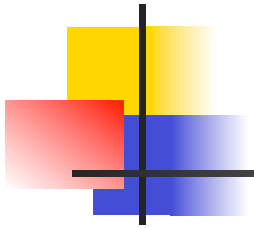


Banco de Dados II

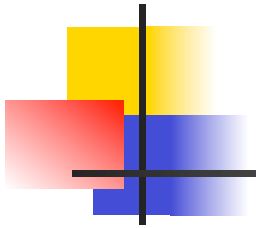
Gerenciamento de Transações

Gerenciamento de Transações

Reserva de Passagem Aérea

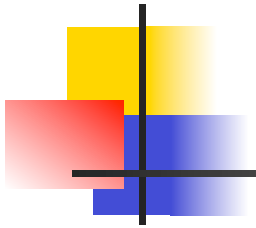
```
begin_transaction Reserva_Voo
Begin
   input(Voo_n,Dia,Passageiro_Nome);
   EXEC SQL select lugares_vendidos, capacidade into temp1, temp2
                 from VOO
                where VNO = Voo_n and DATA =        Dia;
  if temp1 = temp2  then
     output('Não há lugares disponíveis.'); ROLLBACK
  else
     EXEC SQL update VOO
                   set lugares_vendidos = lugares_vendidos + 1
                   where VNO = Voo_n and DATA = dia;
     EXEC SQL insert into VOO(VNO,DATA,NOME_PASSAGEIRO,ESPECIAL)
                   values (Voo_n,Dia,Passageiro_Nome,null);
     COMMIT
     output('Reserva concluída.');
  endif
end
```

Gerenciamento de Transações



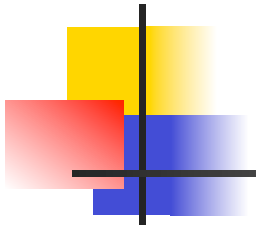
- **Controle de Concorrência**
- **Recuperação**

Gerenciamento de Transações



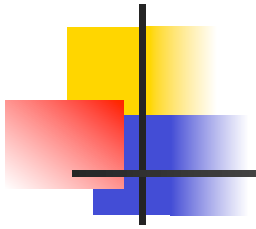
- **Controle de Concorrência**
  - Teoria da Serialização
  - Protocolos (Baseados em Bloqueios e Não-Bloqueios)
- **Recuperação**
  - Falhas
  - Undo, Redo, Checkpoints

Gerenciamento de Transações



Controle de Concorrência: Teoria da Serialização

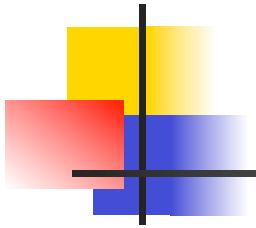
Gerenciamento de Transações



Controle de Concorrência: Teoria da Serialização

- ✓ Serialização no Conflito
- ✓ Serialização na Visão

Gerenciamento de Transações



Controle de Concorrência: Teoria da Serialização

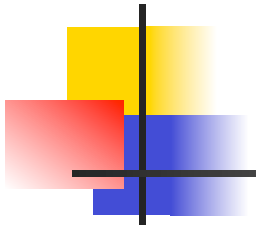
- ✓ Serialização no Conflito

  Teste de Serialização: Grafo de Precedência

- ✓ Serialização na Visão

  Teste de Serialização: Grafo de Precedência Rotulado

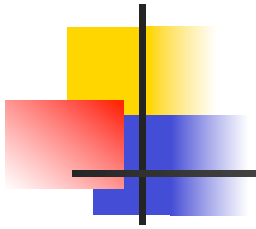
Gerenciamento de Transações



Controle de Concorrência: Teoria da Serialização

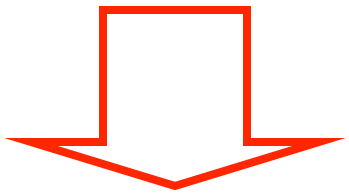
Teste de serialização é inviável na prática!!

Gerenciamento de Transações



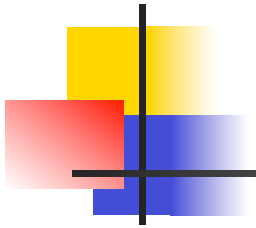
Controle de Concorrência: Teoria da Serialização

Teste de serialização é inviável na prática!!



Protocolos para Controle de Concorrência

Gerenciamento de Transações



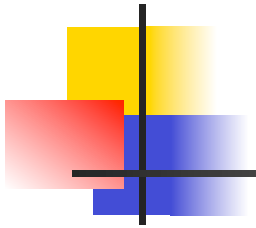
Protocolos para Controle de Concorrência

Garantem que apenas escalas serializáveis

(e, portanto, escalas corretas)

sejam executadas pelo gerente de transações!!

Gerenciamento de Transações

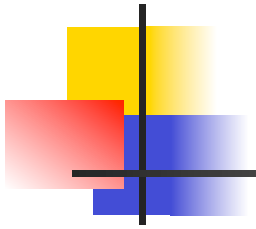


Protocolos para Controle de Concorrência

- Baseados em Bloqueios

- Não Baseados em Bloqueios

Gerenciamento de Transações

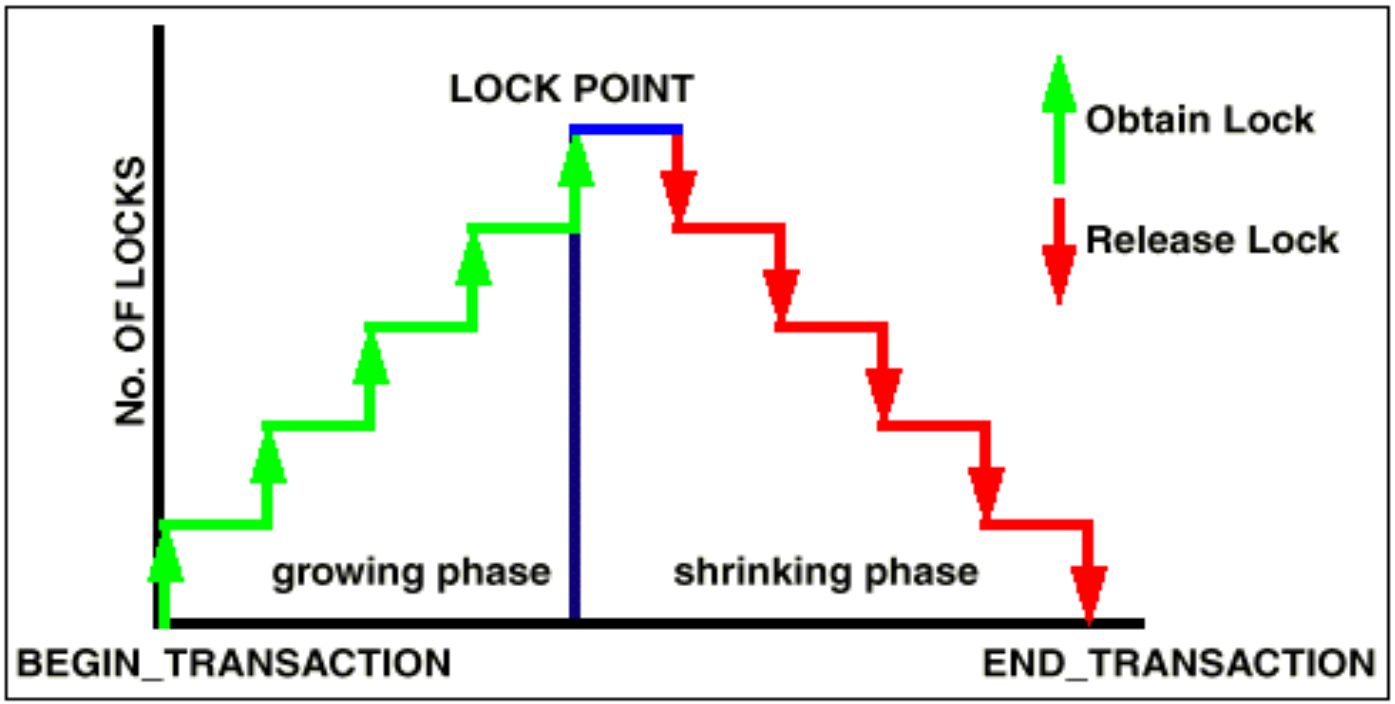


Protocolos para Controle de Concorrência

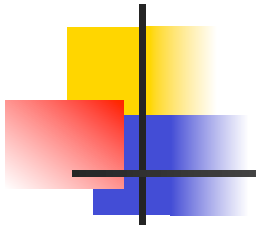
- Baseados em Bloqueios
  - Bloqueio em 2 fases (2PL)
- Não Baseados em Bloqueios
  - Ordenação de TimeStamp (TO)

Gerenciamento de Transações

Protocolo 2PL

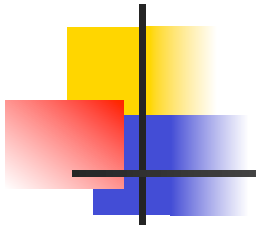


Gerenciamento de Transações



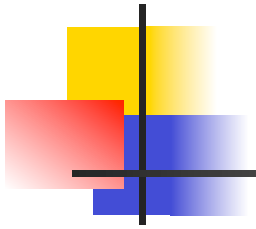
T7	T8
read(B)	read(B)
B = B - 50	read(A)
write(B)	display(A+B)
read(A)	
A = A + 50	
write(A)	

Gerenciamento de Transações



T7	T8
lock-X(B)	lock-S(B)
read(B)	read(B)
B = B - 50	unlock(B)
write(B)	lock-S(A)
unlock(B)	read(A)
lock-X(A)	unlock(A)
read(A)	display(A+B)
A = A + 50	
write(A)	
unlock(A)	

Gerenciamento de Transações

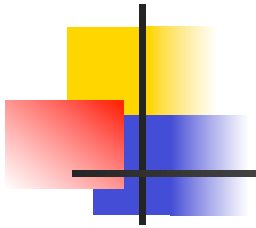


T7	T8
lock-X(B)	lock-S(B)
read(B)	read(B)
B = B - 50	unlock(B)
write(B)	lock-S(A)
unlock(B)	read(A)
lock-X(A)	unlock(A)
read(A)	display(A+B)
A = A + 50	
write(A)	
unlock(A)	

Não 2PL

Não 2PL

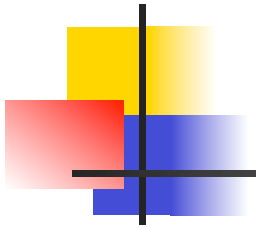
Gerenciamento de Transações



T7	T8
lock-X(B)	
read(B)	
B = B - 50	
write(B)	
unlock(B)	
	lock-S(B)
	read(B)
	unlock(B)
	lock-S(A)
	read(A)
	unlock(A)
	display(A+B)
lock-X(A)	
read(A)	
A = A + 50	
write(A)	
unlock(A)	

Com o uso incorreto de bloqueios e desbloqueios, uma escala não serializável pode ser Executada!!

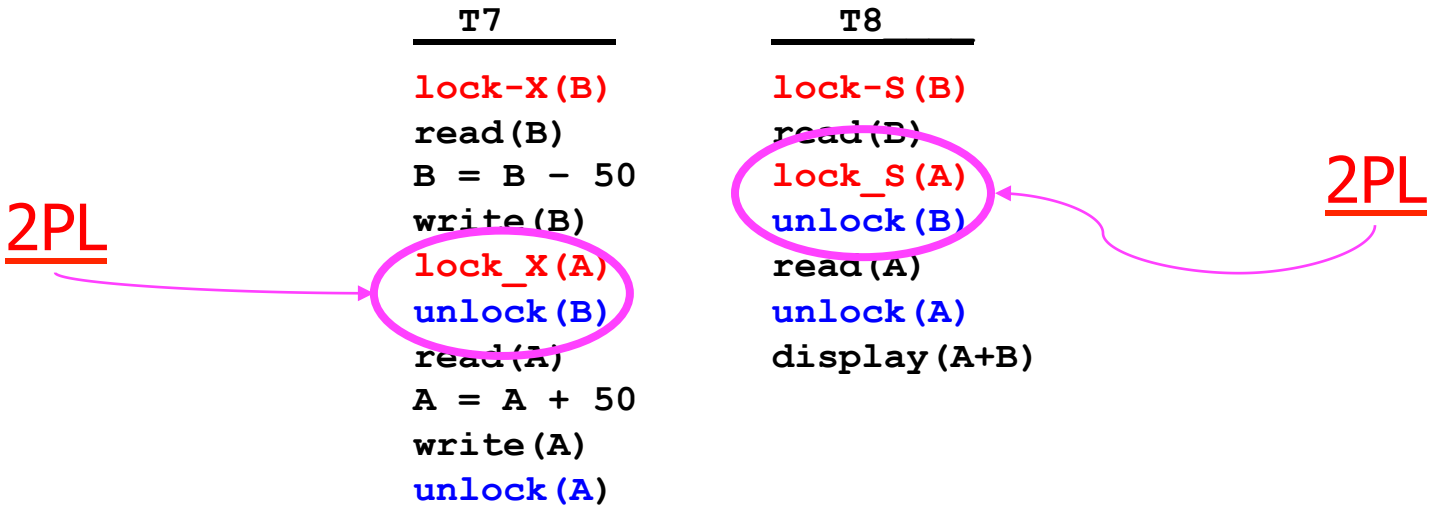
Gerenciamento de Transações



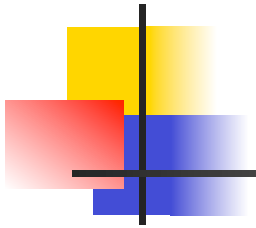
T7	T8
read(B)	
B = B - 50	
write(B)	
	read(B)
	read(A)
	display(A+B)
read(A)	
A = A + 50	
write(A)	

Com o uso incorreto
de bloqueios e desbloqueios,
uma escala não serializável
pode ser
Executada!!

Gerenciamento de Transações



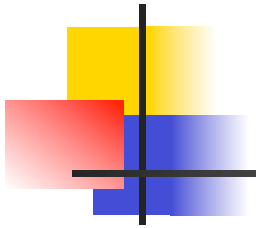
Gerenciamento de Transações



T7	T8
lock-X(B)	
	lock-S(B) ← WAIT
read(B)	
B = B − 50	
write(B)	
lock_X(A)	
unlock(B)	
	read(B)
	lock_S(A) ← WAIT
read(A)	
A = A + 50	
write(A)	
unlock(A)	
	unlock(B)
	read(A)
	unlock(A)
	display(A+B)

2PL gera escalas serializáveis no conflito.

Gerenciamento de Transações



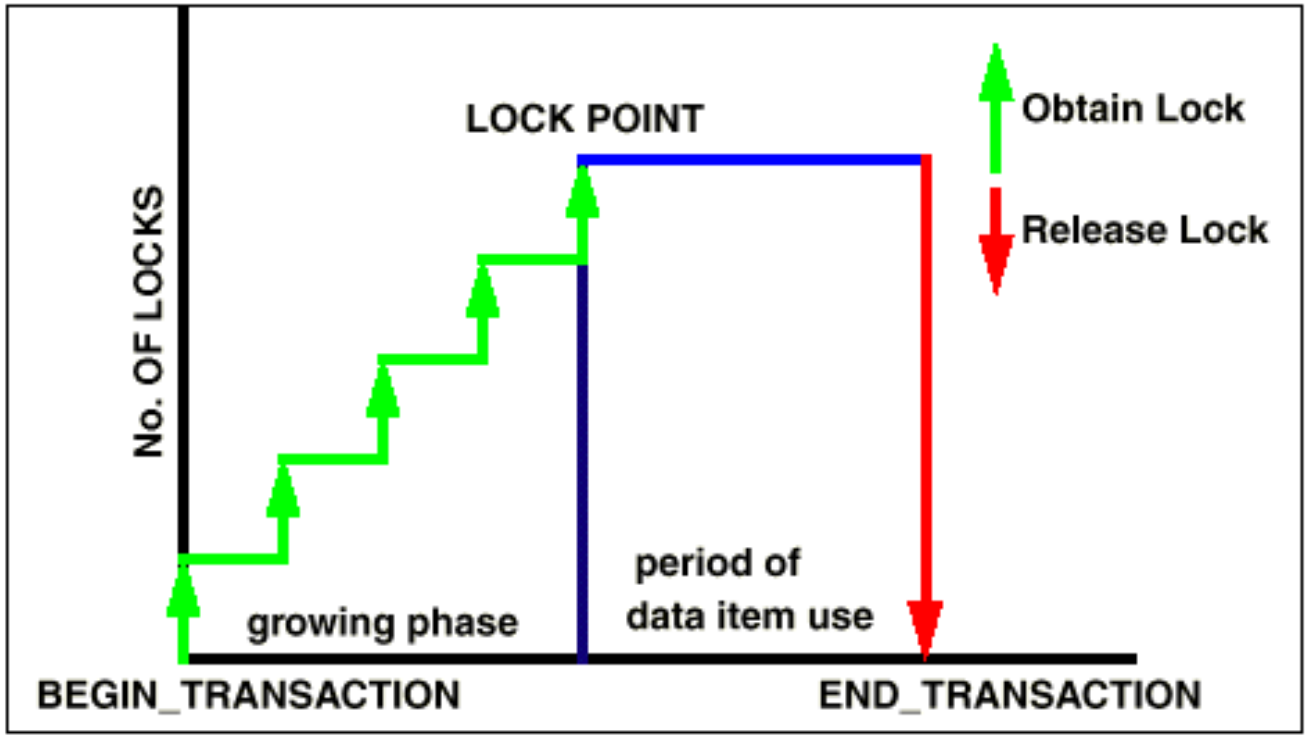
T7	T8
read(B)	
B = B - 50	
write(B)	
	read(B)
read(A)	
A = A + 50	
write(A)	
	read(A)
	display(A+B)

2PL gera escalas serializáveis no conflito.

Gerenciamento de Transações

Protocolo 2PL (Severo ou Rigoroso)

Implementado pela Maioria dos SGBD's comerciais



Gerenciamento de Transações

Protocolo 2PL Severo ou Rigoroso

(Para evitar rollbacks em cascata)

- Severo: bloqueios exclusivos são mantidos até a efetivação da transação
- Rigoroso: todos os bloqueios são mantidos até a efetivação da transação (serialização na ordem das efetivações das transações)

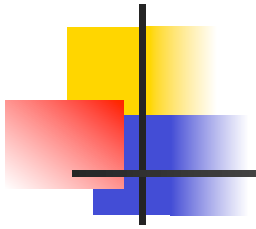
Gerenciamento de Transações

Conversão de Bloqueio

Transaction	System
read(Q)	lock_S(Q)
	read(Q)
write(Q)	if shared lock on Q:
	upgrade(Q)
	write(Q)
	if no shared lock on Q
	lock_X(Q)
	write(Q)

2PL Rigoroso ou Severo com conversão de bloqueio são amplamente utilizados em SGBD's comerciais

Gerenciamento de Transações

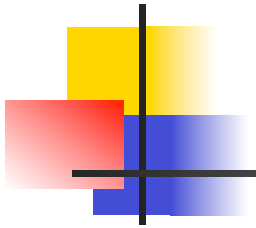


Deadlock

Protocolos baseados em Bloqueios podem causar Deadlocks !

Deadlock ocorre quando uma transação Ti
espera por um bloqueio mantido por uma transação Tj e
vice-versa.

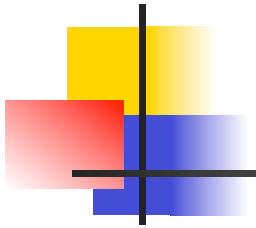
Gerenciamento de Transações



Transações T9 e T10 estão em Deadlock.

T9	T10
lock-X(B)	
	lock-S(A);
read(B)	
	read(A);
B = B – 50	
	lock-S(B); → T10 espera T9
write(B)	
lock-X(A) → T9 espera T10	
unlock(B) read(A) A = A + 50 write(A) unlock(A)	read(B); display(A+B) unlock(A) unlock(B)

Gerenciamento de Transações



Deadlock: Estratégias

- Detecção e Recuperação
- Prevenção

Gerenciamento de Transações

Detecção de Deadlocks

Grafos de Espera

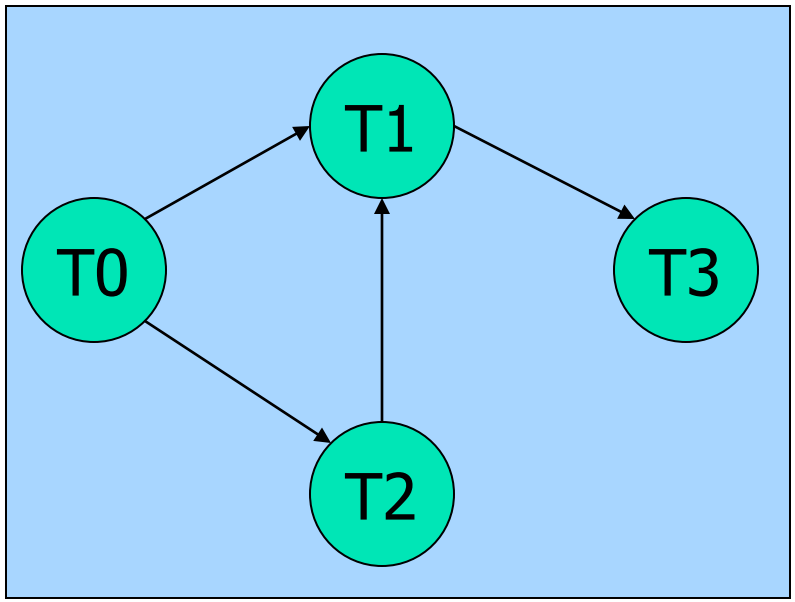
- **Quando uma transação entra no sistema, um vértice é adicionado ao grafo de espera;**
- **Quando uma transação Ti requer um item de dados bloqueado por Tj, então a aresta Ti → tj é adicionada ao grafo de espera.**
- **Quando a transação Tj libera o bloqueio sobre um item de dados que é concedido para uma transação Ti, que estava em estado de espera, então remove-se do grafo de espera a aresta Ti → Tj;**
- **Quando uma transação é efetivada, seu vértice correspondente é eliminado do grafo de espera;**

Se o grafo de espera possuir um ciclo, então existe um deadlock no sistema.

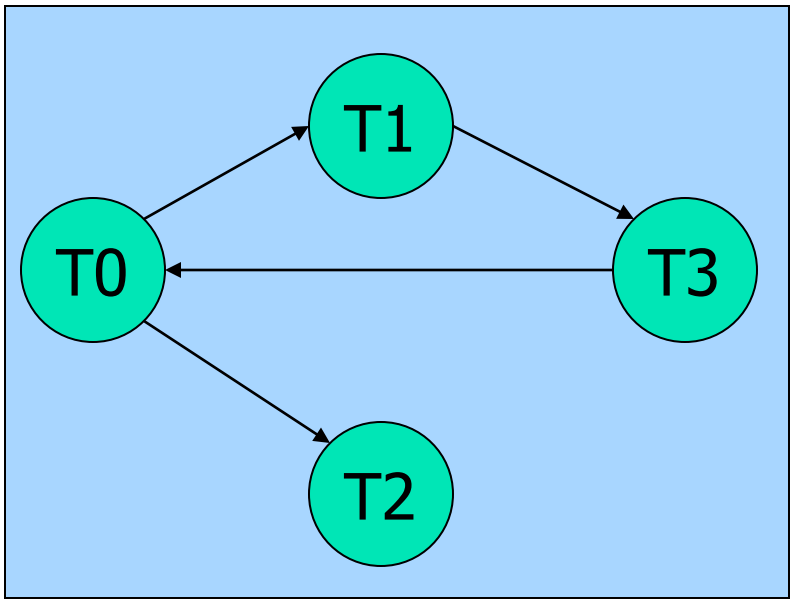
Gerenciamento de Transações

Detecção de Deadlocks

Grafos de Espera



Sem Deadlock



Com Deadlock

Gerenciamento de Transações

Detecção de Deadlocks

Quando o algoritmo de detecção de deadlocks deve ser invocado?

Toda vez que uma transação entra em estado de espera.

Gerenciamento de Transações

Recuperação de Deadlocks

A solução mais comum é escolher uma transação para ser desfeita.

Problemas:

Seleção da vítima: **dever-se-ia escolher a transação mais fácil e de menor custo para ser desfeita. Este custo não é fácil de determinar, ele depende de:**

- **Quão velha é a transação e quanto dela já foi executado;**
- **Quantos itens de dados a transação utilizou;**
- **Quantos itens de dados mais ela necessitará antes de concluir;**
- **Quantas transações serão desfeitas em cascata.**

Gerenciamento de Transações

Recuperação de Deadlocks

Problemas:

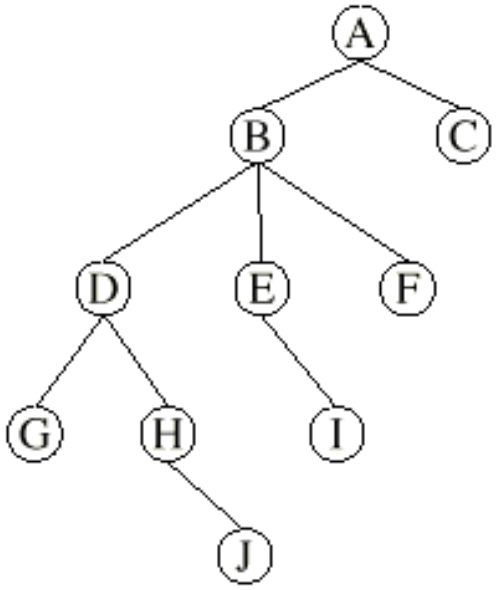
Rollback: Quanto deve ser desfeito da transação? A solução mais fácil é desfazer totalmente a transação, mas se o sistema possuir informações adicionais, poder-se-ia fazer um **roolback parcial.**

Starvation: Se a seleção da vítima é baseada em alguma função custo, a mesma transação poderá ser sempre a escolhida como vítima, o que levaria ao starvation.

Gerenciamento de Transações

Prevenção de Deadlock

Protocolo de Bloqueio Baseado em Grafos (Árvore)



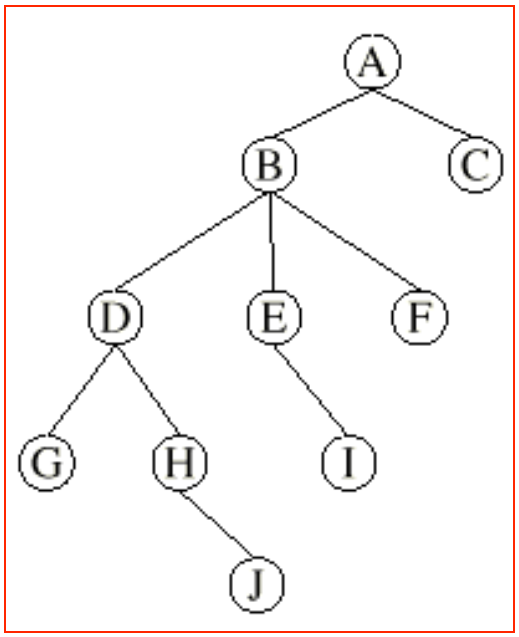
Exige a ordenação parcial do conjunto de itens de dados a serem acessados.

Gerenciamento de Transações

Protocolo de Bloqueio Baseado em Grafos (Árvore)

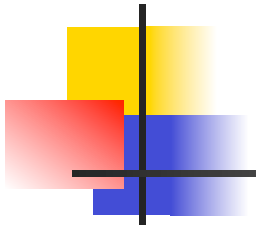
- **o primeiro bloqueio de Ti pode ser feito sobre qualquer item de dados;**
- **subseqüentemente, um item de dados Q só pode ser bloqueado por Ti se o pai de Q estiver bloqueado naquele instante por Ti;**
- **itens de dados podem ser desbloqueados em qualquer instante**
- **um item de dados desbloqueado por Ti não pode ser bloqueado novamente por Ti.**

Gerenciamento de Transações



T13	T14	T15	T16
lock-X(B);			
	lock-X(D);		
	lock-X(H);		
	unlock(D);		
lock-X(E);			
unlock(E);			
lock-X(D);			
unlock(B);			
		lock-X(B);	
		lock-X(E);	
	lock-X(J);		
	unlock(J);		
	unlock(H);		
lock-X(G);			
unlock(D);			
			lock-X(D);
			lock-X(H);
			unlock(D);
			unlock(H);
		unlock(E);	
		unlock(B);	
unlock(G);			

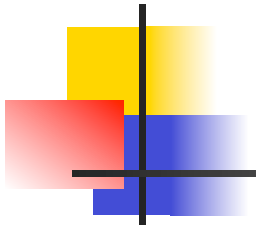
Gerenciamento de Transações



Protocolo de Bloqueio baseado em Grafos (Árvore)

- gera escalas serializáveis no conflito
- gera escalas livres de deadlocks
- podem bloquear dados que não necessitam (desvantagem)
- admite apenas bloqueios exclusivos
- nem sempre é possível ordenar os acessos aos itens de dados

Gerenciamento de Transações

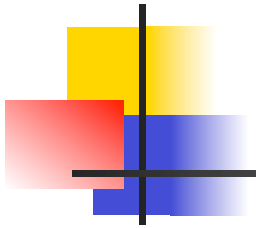


Prevenção de Deadlocks:

Pré-alocação de Recursos: Cada transação aloca (bloqueia) antecipadamente todos os recursos (itens de dados) que precisará.

- **Baixo nível de concorrência**
- **Nem sempre é possível saber com antecedência quais dados serão acessados**

Gerenciamento de Transações



Prevenção de Deadlocks:

Esquema Esperar-Morrer: **quando Ti solicita um bloqueio mantido por Tj, Ti espera se for mais antiga do que Tj, caso contrário Ti morre.**

Esquema Ferir-Esperar: **quando Ti solicita um bloqueio mantido por Tj, Ti espera se for mais nova do que Tj, caso contrário Tj é desfeita (Ti assume o bloqueio requisitado quando for mais velha).**

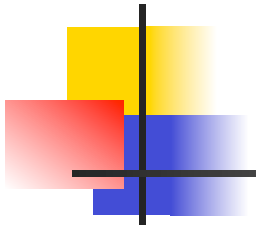
Gerenciamento de Transações

Prevenção de Deadlocks:

Esquema Esperar-Morrer e Esquema Ferir-Esperar

- **As transações desfeitas (rolled back) são re-submetidas com o mesmo timestamp (para evitar starvation);**
- **Em ambos os esquemas podem haver roll-backs desnecessários;**
- **Há menos roll-backs no esquema ferir-esperar.**

Gerenciamento de Transações



Prevenção de Deadlocks:

Esquema baseado em Tempo Esgotado (Timeout): **quando Ti solicita um bloqueio mantido por outra transação, Ti espera pelo desbloqueio durante um certo tempo. Se neste intervalo de tempo, Ti não conseguir o bloqueio, Ti é desfeita (rolled back).**

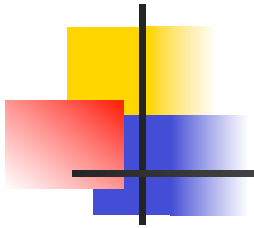
Gerenciamento de Transações

Prevenção de Deadlocks:

Esquema baseado em Tempo Esgotado (Timeout)

- **Esquema de fácil implementação;**
- **Indicado para transações curtas;**
- **É difícil determinar o intervalo de tempo de espera;**
- **Pode ocorrer starvation**

Gerenciamento de Transações



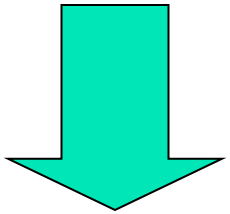
Prevenção de Deadlocks:

Protocolo Não Baseado em Bloqueios

Gerenciamento de Transações

Prevenção de Deadlocks:

Protocolo Não Baseado em Bloqueios



Protocolo de Ordenação de TimeStamp

Gerenciamento de Transações

Protocolo de Ordenação de TimeStamp (TO)

- **cada transação recebe um timestamp;**
- **a escala de execução deve ser equivalente à execução serial determinada pela ordenação dos timestamps;**
- **para cada item de dados é atribuído um timestamp de leitura (R_timestamp) e um timestamp de escrita (W-timestamp), contendo o maior timestamp de transações que leram e escreveram aquele item de dados, respectivamente;**

Gerenciamento de Transações

Protocolo de Ordenação de TimeStamp (TO)

1. **Suponha que a transação T emita uma operação read(Q)**

- **Se TS(T) < W-timestamp(Q), então T deveria ter lido um valor para Q que já foi reescrito. Neste caso, a leitura deve ser rejeitada e T deve ser desfeita.**
- **Se TS(T) >= W-timestamp(Q), então a leitura é executada e R-timestamp(Q) recebe o valor max{ TS(T) , R-timestamp(Q)}.**

Gerenciamento de Transações

Protocolo de Ordenação de TimeStamp (TO)

2. Suponha que T emita uma operação write(Q):

- **Se TS(T) < R-timestamp(Q), então o valor de Q sendo escrito por T foi necessário anteriormente para outra transação. Neste caso a escrita deve ser rejeitada e a transação T deve ser desfeita;**
- **Se TS(T) < W-timestamp(Q), então T está tentanto escrever um valor obsoleto de Q. Neste caso a escrita deve ser rejeitada e a transação T deve ser desfeita;**
- **Caso contrário, a escrita é executada e W-timestamp(Q) recebe o valor TS(T).**

Obs: se uma transação é desfeita, ela recebe um novo timestamp e em seguida é reiniciada.

Gerenciamento de Transações

Protocolo de Ordenação de TimeStamp (TO)

Escala Legal

TS(T1) assigned →

T_1	T_2
	read(bal_B)
read(bal_B)	
bal_B:=bal_B-500	
write(bal_B)	
	read(bal_A)
read(bal_A)	
	display(bal_A+bal_B)
bal_A:=bal_A+500	
write(bal_A)	
display(bal_A+bal_B)	

← TS(T2) assigned

Gerenciamento de Transações

Protocolo de Ordenação de TimeStamp (TO)
Escala Não-Legal

TS(T1) assigned →

T_1	T_2
	read(balA)
read(bal_B)	
bal_B:=bal_B-500	
write(bal_B)	
	read(balB)
read(bal_A)	
	display(bal_A+bal_B)
bal_A:=bal_A+500	
write(bal_A)	
display(bal_A+bal_B)	

← TS(T2) assigned

Gerenciamento de Transações

Protocolo de Ordenação de TimeStamp (TO)

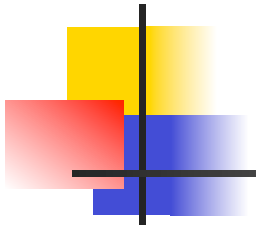
S1

T17	T18
read(B)	
	read(B)
	B = B - 50
	write(B)
read(A)	
	read(A)
display(A+B)	
	A = A + 50
	write(A)
	display(A+B)

S2

T17	T18
read(B)	
	read(B)
	B = B - 50
	write(B)
	read(A)
	A = A + 50
	write(A)
	display(A+B)
read(A)	
display(A+B)	

Gerenciamento de Transações



Protocolo de Ordenação de TimeStamp (TO)

- escalas serializáveis no conflito
- escalas livre de deadlocks
- pode gerar escalas não-recuperáveis e com-cascata
- existem escalas legais sob o protocolo 2PL que não são legais sob o protocolo TO, e vice-versa.

Gerenciamento de Transações

Protocolo de Ordenação de TimeStamp (TO)

Escala Não Legal

T_1	T_2
read(Q)	
	write(Q)
write(Q)	

Gerenciamento de Transações

Protocolo de Ordenação de TimeStamp (TO) com Regra de Escrita de Thomas

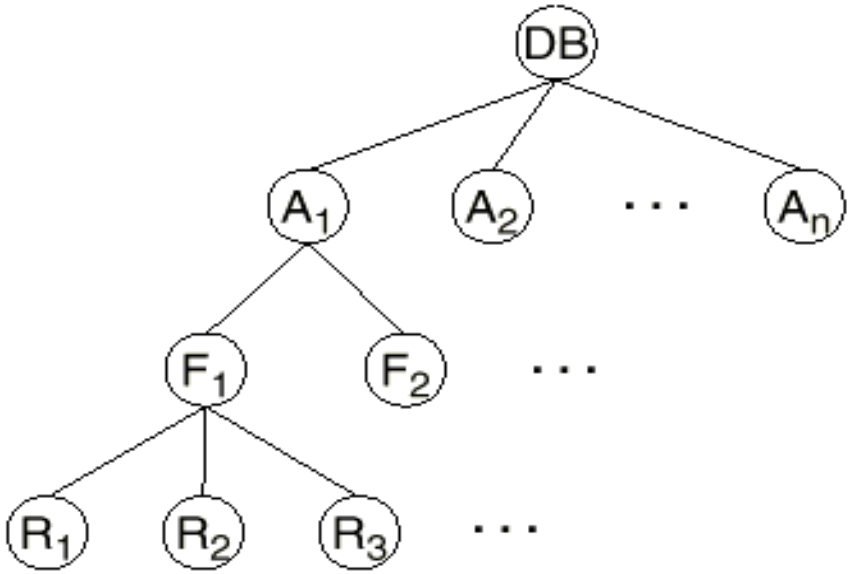
1. **Suponha que T emita uma operação read(Q): (idem TO)**

2. **Suponha que T emita uma operação write(Q):**

- **Se TS(T) < R-timestamp(Q), então a transação T deve ser desfeita;**
- **Se TS(T) < W-timestamp(Q), então ignore esta operação de escrita;**
- **Caso contrário, a escrita é executada e W-timestamp(Q) recebe o valor TS(T).**

Gerenciamento de Transações

Bloqueios: Granularidade Múltipla



Granularidade

DB: banco de dados

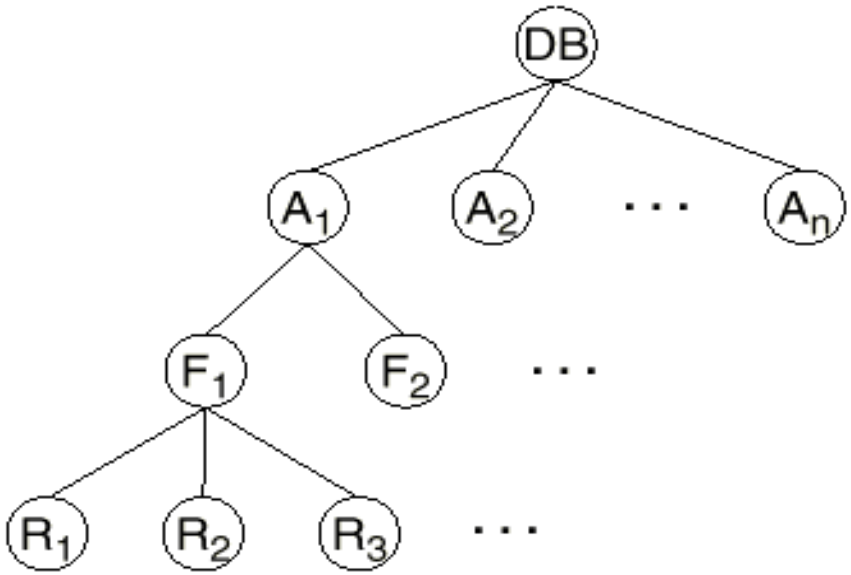
Ai: área i

Fi: arquivo i

Ri: tupla i

Gerenciamento de Transações

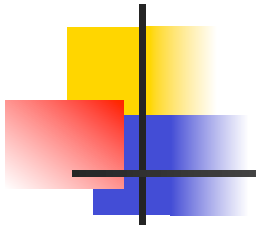
Bloqueios: Granularidade Múltipla



Como o sistema saberá quais nós foram bloqueados?

Percorre-se a árvore da raiz até o nó a ser bloqueado. Se algum nó neste caminho estiver bloqueado, então estão bloqueados todos os nós abaixo dele.

Gerenciamento de Transações



Bloqueios: Granularidade Múltipla

Bloqueios pode ser: exclusivos, compartilhados ou intencionais.

Bloqueios intencionais implícitos são feitos em todos os nós antecessores de um nó antes que ele seja bloqueado explicitamente.

Gerenciamento de Transações

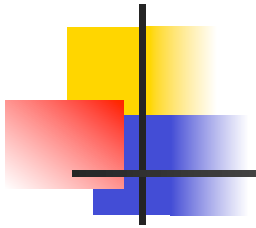
Bloqueios: Granularidade Múltipla

Matriz de Compatibilidade

	IS	IX	S	SIX	X
IS	yes	yes	yes	yes	no
IX	yes	yes	no	no	no
S	yes	no	yes	no	no
SIX	yes	no	no	no	no
X	no	no	no	no	no

IS - intention-shared
IX - intention exclusive
SIX - shared and intention-exclusive

Gerenciamento de Transações



Protocolo de Bloqueio de Granularidade Múltipla

Uma transação Ti pode bloquear um nó Q, obedecendo as seguintes regras:

- **a matriz de compatibilidade deve ser observada**
- **a raiz da árvore precisa ser bloqueada primeiro e em qualquer modo**
- **um nó Q pode ser bloqueado por Ti no modo S ou IS somente se o pai de Q for bloqueado por Ti no modo IX ou IS**
- **um nó Q pode ser bloqueado por Ti no modo X, SIX ou IX somente se o pai de Q for bloqueado por Ti no modo IX ou SIX**

Gerenciamento de Transações

Protocolo de Bloqueio de Granularidade Múltipla

Uma transação Ti pode bloquear um nó Q, obedecendo as seguintes regras:

- **Ti pode bloquear um nó somente se ele não desbloqueou outro nó anteriormente (ou seja, Ti satisfaz 2PL)**
- **Ti pode desbloquear um nó Q somente se nenhum dos seus filhos estiver bloqueado por Ti.**

Gerenciamento de Transações

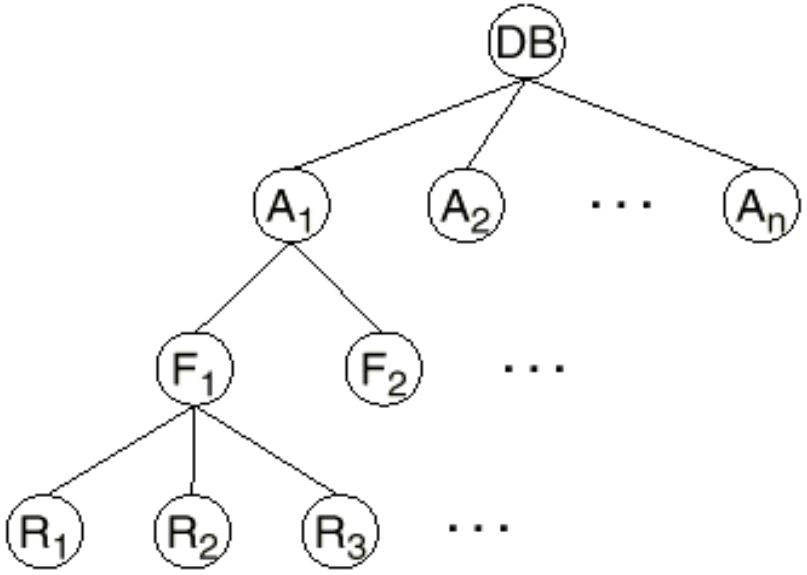
Protocolo de Bloqueio de Granularidade Múltipla

Bloqueios → Top – down

Desbloqueios → Bottom - up

Gerenciamento de Transações

Protocolo de Bloqueio de Granularidade Múltipla

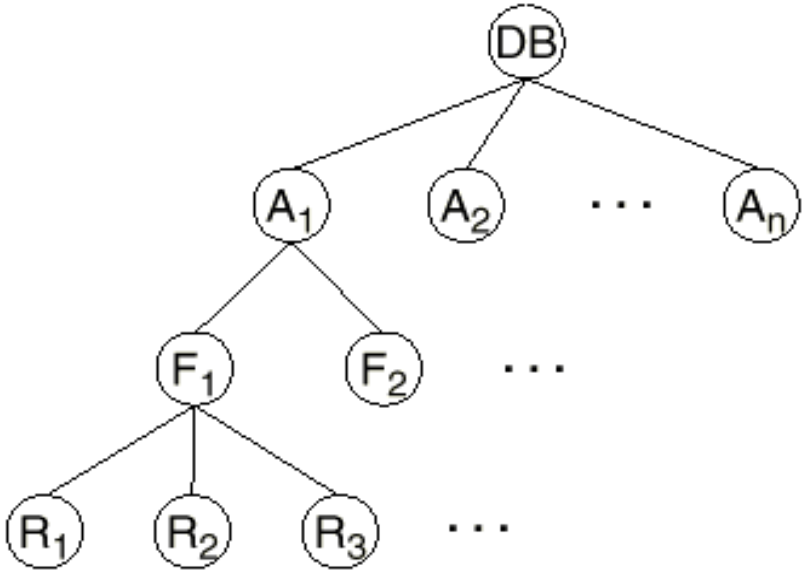


T18 lê o registro R2 do arquivo F1.

Então, T18 precisa bloquear o BD, a área A1 e o arquivo F1, no modo IS (nessa ordem) e, finalmente, bloquear R2 de F1 no modo S.

Gerenciamento de Transações

Protocolo de Bloqueio de Granularidade Múltipla

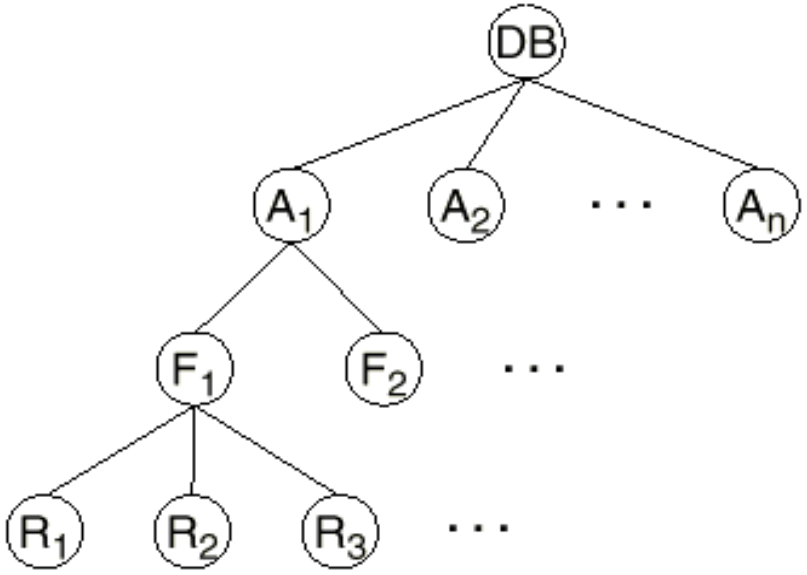


T19 altera o registro R9 do arquivo F2.

Então, T19 precisa bloquear o BD, a área A1 e o arquivo F2, no modo IX (nessa ordem) e, finalmente, bloquear R9 de F2 no modo X.

Gerenciamento de Transações

Protocolo de Bloqueio de Granularidade Múltipla

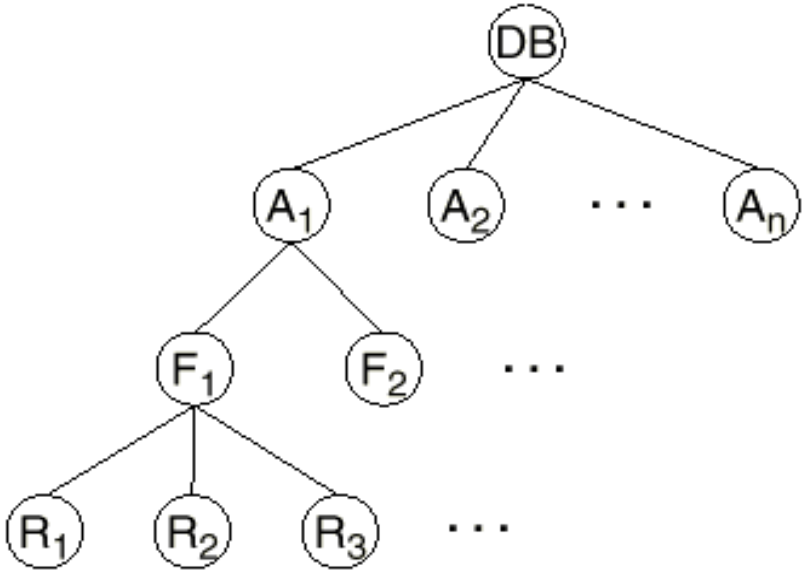


T20 lê todos os registros do arquivo F2.

Então, T20 precisa bloquear o BD e a área A1 no modo IS (nessa ordem) e, finalmente, bloquear F2 no modo S.

Gerenciamento de Transações

Protocolo de Bloqueio de Granularidade Múltipla



T21 lê todos os registros do Banco de Dados.

Então T21 precisa bloquear o BD no modo S.

Gerenciamento de Transações

Protocolo de Bloqueio de Granularidade Múltipla

- **diminui o overhead por bloqueio**
- **útil em aplicações que misturam transações curtas que mantém acessos a poucos itens de dados e transações longas que produzem relatórios a partir de um arquivo ou de um conjunto de arquivos**

Gerenciamento de Transações

Protocolo TO Multiversão

- **associe um timestamp TS(Ti) com cada transação**
- **cada versão de um item de dados Q contém:**
  - **conteúdo (Q_k)**
  - **write_TS(Q_k) – timestamp da transação que criou Q_k**
  - **read_TS (Q_k) – maior timestamp das transações que leram Q_k com êxito**

Gerenciamento de Transações

Protocolo TO Multiversão

Seja Q_k a versão de Q cujo timestamp tem o maior write_TS menor ou igual a TS(Ti).

Para read(Q) $\in$ Ti:

- **retorne Q_k**

Gerenciamento de Transações

Protocolo TO Multiversão

Seja Q_k a versão de Q cujo timestamp tem o maior write_TS menor ou igual a TS(Ti).

Para write(Q) ∈ Ti:

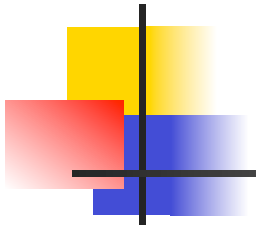
- **se TS(Ti) < read_TS(Q_k) → Ti é desfeita**
- **senão, se TS(Ti) = write_TS(Q_k) → write(Q_k)**
- **senão → cria uma nova versão de Q**

Gerenciamento de Transações

Protocolo 2PL Multiversão

- **cada item de dado tem um único timestamp, TS_counter, que é incrementado durante o processo de efetivação das transações.**

Gerenciamento de Transações



Protocolo 2PL Multiversão

Para transações read-only:

- **utilize o protocolo TO Multiversão**
  - **usa o valor de TS_Counter como sendo o timestamp das transações**
  - **lê a versão do valor com maior timestamp, menor ou igual ao timestamp da transação**

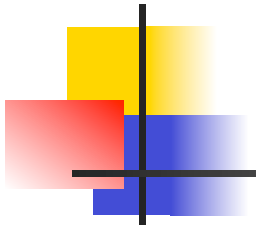
Gerenciamento de Transações

Protocolo 2PL Multiversão

Para transações de atualização:

- **utilize o protocolo 2PL rigoroso**
- **crie novas versões dos itens de dados com TS_Counter = ∞**
- **durante a efetivação, os timestamps das versões criadas são alterados para TS_Counter + 1 e TS_Counter é incrementado de 1;**

Gerenciamento de Transações

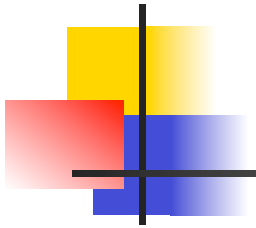


Protocolo 2PL Multiversão

- **transações read-only que iniciam depois do incremento de TS_Counter lêem as novas versões, as demais lêem as versões antigas**

- **nenhuma transação precisa esperar**

Gerenciamento de Transações

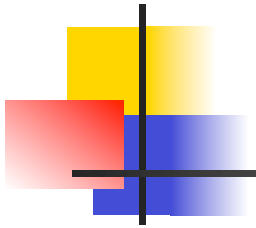


Novas Operações

Remoção → Delete(Q)

Inserção → Insert(Q)

Gerenciamento de Transações



Casos a Serem Considerados

Read(Q) depois de Delete(Q)

Read(Q) antes de Insert(Q)

Delete(Q) antes de Insert(Q)

Gerenciamento de Transações

Operação de Remoção

- **Se for utilizado o 2PL, então é necessário um bloqueio exclusivo sobre o item de dados Q a ser removido - delete(Q);**

- **Se for utilizado o TO, então deve-se efetuar para a operação delete(Q) o mesmo teste efetuado para a operação write(Q).**

Gerenciamento de Transações

Operação de Inserção

- **Se for utilizado o 2PL, então é necessário um bloqueio exclusivo sobre o item de dados Q a ser inserido - insert(Q);**

- **Se for utilizado o TO e T efetuar uma operação insert(Q), então os valores de W_timestamp(Q) e R_timestamp(Q) receberão o valor TS(T).**