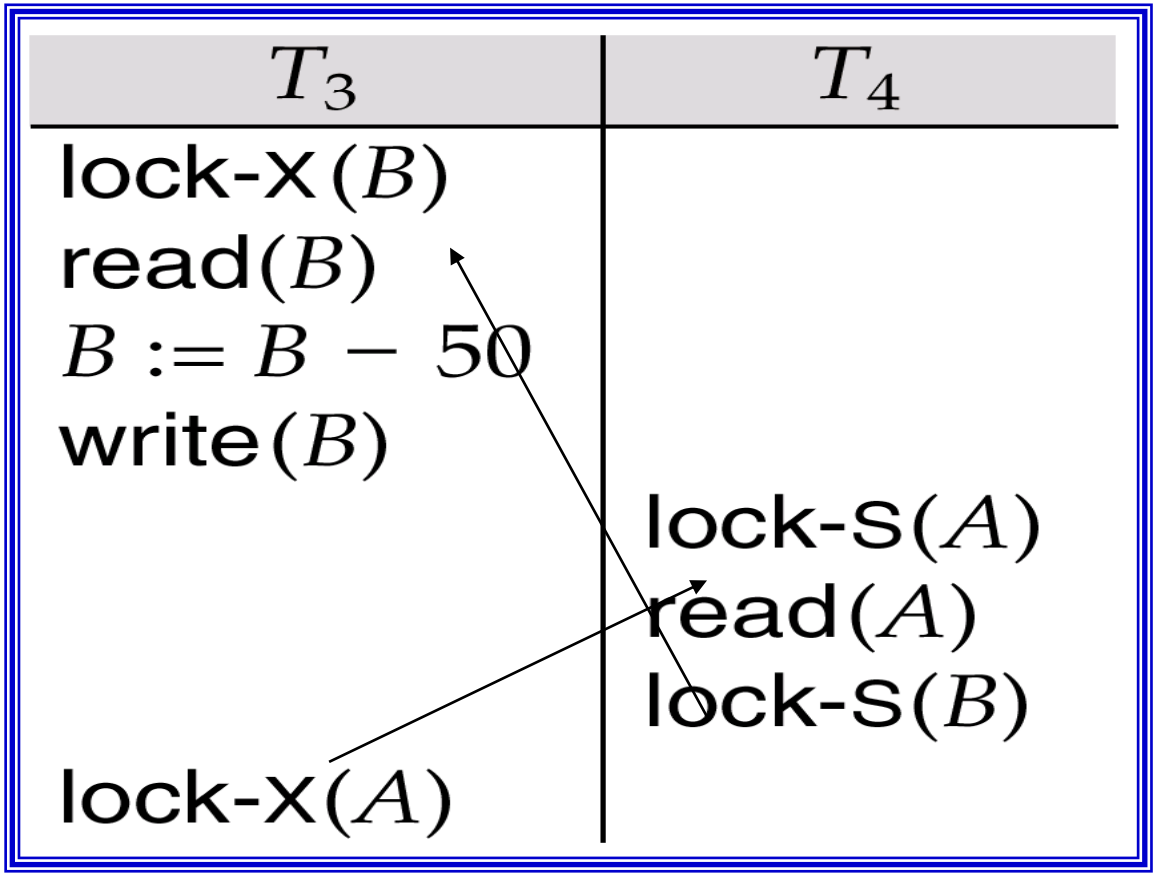
- Coordenador pode ficar bloqueado em WAIT, aguardando os votos dos participantes
  - Decide por *abort* se todos os votos não chegarem dentro de um certo tempo.
- Participante pode ficar bloqueado em INIT, aguardando um VOTE_REQUEST do coordenador
  - Vota por *abort* se o VOTE_REQUEST não chegar dentro de um certo tempo.

2.2.2 - Falhas durante o 2PC Distribuído



- Participante pode ficar bloqueado em READY, aguardando a decisão do coordenador
  - Se a decisão não chegar dentro de um certo tempo, consulta outros participantes para descobrir qual foi a decisão.
  - Caso um participante consultado esteja bloqueado em INIT, a decisão é por *abort*.
  - Caso todos os participantes estejam bloqueados em READY, o protocolo bloqueia até que o coordenador se recupere.

3 - Deadlock (impasse)



T4 espera T3 desbloquear B

T3 espera T4 desbloquear A para prosseguir

Uma das transações tem que ser desfeita (rollback)