**Lock**

**Binary Lock**

- Lock (item)
- Unlock (item)

**Lock (item):**

```
Start: if (Lock(item) == 0)
    { // item is unlocked
        Lock(item) = 1
    }
else // item is locked.
    & wait (until Lock(item) == 0) // process goes to sleep
    Lock manager wakes up the transaction
    when the lock is released.
    goto start
```

**Unlock (item):**

```
Unlock (item):
    Lock (item) = 0
    Signal (activate) Lock Manager
    so that Lock Manager will wake up
    transactions that are waiting for this item.
```
Read Lock & Write Lock

Mode:
Read Locked (Shared) with no. of readers.
Write Locked (Excluding): 0 or 1
Unlocked

ReadLock(item), WriteLock(item), Unlock(item)

ReadLock(item):
Start: if (Lock(item) == 'Unlocked') {
  Lock(item) = 'Readlocked'
  no-of-readers = 1
} else if (Lock(item) == 'Readlocked') {
  no-of-readers++
} else // item is write locked
  wait (until Lock(item) == 'Unlocked')
  Lock Manager wakes up the transaction
  goto start

WriteLock(item):
Start: if (Lock(item) == 'Unlocked') {
  Lock(item) = 'Write locked'
} else
  wait (until Lock(item) == 'Unlocked')
  Lock Manager wakes up the transaction
  goto start

Unlock(item)
Unlock (item): (Read lock & Write lock)

if (Lock (item) == 'Write locked')
{
    Lock (item) = 'Unlocked'
    Signal (activate) Lock Manager
    so that it will wake up transactions
    that are waiting for this item
}
else
{
    "read locked"
    No_of_readers--
    if (No_of_readers == 0)
    {
        Lock (item) = 'Unlocked'
        Signal (activate) Lock Manager
        so that it will wake up transactions
        that are waiting for this item
    }
}

Granularity of Locks
- Server level
  - Database Level
    - Object (table) level
      - Row level
        - Column level
**Intentions Locks** (meaningful in a hierarchy of objects)

IS (intention shared)
- placed at a higher level unit
- and
  - Shared lock can be placed at a lower level

E.g.
- Company DB — IS
  - Employee Table — Shared lock

IE (intention exclusive)

E.g.
- T4: Employee Table — IX
  - on a specific row — Exclusive lock.

T6: Read on some row of Employee Table
  - readlock(row)
  - Check the status on Employee Table: IX

To understand why all may not be available for read.
Lost update Example

\[ T_1 \]
not following 2-PL

\[ + \text{Lock}(X) \]
\[ \text{Read}(X) \]
\[ X = X - N \]
\[ **\text{Write}(X) \]
\[ - \text{Unlock}(X) \]

\[ + \text{Lock}(Y) \]
\[ \text{Read}(Y) \]
\[ Y = Y + N \]
\[ - \text{Unlock}(Y) \]

\[ T_2 \]

\[ + \text{Lock}(X) \]
\[ \text{Read}(X) \]
\[ X = X + M \]
\[ - \text{Unlock}(X) \]

\[ + \text{Lock}(Y) \]
\[ \text{Write}(Y) \]
\[ - \text{Unlock}(Y) \]

\[ T_3 \]
Following 2-PL

\[ + \text{Lock}(X) \]
\[ \text{Read}(X) \]
\[ X = X - N \]
\[ - \text{Unlock}(X) \]

\[ + \text{Lock}(Y) \]
\[ \text{Write}(Y) \]
\[ - \text{Unlock}(Y) \]

Two phase Locking Protocol

Each transaction has two distinct phases:

1. Growth phase
   - Acquires locks for DB items
   - w/o releasing any locks

2. Shrink phase
   - Releases locks of DB items
   - w/o acquiring new locks

Benefits:
- Any schedule made of transactions that follows two-phase locking protocol is guaranteed to be conflict serializable.
- However, a deadlock may occur with two-phase locking protocol transactions.
**T1**
- **Conservative 2PL**
  - Lock(x)
  - Lock(y)
  - read(x)
  - x = x - N
  - write(x)
  - Unlock(x)
  - Unlock(y)

**T3**
- **Strict 2PL**
  - RLock(A)
  - Read(A) in Growth Phase
  - WLock(B)
  - Read(B) in Shrink Phase
  - B += B + 100
  - Unlock(A)
  - Write(B) in Shrink Phase
  - Unlock(B)

**T3**
- **Rigorous 2PL**
  - RLock(A)
  - Read(A)
  - WLock(B)
  - Read(B)
  - B = B + 100
  - Unlock(B)
  - Unlock(A)

---

**Deadlock**

**Using Basic 2PL**

- **T5 (Ryan)**
  - Lock (Washer)
  - read (w)
  - Lock (Dryer)

- **T6 (Elena)**
  - Lock (Dryer)
  - read (d)
  - Lock (Washer)

**Deadlock State**

- **SP**
  - Write (Dryer)
  - Unlock (Washer)
  - Unlock (Dryer)
  - Unlock (Washer)
  - **SP**
  - Unlock (Dryer)
  - Unlock (Washer)
Deadlock Prevention

With Time stamp (chronological order)

<table>
<thead>
<tr>
<th>T1</th>
<th>elder</th>
<th></th>
<th>T5 (Ryan)</th>
<th>Younger</th>
<th>T6 (Elena)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2</td>
<td></td>
<td></td>
<td>Lock (Washer)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td></td>
<td></td>
<td>Read (Washer)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td></td>
<td></td>
<td>Lock (Dryer)</td>
<td>wait</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td></td>
<td></td>
<td>Read (Dryer)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) Wound-wait

<table>
<thead>
<tr>
<th>T5 (Ryan)</th>
<th>T6 (Elena)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td></td>
</tr>
</tbody>
</table>

T6 gets re-started
with previous timestamp T6

T6 gets re-started
with the same timestamp
Timestamp Ordering

(no locks are used)

For each DB item (i.e. DB item in memory)

- a read time stamp (RTS)
- and a write time stamp (WTS) are associated
  with the item.

RTS: for an item:

The TS of the youngest transaction that read the item.

WTS: for an item:

The TS of the youngest transaction that performed write operation on the item.

\[
\begin{align*}
T_1 & \text{ eldest} \\
T_2 \\
T_3 \\
T_4 & \text{ youngest}
\end{align*}
\]

\[
\begin{align*}
t_1 &: T_1. \text{Read}(A) \rightarrow 1 \\
t_2 &: T_4. \text{Read}(A) \rightarrow 4 \\
t_3 &: T_2. \text{Read}(A) \rightarrow 4 \\
t_4 &: T_5. \text{Write}(A) \rightarrow WTS 5 \\
t_5 &: T_9. \text{Write}(A) \rightarrow 9 \\
t_6 &: T_7. \text{Write}(A) \rightarrow 9
\end{align*}
\]

\[
\begin{align*}
\text{RTS} & \rightarrow 8 \rightarrow 4 \\
\text{WTS} & \rightarrow 5 \rightarrow 9
\end{align*}
\]