(3) (S,U,X) locks

U lock --- update lock. For an update access the transaction first acquires a U-lock and then promote it to X-lock.

Purpose: shorten the time of exclusion, so as to boost concurrency degree, and reduce deadlock.

	NL	S	U	X
NL	Y	Y	Y	Y
S	Y	Y	Y	N
U	Y	Y	N	N
X	Y	N	N	N

X (S,X) (S,U,X)

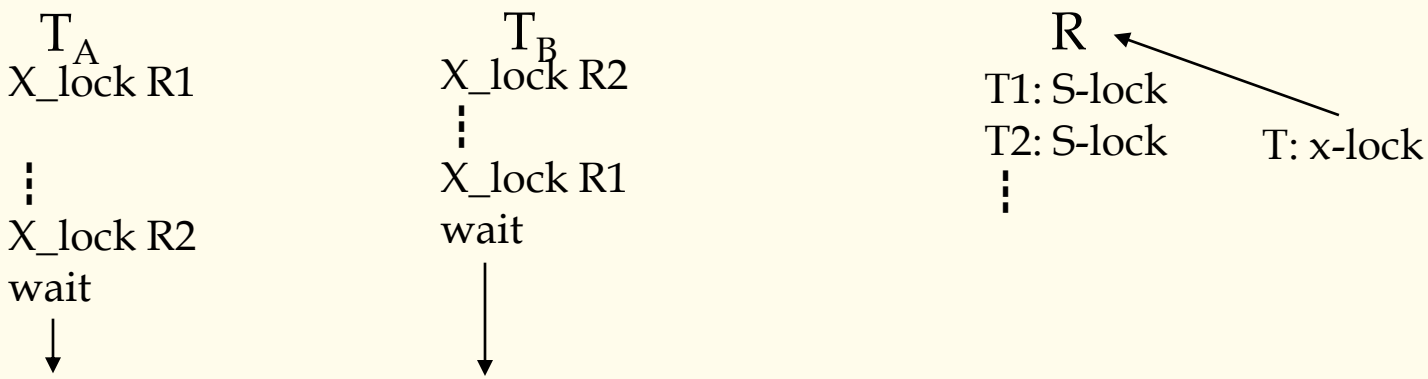
Concurrency degree →

Overhead →

4.6.4 Deadlock & Live Lock

Dead lock: wait in cycle, no transaction can obtain all of resources needed to complete.

Live lock: although other transactions release their resource in limited time, some transaction can not get the resources needed for a very long time.



- Live lock is simpler, only need to adjust schedule strategy, such as FIFO
- Deadlock: (1) Prevention(don't let it occur); (2) Solving(permit it occurs, but can solve it)

(1) Deadlock Detection

1) Timeout: If a transaction waits for some specified time then deadlock is assumed and the transaction should be aborted.

2) Detect deadlock by wait-for graph G=<V,E>

   V : set of transactions $\{T_i \mid T_i$ is a transaction in DBS (i=1,2,…n)$\}$

   E : $\{<T_i,T_j> \mid T_i$ waits for T_j $(i \neq j)\}$

- If there is cycle in the graph, the deadlock occurs.
- When to detect?

1) whenever one transaction waits.

2) periodically

- What to do when detected?

1) Pick a victim (youngest, minimum abort cost, …)
2) Abort the victim and release its locks and resources
3) Grant a waiter
4) Restart the victim (automatically or manually)

(2) Deadlock avoidance

1) Requesting all locks at initial time of transaction.
2) Requesting locks in a specified order of resource.
3) Abort once conflicted.
4) Transaction Retry