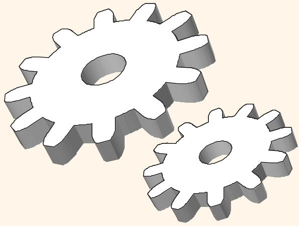
Concurrency Control

Chapter 17

Conflict Serializable Schedules

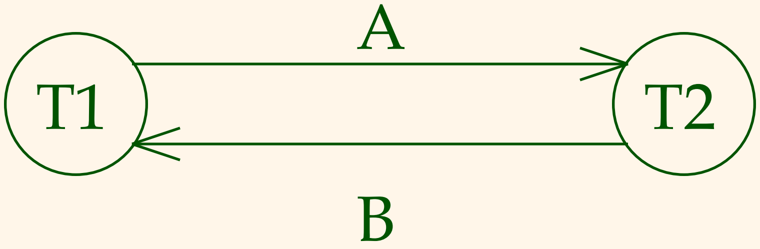


- Two schedules are conflict equivalent if:
  - Involve the same actions of the same transactions
  - Every pair of conflicting actions is ordered the same way
- Schedule S is conflict serializable if S is conflict equivalent to some serial schedule
- Every conflict serializable schedule is serializable but the reverse is not true
- *Precedence graph*: One node per Xact; edge from *Ti* to *Tj* if an action of *Ti* precedes and conflicts with one *of Tj actions*
- Theorem: Schedule is conflict serializable if and only if its dependency graph is acyclic

Example

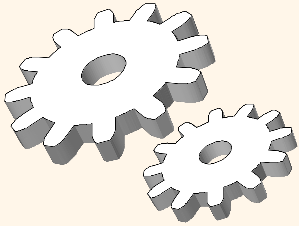
- A schedule that is not conflict serializable:

T1:	R(A), W(A),		R(B), W(B)
T2:		R(A), W(A), R(B), W(B)	



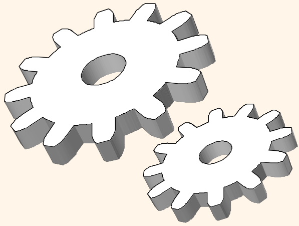
Precedence graph

- The cycle in the graph reveals the problem. The output of T1 depends on T2, and vice-versa.



Review: Strict 2PL

- *Strict Two-phase Locking (Strict 2PL) Protocol*:
  - Each Xact must obtain a S (*shared*) lock on object before reading, and an X (*exclusive*) lock on object before writing.
  - All locks held by a transaction are released when the transaction completes
  - If an Xact holds an X lock on an object, no other Xact can get a lock (S or X) on that object.
- Strict 2PL allows only schedules whose precedence graph is acyclic



Two-Phase Locking (2PL)

- Two-Phase Locking Protocol
  - Each Xact must obtain a S (*shared*) lock on object before reading, and an X (*exclusive*) lock on object before writing.
  - A transaction can not request additional locks once it releases any locks.
  - If an Xact holds an X lock on an object, no other Xact can get a lock (S or X) on that object.