

## 2-Phase Locking Protocol

Use Locks to Ensure Serializable Schedule.

## Problem with Serializability

- Definition: "Equivalent to some serial schedule"
- Calculation of Equivalence takes too long
- Example: 10 transactions in schedule
  - How many serial schedules?
  - $10 \times 9 \times 8 \times \dots \times 1 = 10! = 3,628,800$

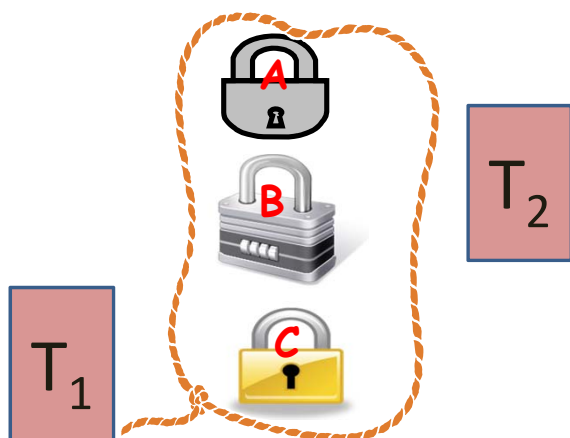
## Solution

- Every transaction follows a protocol
  - protocol = rules of behavior
- Protocol guarantees serializable schedule

## Basic Idea

- At one point in its life, every transaction holds all the locks it will use.
- So so any other transaction must have got its locks on the conflicting items
  - All before, or
  - All after
- 2 Phases are
  - Growing Phase (acquire locks)
  - Shrinking Phase (give them up)

## 2 Phase Locking Protocol



- $T_1$  &  $T_2$  conflict in A,B,C
- Both must get 3 locks to complete.
- If  $T_1$  gets all 3 **now**,  $T_2$  must get them **all before** or **all after**.
- So with respect to **conflict items**, the schedule will be serial.

## Growing Phase:

- Can only LOCK items during this Phase.
- May also UPGRADE
- May also Read & Write once items are locked.
- NO UNLOCKING in this phase
- If Transaction is **successful**, it **gets all** its locks.
- If Transaction is **not successful**, it may **deadlock** or **fail** because it cannot get a lock.

## Shrinking Phase:

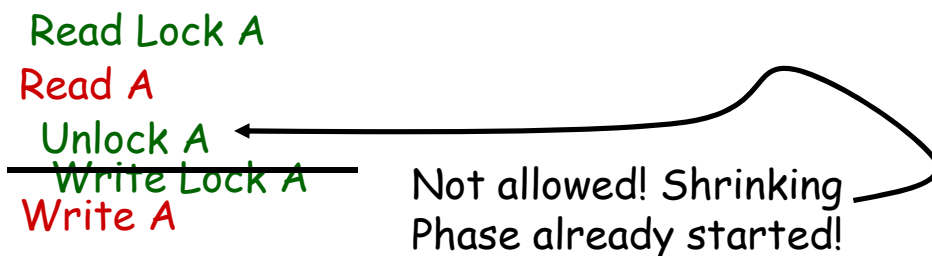
- Can only UNLOCK items during this Phase.
- May also DOWNGRADE
- May still Read & Write items which are still locked.
- Phase begins with FIRST UNLOCK
- NO LOCK after first unlock
  
- This Phase cannot fail, but transaction may still fail because of Dirty Read
- If no Dirty Read, transaction will be serial.

## Variations

- Several variants of this protocol.
- Will look at the basic one first.

## No Upgrade Protocol

- Only read/writelocks. No up/downgrades.
- If transaction reads and writes item, it must start with writelock unless upgrades are allowed.



## TRANSFER 1<sup>st</sup> Example of No-Upgrade

Write lock NumTrans  
 Write lock Bal\_B  
 Write lock Bal\_A  
 Read NumTrans  
 Read Bal\_B  
 Write Bal\_B  
 Read Bal\_A  
 Write Bal\_A  
 Write NumTrans  
 Unlock NumTrans  
 Unlock Bal\_B  
 Unlock Bal\_A

All items are read and written so all locks must be write locks.

This does not allow much interleaving!

## 2<sup>nd</sup> Example of No-Upgrade

### TRANSFER

Write lock NumTrans

Read NumTrans

Write lock Bal\_B

Read Bal\_B

Write Bal\_B

Write lock Bal\_A    Unlock Bal\_B

Read Bal\_A

Write Bal\_A

Unlock Bal\_A

Write NumTrans

Unlock NumTrans

We will Lock Late and  
Unlock Early.

After locking A we  
have all locks so we  
can Unlock B

## 2-PHASE LOCKING PROTOCOL

Upgrading Allowed

## Rules of Upgrading

- You are allowed to upgrade locks from
  - ReadLocks to WriteLocks
  - during the GROWING PHASE
- and to downgrade them from
  - WriteLocks to ReadLocks
  - during the SHRINKING PHASE.
- Constraint: No one else holds Read Lock.
- Downgrade or Unlock starts Shrinking Phase

## Example of Upgradeable Locks

### TRANSFER

Read lock NumTrans

Read NumTrans

Read lock Bal\_B

Read Bal\_B

Upgrade Bal\_B

Write Bal\_B

Read lock Bal\_A

Read Bal\_A

Upgrade Bal\_A

Write Bal\_A

Upgrade NumTrans    Unlock Bal\_A    Unlock Bal\_B

Write NumTrans

Unlock NumTrans

*We will try to Lock Late  
and Unlock Early. But  
there are several ways  
to do it.*