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?
  - $10 \times 9 \times 8 \times \ldots \times 1 = 10! = 3,628,800$
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.

```
Read Lock A
Read A
Unlock A
Write Lock A
Write A
```

Not allowed! Shrinking Phase already started!

TRANSFER 1st 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!
2nd 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

<table>
<thead>
<tr>
<th>TRANSFER</th>
<th>We will try to Lock Late and Unlock Early. But there are several ways to do it.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read lock NumTrans</td>
<td></td>
</tr>
<tr>
<td>Read NumTrans</td>
<td></td>
</tr>
<tr>
<td>Read lock Bal_B</td>
<td></td>
</tr>
<tr>
<td>Read Bal_B</td>
<td></td>
</tr>
<tr>
<td>Upgrade Bal_B</td>
<td></td>
</tr>
<tr>
<td>Write Bal_B</td>
<td></td>
</tr>
<tr>
<td>Read lock Bal_A</td>
<td></td>
</tr>
<tr>
<td>Read Bal_A</td>
<td></td>
</tr>
<tr>
<td>Upgrade Bal_A</td>
<td></td>
</tr>
<tr>
<td>Write Bal_A</td>
<td></td>
</tr>
<tr>
<td>Upgrade NumTrans</td>
<td></td>
</tr>
<tr>
<td>Unlock Bal_A</td>
<td></td>
</tr>
<tr>
<td>Unlock Bal_B</td>
<td></td>
</tr>
<tr>
<td>Write NumTrans</td>
<td></td>
</tr>
<tr>
<td>Unlock NumTrans</td>
<td></td>
</tr>
</tbody>
</table>