

# Penguncian pada Concurrency Control

Teknik Informatika  
Universitas Kristen Duta Wacana  
Yogyakarta

# Tujuan

- Memahami tentang konsep penguncian pada concurrency control terhadap transaksi database.

# Pendahuluan

- Ketika beberapa transaksi berjalan secara bersamaan dalam database, properti isolasi tidak dapat dipertahankan.
- Untuk itu, sistem perlu menerapkan skema concurrency-control.
  - Untuk menjamin interaksi antar transaksi concurrent
  - Didasarkan pada properti serializability.

# Protokol Berbasis Kunci

- Untuk memastikan serializability, salah satunya dengan cara *mutually exclusive*:
  - Ketika satu transaksi mengakses item data, tidak ada transaksi lain yang dapat memodifikasi item data tersebut.
  - Dengan cara mengunci item data.

# Penguncian

- Shared
  - Jika  $T_i$  mendapat shared-mode lock (S) pada item Q, maka  $T_i$  dapat membaca, tapi tidak dapat menulis item Q.
- Exclusive
  - Jika  $T_i$  mendapat exclusive-mode lock (X) pada item Q, maka  $T_i$  tidak dapat membaca atau menulis item Q.

# Compatibility Function

- Setiap transaksi meminta kunci ke *concurrency-control manager*.
  - Transaksi dapat melanjutkan operasinya hanya setelah mendapat *grant* dari manager.
- Dari sekumpulan mode penguncian, kita dapat mendefinisikan *compatibility function*.

	S	X
S	true	false
X	false	false

# Locking Protocol

- Jika kita tidak menggunakan penguncian, atau jika kita melepas kunci segera setelah baca atau tulis item data, kita mungkin akan mendapat kondisi tidak konsisten.
- Jika kita tidak melepas kunci item data sebelum meminta untuk mengunci item data lain, deadlock mungkin terjadi.
- Harus mengikuti aturan: **locking protocol**
  - Menentukan kapan suatu transaksi mengunci atau melepas kunci untuk setiap item data.

# Locking Protocol

- $\{T_0, T_1, \dots, T_n\}$  adalah himpunan transaksi dalam schedule  $S$ .
- $T_i$  dan  $T_j$  dalam  $S$  jika
  - Terdapat item data  $Q$ , dan
  - $T_i$  mengunci  $Q$  mode  $A$ , dan  $T_j$  mengunci  $Q$  mode  $B$  kemudian, dan
  - $\text{comp}(A, B) = \text{false}$

# Locking Protocol

- Jadwal  $S$  dikatakan **legal** di bawah locking protocol, jika  $S$  adalah jadwal yang mengikuti aturan locking protocol.
- Locking protocol memastikan conflict serializability.

# Granting Lock

- Ketika Ti meminta kunci pada item data Q dalam mode M, concurrency-control manager memberikan kunci :
  - Tidak ada transaksi lain menyimpan kunci untuk Q dalam mode yang konflik dengan M;
  - Tidak ada transaksi lain yang sedang menunggu untuk mengunci Q sebelum permintaan Ti.

# Contoh

- Diberikan T1 dan T2 berikut:

$T_1$ : lock-X(B);  
read(B);  
 $B := B - 50$ ;  
write(B);  
unlock(B);  
lock-X(A);  
read(A);  
 $A := A + 50$ ;  
write(A);  
unlock(A).

$T_2$ : lock-S(A);  
read(A);  
unlock(A);  
lock-S(B);  
read(B);  
unlock(B);  
display(A + B).

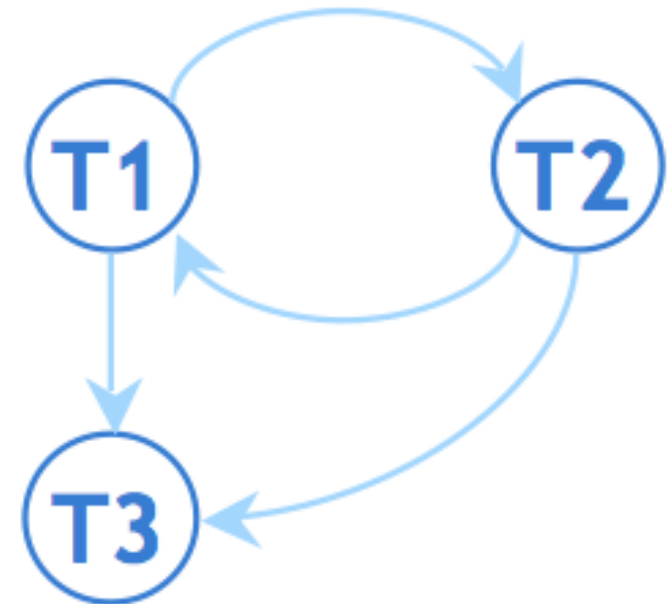
- Jika  $A = 100$ , dan  $B=200$ , berapa nilai akhir jika dijalankan secara serial?

# Jika secara concurrent?

$T_1$	$T_2$	concurrency-control manager
lock-X(B)		grant-X(B, $T_1$ )
read(B)		
$B := B - 50$		
write(B)		
unlock(B)		
	lock-S(A)	grant-S(A, $T_2$ )
	read(A)	
	unlock(A)	
	lock-S(B)	grant-S(B, $T_2$ )
	read(B)	
	unlock(B)	
	display(A + B)	
lock-X(A)		grant-X(A, $T_2$ )
read(A)		
$A := A + 50$		
write(A)		
unlock(A)		

# Presecende Graph

T1	T2	T3
R(A)		
	W(A)	
	Commit	
W(A)		
Commit		
		W(A)
		Commit



# Kegagalan Locking Protocol

- Masih mungkin terjadi deadlock
- Starvation
  - Sebuah transaksi menunggu untuk X-lock untuk item data, sedangkan sederetan transaksi lain meminta dan diberi suatu S-lock untuk item data yang sama.
  - Transaksi yang sama di rollback secara berulang karena deadlock.
  - Concurrency control manager dirancang untuk mencegah starvation.

# deadlock

$T_3$ : lock-X( $B$ );  
 read( $B$ );  
 $B := B - 50$ ;  
 write( $B$ );  
 lock-X( $A$ );  
 read( $A$ );  
 $A := A + 50$ ;  
 write( $A$ );  
 unlock( $B$ );  
 unlock( $A$ ).

$T_4$ : lock-S( $A$ );  
 read( $A$ );  
 lock-S( $B$ );  
 read( $B$ );  
 display( $A + B$ );  
 unlock( $A$ );  
 unlock( $B$ ).

$T_3$	$T_4$
lock-X( $B$ )	
read( $B$ )	
$B := B - 50$	
write( $B$ )	
	lock-S( $A$ )
	read( $A$ )
	lock-S( $B$ )
lock-X( $A$ )	

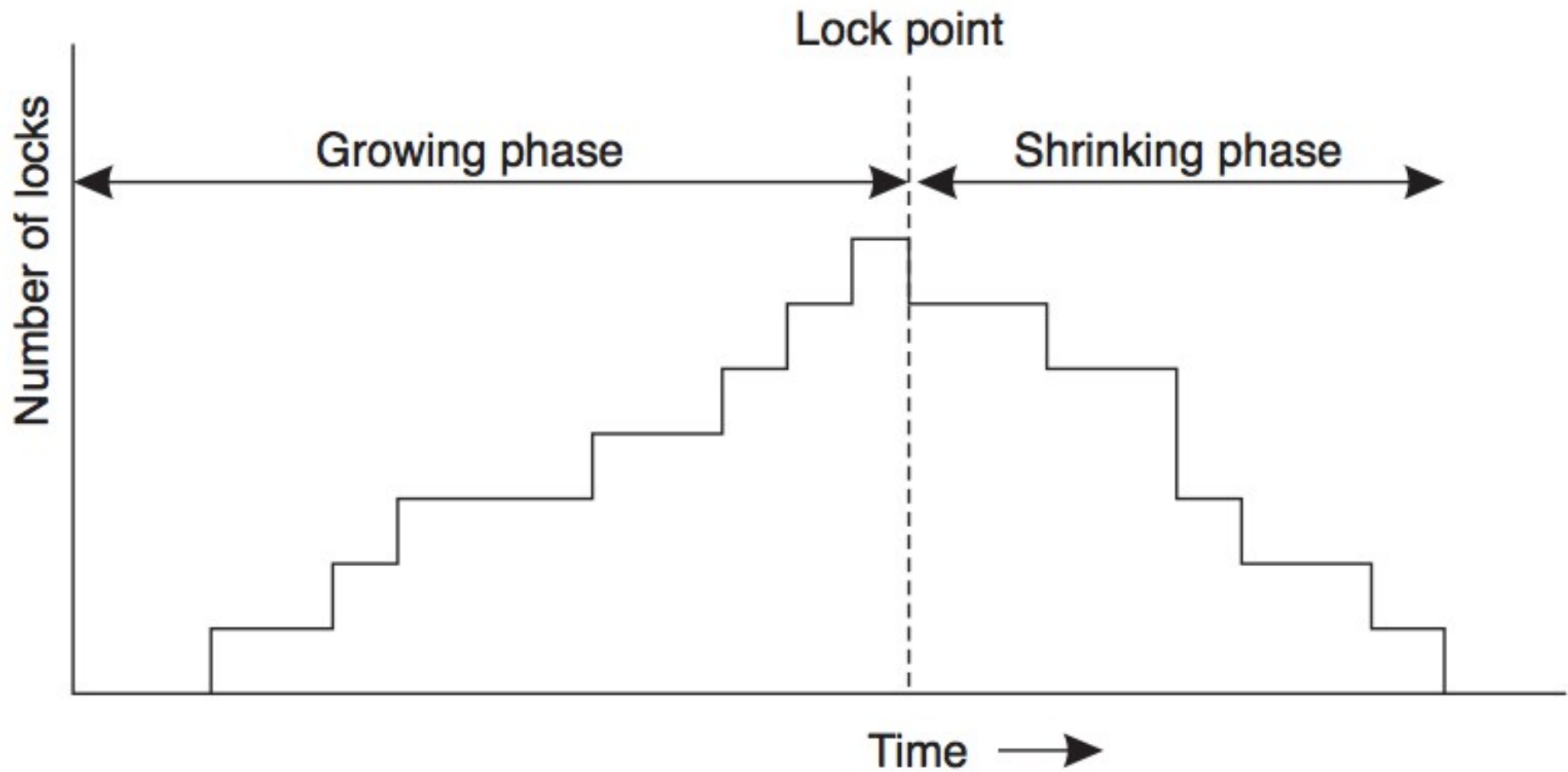
# Contoh lain

<i>T1: Transfer \$100 from A to B</i>	<i>T2: Add 10% interest to A &amp; B</i>
begin	
	begin
R(A); A -= 100	
W(A)	
	R(B); B *= 1.1
	W(B)
R(B); B += 100	
W(B)	
commit	
	R(A); A *= 1.1
	W(A)
	commit

# Two-Phase Locking Protocol

- Protokol ini membutuhkan setiap transaksi yang ingin kunci atau melepas kunci, memintanya dalam 2 fase:
  - Growing phase:
    - Sebuah transaksi dapat memperoleh kunci, tetapi tidak dapat melepaskan kunci apapun.
  - Shrinking phase
    - Transaksi mungkin melepas kunci, namun tidak mendapatkan kunci baru.

# Two-phase Locking



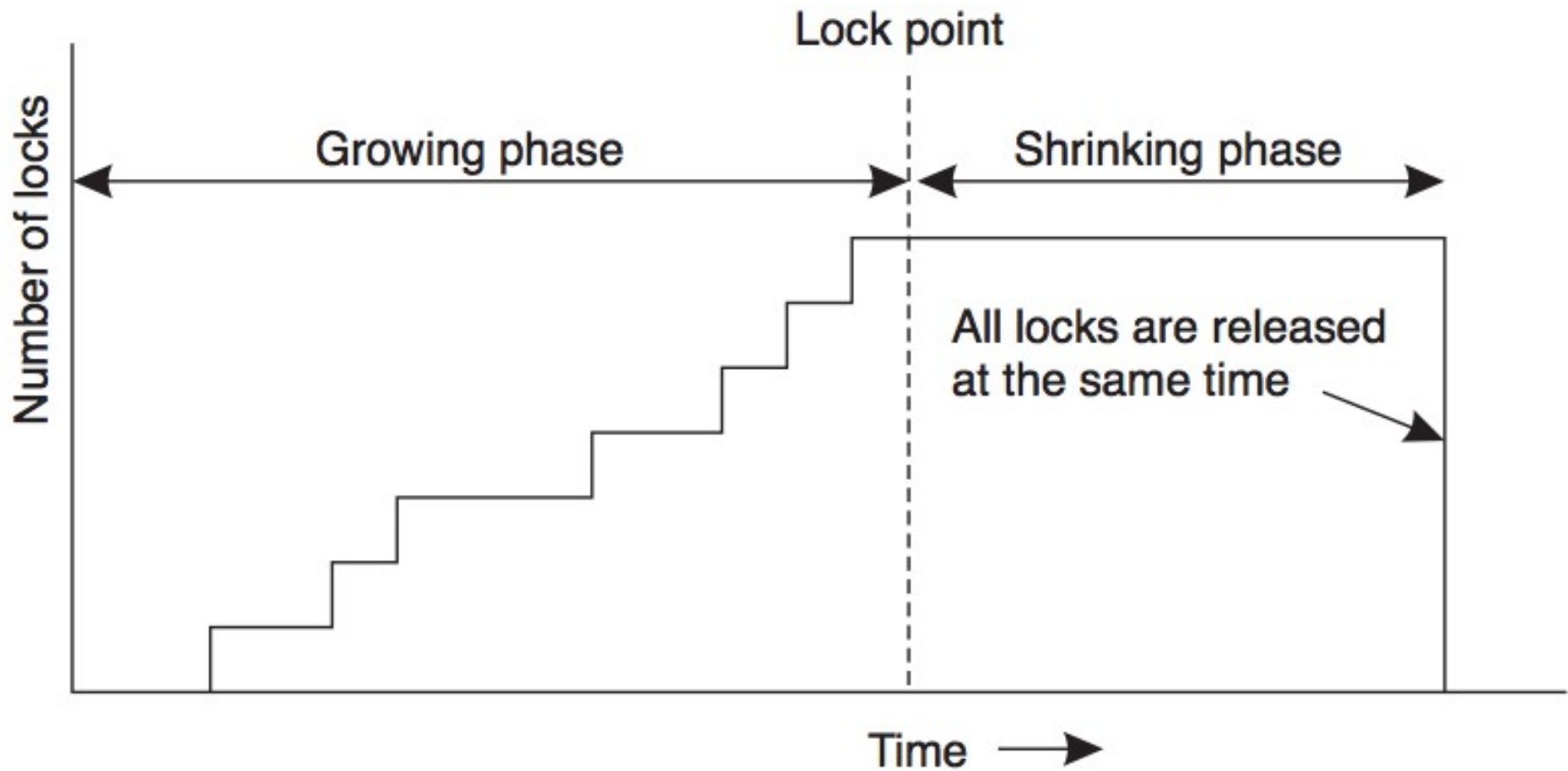
# Two-Phase Locking Protocol

- Pada awalnya, sebuah transaksi dalam growing phase.
- Sekali melepas sebuah kunci, masuk ke shrinking phase, dan tidak dapat meminta kunci lagi.
- Suatu titik pada schedule dimana transaksi mencapai kunci akhir (akhir growing phase) disebut **lock point** dari transaksi.

# Two-Phase Locking Protocol

- Protokol Two-phase tetap memungkinkan deadlock.
- Modifikasinya:
  - Strict two-phase locking protocol
    - Sembarang data yang ditulis oleh uncommit transaction dikunci mode exclusive sampai commit, untuk mencegah transaksi lain membaca data.
  - Rigorous two-phase locking protocol
    - Semua kunci disimpan, sampai commit atau rollback.

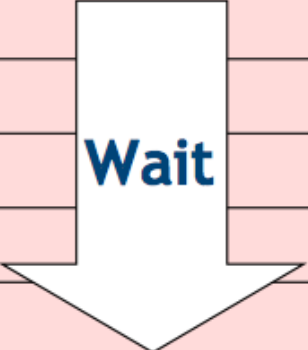
# Strict two-phase locking protocol



# Konversi Kunci

- Two-phase locking dengan konversi kunci:
  - First Phase:
    - Dapat memperoleh lock-S untuk item
    - Dapat memperoleh lock-X untuk item
    - Dapat mengubah lock-S ke lock-X (**upgrade**)
  - Second Phase:
    - Dapat melepas lock-S
    - Dapat melepas lock-X
    - Dapat mengubah lock-X ke lock-S (**downgrade**)
- Protokol ini memastikan serializability. Namun tetap mengandalkan programmer untuk menyisipkan perintah penguncian.

# Contoh

<i>T1: Transfer \$100 from A to B</i>	<i>T2: Add 10% interest to A &amp; B</i>
begin	
	begin
R(A); A -= 100	 Wait
W(A)	
R(B); B += 100	
W(B)	
commit	
	R(A); A *= 1.1
	W(A)
	R(B); B *= 1.1
	W(B)
	commit

# Graph Based Protocol

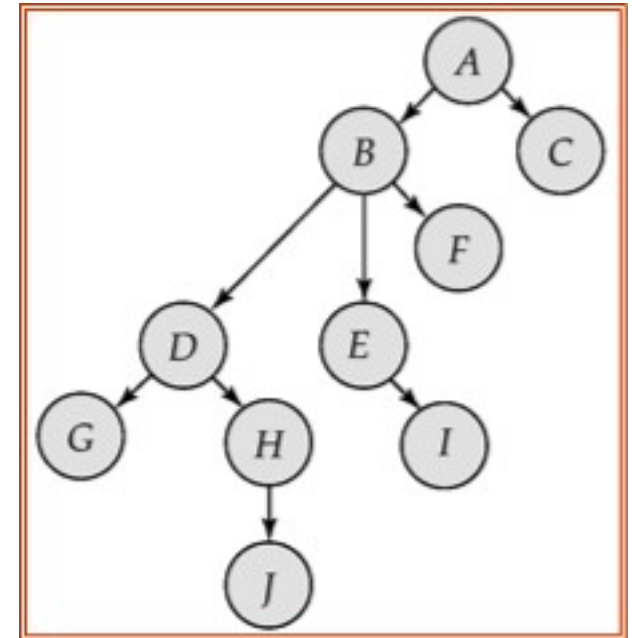
- Protokol alternatif dari two-phase
- Memaksa partial ordering  $\rightarrow$  pada himpunan  $D = \{d_1, d_2, \dots, d_n\}$  dari semua item data.
  - Jika  $d_i \rightarrow d_j$ , maka sembarang transaksi mengakses  $d_i$  dan  $d_j$  harus mengakses  $d_i$  dahulu baru  $d_j$ .
- Menggunakan tree protocol

# Tree Protocol

- Hanya pengucian eksklusif.
- Kunci pertama oleh  $T_i$  dapat dilakukan untuk sembarang item data. Selanjutnya, data  $Q$  dapat dikunci oleh  $T_i$ , hanya jika induk  $Q$  dikunci oleh  $T_i$ .
- Item data dapat di unlocked setiap saat.
- Item data yang di kunci dan di unlock oleh  $T_i$  berikutnya tidak dapat dikunci ulang oleh  $T_i$ .

# Tree Protocol

$T_{10}$	$T_{11}$	$T_{12}$	$T_{13}$
lock-X(B)	lock-X(D) lock-X(H) unlock(D)		
lock-X(E) lock-X(D) unlock(B) unlock(E)		lock-X(B) lock-X(E)	
lock-X(G) unlock(D)	unlock(H)		
		unlock(E) unlock(B)	
unlock (G)			lock-X(D) lock-X(H) unlock(D) unlock(H)



# Akuisisi Kunci otomatis

- Sebuah transaksi  $T_i$  melakukan perintah read/write standar, tanpa melakukan pemanggilan kunci secara eksplisit.
- The operation  $\text{read}(D)$  is processed as:

```
if  $T_i$  has a lock on D
  then
    read(D)
  else begin
    if necessary wait until no other
      transaction has a lock-X on D
    grant  $T_i$  a lock-S on D;
    read(D)
  end
```

# Akuisisi Kunci otomatis

- write(D) diproses sebagai :  
if Ti has a lock-X on D  
  then  
    write(D)  
  else begin  
    if necessary wait until no other trans. has any lock on D,  
    if Ti has a lock-S on D  
      then  
        upgrade lock on D to lock-X  
      else  
        grant Ti a lock-X on D  
    write(D)  
  end;
- Semua kunci dilepas setelah commit atau abort

# Penanganan Deadlock

- Terdapat 2 transaksi:

T1:    write (X)  
        write(Y)

T2:    write(Y)  
        write(X)

- Schedule dengan deadlock

$T_1$	$T_2$
<p><b>lock-X</b> on X            write (X)</p> <p>wait for <b>lock-X</b> on Y</p>	<p><b>lock-X</b> on Y            write (X)            wait for <b>lock-X</b> on X</p>

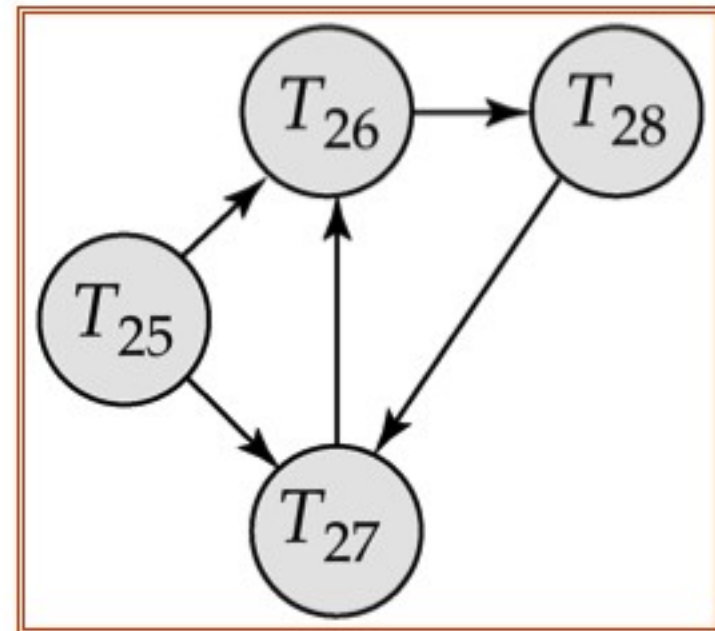
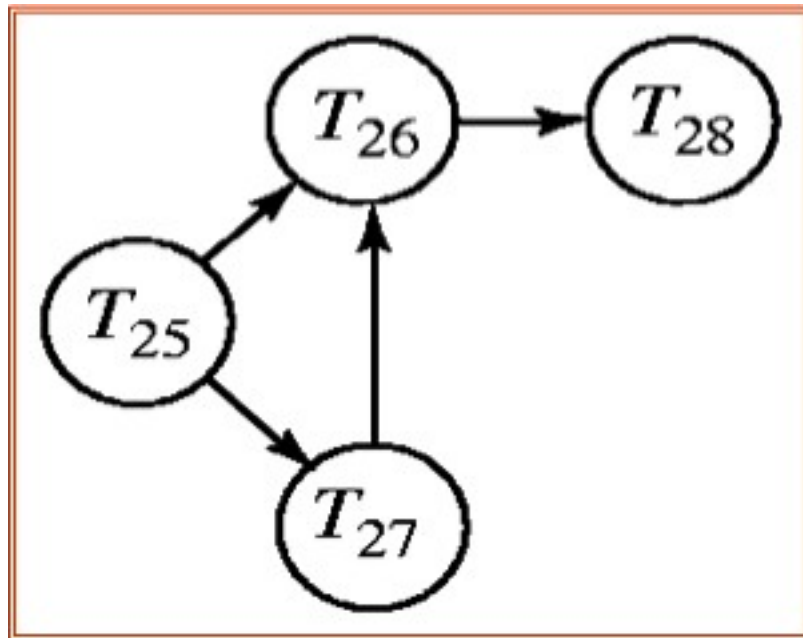
# Deadlock prevention protocol

- Suatu protocol yang menjamin bahwa sistem tidak akan pernah masuk dalam status deadlock.
- Strateginya
  - Membutuhkan bahwa setiap transaksi mengunci semua item data sebelum mengeksekusinya.
  - Memaksa partial ordering dari semua data dan transaksi dapat mengunci item data hanya dalam suatu urutan tertentu dengan partial order (graph based protocol).

# Deadlock Detection

- Deadlock dapat digambarkan sebagai graf wait-for
  - Terdiri dari  $G = (V, E)$
  - $V$  adalah transaksi
  - $E$  adalah pasangan transaksi  $T_i \rightarrow T_j$ 
    - Artinya  $T_i$  menunggu  $T_j$  untuk melepas kunci
- Sistem akan deadlock, jika dan hanya jika graf wait-for berulang.

# Deadlock Detection



- Tanpa cycle

dengan cycle

# Pencegahan Deadlock

- Memberikan prioritas ke transaksi
  - Menggunakan timestamp untuk menyatakan prioritas
- Ti meminta suatu kunci yang dipegang oleh Tj
  - Wait-die:
    - Jika Ti memiliki prioritas lebih tinggi, ia menunggu; selain itu di batalkan
  - Wound-wait
    - Jika Ti memiliki prioritas lebih tinggi, batalkan Tj, selain itu Tj menunggu.
- Setelah dibatalkan, restart timestamp awalnya.