Locking

What is locked?

• Some unit of database
• Whole Database
  – Inhibits concurrency
• Table of Database
  – Less inhibition but still severe limit
• Record in Table
  – Most common level
• Field in Record
  – May have high overhead
Why lock anything?

- Enforce transaction concept
- Database often inconsistent in midst of transaction
  - example: transfer money between accounts takes 2 steps.
  - DB only accurate after money deducted from A and added to B
    - You can only read A or B before I write either or after I write both
    - Otherwise information is inconsistent

How lock works

- Transaction needs lock before read or write
- Transaction must ask for lock
- If it does not get the lock
  - maybe another transaction already holds the lock
    - it is suspended
    - put on queue waiting for lock
- When transaction releases lock
  - Some suspended transaction awakened and given lock.
Lock State Diagram

Multi-transaction Rules

- Multiple Read Locks OK
  - Called Shared Lock
  - Several transactions can read same info
- Write Lock must be only lock on item
  - 2 Write Locks lead to race condition: lost update
  - Sharing with Read lock leads to dirty read or non-repeatable read.
ReadLock state diagram

- Available
- Not available
- Readlocks granted & released
- Last WriteLock unlocked
- $T_x$ gets Write Lock

WriteLock state diagram

- Available
- Not available
- Last Lock unlocked
- $T_x$ gets Read or Write Lock
### ReadLock WriteLock Matrix

<table>
<thead>
<tr>
<th>First Transaction Holds</th>
<th>Second Transaction Wants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Lock</td>
<td>Read Lock</td>
</tr>
<tr>
<td></td>
<td><img src="emoji" alt="Smiley" /></td>
</tr>
<tr>
<td>Write Lock</td>
<td><img src="emoji" alt="Sad" /></td>
</tr>
</tbody>
</table>

### Lock Upgrades/Downgrades

- Some schemes add this feature
- If transaction holds Readlock it can try for Writelock **without releasing** Readlock first
  - It will sleep and wait on queue if it cannot get the upgrade
- If transaction holds Writelock it can exchange for Readlock **without releasing** Writelock first.
  - Never a problem of not getting readlock.
- Aims
  - Increase concurrency
  - Reduce rollbacks
### Readlock Writelock Upgrade Matrix

<table>
<thead>
<tr>
<th>First Transaction Holds</th>
<th>Second Transaction Wants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Lock</td>
<td>Read Lock</td>
</tr>
<tr>
<td></td>
<td>Write Lock</td>
</tr>
<tr>
<td></td>
<td>Upgrade to Write Lock</td>
</tr>
<tr>
<td>Write Lock</td>
<td></td>
</tr>
</tbody>
</table>

N/A because 2nd needs to have a readlock to upgrade, but 1st has a write lock.