CONTROLE DE CONCORRÊNCIA - DEADLOCK

Sérgio Mergen

Versão modificada de Database System Concepts, 5th Ed.

Deadlocks

- O sistema está em deadlock se
  - Existe um conjunto de transações de modo que toda transação do conjunto está esperando por outra transação do conjunto.

T1	T2
lock-X*(A)*;	
write (A)	
	lock-X*(B)*;
	write (B)
lock-X*(B)*; (travou)	
	lock-X*(A)*; (travou)

Tratamento de Deadlock

- Basicamente, deadlocks podem ser tratados de duas formas:
  - Prevenção de deadlocks
  - Recuperação de deadlocks

PREVENÇÃO DE DEADLOCK

Prevenção de Deadlocks

- Protocolos de ***Prevenção de Deadlock*** garantem que o sistema nunca entre em um estado de deadlock.

- Algumas estratégias de prevenção:
  - Pre-declaração
  - Ordenação de locks
  - Uso de timestamps

Prevenção de Deadlocks

- Pre-declaração
  - Exigir que todas transações façam lock em todos itens de dados que precisarem antes do começo da execução.
  - Normalmente inviável. Porquê?

- Ordenação de locks
  - Impor uma ordem parcial nos itens de dados
  - Exigir que uma transação obtenha lock somente na ordem especificada pela ordenação parcial
    - Ex. protocolo baseado em árvore.

Prevenção de Deadlocks

- Uso de timestamps
  - Carimbar a transação com a data e hora em que ela iniciou
  - Usar essa informação para controlar o acesso aos itens de dados

- Os esquemas abaixo usam timestamps para a prevenção.
  - **Wait-die**
  - **Wound-wait**
  - **Baseado em time-out**

Prevenção de Deadlocks com Timestamps

- **wait-die** — não preemptivo
  - Transações mais velhas esperam que as mais novas liberem um item de dados.
  - Transações mais novas nunca esperam pelas mais velhas
    - Em vez disso, elas são revertidas.
  - Característica
    - Uma transação pode morrer diversas vezes antes de obter acesso a algum item de dados

Prevenção de Deadlocks com Timestamps

- **wound-wait** — preemptivo
  - Transações mais velhas ferem (forçam rollback) das transações mais novas, em vez de esperar por elas.
  - Transações mais novas esperam pelas mais velhas.
  - Característica
    - Pode ter menos rollbacks do que o esquema *wait-die*.

Prevenção de Deadlocks com Timestamps

- Esquemas *wait-die* e *wound-wait*
  - Uma transação que foi revertida é reiniciada com seu timestamp original.
  - Assim, transações mais velhas tem precedência sobre as mais novas
    - e o starvation é evitado.

Prevenção de Deadlocks com Timestamps

- Esquemas baseados em timeout
  - Uma transação espera por um lock apenas por um período específico de tempo.
  - Depois disso, a espera é finalizada e a transação é revertida.
  - Assim, deadlocks são impossíveis
  - Simples para implementar
  - Não evita starvation.
  - Difícil determinar um valor bom para o período de tempo.

RECUPERAÇÃO DE DEADLOCK

Recuperação de Deadlock

- Quando for detectado um deadlock
  - Alguma transação precisa sofrer rollback (ser a vítima) para quebrar o deadlock
    - Selecionar como vítima a transação que incorrer no menor custo
  - Starvation acontece se a mesma transação sempre for escolhida como vítima.
  - Para evitar isso
    - Pode-se incluir no fator de custo o número de vezes que uma transação foi revertida

Recuperação de Deadlock

- Para usar alguma estratégia de deadlock é necessário antes detectar que ocorreu um deadlock

- Forma de detecção?
  - Grafo de espera (grafo *wait-for*)

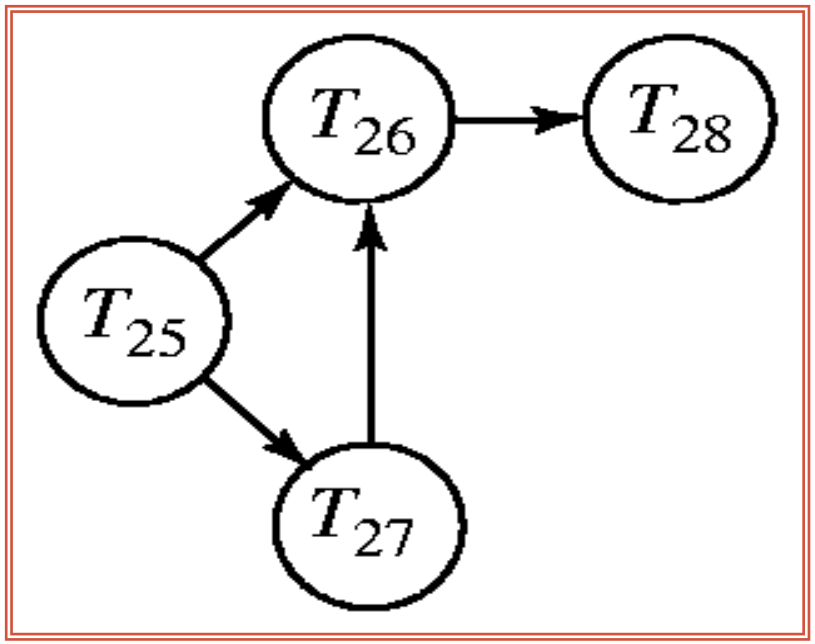
Detecção de Deadlock

- Um grafo de espera consiste de pares $G = (V,E)$,
  - V é um conjunto de vértices (todas transações)
  - E é um conjunto de arcos dirigidos $T_x \rightarrow T_y$.
- O arco $T_x \rightarrow T_y$ significa que
  - T_x está esperando que T_y libere um item.

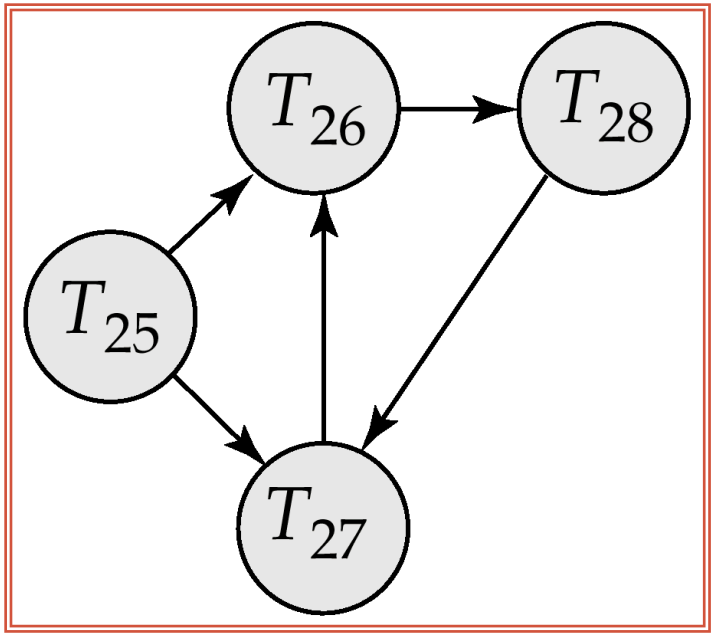
- Quando T_x pede um item que está mantido por T_y, o arco $T_x \rightarrow T_y$ é inserido no grafo.
  - Esse arco é removido somente quando T_y liberar o item de dados solicitado por T_x.

- Existe um deadlock se o grafo possuir um ciclo.
  - Deve-se invocar um algoritmo de detecção de ciclos para identificar a ocorrência de um deadlock.

Detecção de Deadlock



Grafo de espera sem ciclo



Grafo de espera com ciclo