Basic Two phase Locking Protocol

Each transaction has two distinct phases

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

  Then

- Shrink phase
  Releases locks of DB item
  w/o acquiring new lock.

Benefit: Any schedule made of transactions
that follow 2-phase locking protocol,
is guaranteed to be serializable.

However a deadlock can occur with 2-phase locking
protocol transactions.

Deadlock

Ti (Ryan) - 2PL

Growth

- Lock (washer) +
- Read (washer) +
- Lock (Dryer) ?
- Read (Dryer)
- Unlock (washer)
- Unlock (Dryer)

Shrink

T2 (Elena) - 2PL

Growth

- t2 Lock (Dryer)
- t4 Read (Dryer)
- Unlock (washer)
- Unlock (Dryer)
- Unlock (washer)

Shrink
Basic Two-Phase Locking Protocol

Each transaction has two distinct phases

- Growth phase
  - Acquires locks
  - Without releasing any lock

Then

- Shrink phase
  - Releases locks
  - Without acquiring new locks

Benefit:

Any schedule made out of transaction that follow 2-phase locking protocol is guaranteed to be serializable.

It may encounter deadlock.

\[ T_1 \] (Basic)

- Lock (x)
- Read (x)
- Lock (y)
- Read (y)
- Write (y)
- Unlock (y)
- Unlock (x)

\[ T_2 \] (Conservative)

- Lock (x)
- Lock (y)
- Read (x)
- Write (x)
- Unlock (x)
- Read (y)
- Write (y)
- Unlock (x)
- Unlock (y)
T1
Conservative 2PL
+ Lock(X)
+ Lock(Y)
read(X)
X = X - N
write(X)
- Unlock(X)
Read(Y)
Y = Y + N
Write(Y)
- Unlock(Y)

This will not lead to Deadlock

T3
Strict 2PL
+ RLock(A)
Read(A)

+ WLock(B)
Read(B)
B = B + 100
- Unlock(A)
Shrink Phase
Write(B)
Commit
- Unlock(B)

Rigorous 2PL
+ RLock(A)
Read(A)
+ WLock(B)
Read(B)
B = B + 100
Write(B)
Commit
- Unlock(B)
- Unlock(A)

DeadLock
Using Basic 2PL

T5 (Ryan)
lock (Washer)

T6 (Elena)
lock (Dryer)

T2

T3

T4

T5

lock (Dryer)
wait

lock (Washer)
wait

Deadlock State

T1

SP

Unlock (Washer)
Unlock (Dryer)

SP

Unlock (Washer)
Unlock (Dryer)
Deadlock Prevention

With Time stamp (chronological order)

\[
\begin{align*}
T_1 & \quad \text{elder} \\
T_2 & \\
T_3 & \\
T_4 & \\
T_5 & \\
T_6 & \text{youngest.} \\
\end{align*}
\]

a) Wait - Die

\[
\begin{align*}
T_5 \text{ (Ryan)} & \quad \text{Younger} \\
T_6 \text{ (Elena)} & \\
\end{align*}
\]

\[
\begin{align*}
t_1 & \quad \text{Lock (Washer)} \\
t_2 & \quad \text{Read (Washer)} \\
t_3 & \quad \text{Read (Washer)} \\
t_4 & \quad \text{Lock (Dryer)} \quad \text{wait} \\
t_5 & \\
t_6 & \quad \text{Read (Dryer)} \\
\end{align*}
\]

\[\text{Unlock (Dryer)} \quad \text{rollback} \]

T6 gets restarted with previous timestamp T6

b) Wound - Wait

\[
\begin{align*}
T_5 \text{ (Ryan)} & \quad T_6 \text{ (Elena)} \\
\end{align*}
\]

\[
\begin{align*}
t_1 & \quad \text{Lock (Washer)} \\
t_2 & \quad \text{Read (Washer)} \\
t_3 & \quad \text{Read (Washer)} \\
t_4 & \quad \text{Lock (Dryer)} \quad \text{abort} \quad \text{T6 gets aborted} \\
t_5 & \\
t_6 & \quad \text{Read (Dryer)} \\
\end{align*}
\]

\[\text{Unlock (Dryer)} \quad \text{rollback} \]

T6 gets restarted with the same timestamp
Timestamp Ordering
(no locks are used)

For each active 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.

\[
\begin{align*}
T_1 \quad \text{eldest} \\
\downarrow \\
T_2 \\
T_3 \quad T_4 \quad \text{youngest} \\
\uparrow \quad \text{time}
\end{align*}
\]

\[
\begin{align*}
t_1 & \quad T_{1}. \text{Read}(A) \quad 1 \\
t_2 & \quad T_{4}. \text{Read}(A) \quad \checkmark \quad 4 \\
t_3 & \quad T_{2}. \text{Read}(A) \quad 4 \\
t_4 & \quad T_{5}. \text{Write}(A) \quad \text{WTS} \quad 5 \\
t_5 & \quad T_{9}. \text{Write}(A) \quad 9 \\
t_6 & \quad T_{7}. \text{Write}(A) \quad 9
\end{align*}
\]