Concurrency Control

Lock-Based Protocol

Mode of Lock

- Exclusive (X) : read & write, lock-X
- Shared (S) : read only, lock-S

Granting of Locks

1. When access to data, send lock request to lock manager of data
2. lock manager check **lock-compatibility** and accept the request
    - lock-compatibility matrix

	S	X
S	true	false
X	false	false

    - lock manager record a transaction that held lock and based on these data structure(maybe a hash map), decide compatibility

Example

```
lock-S (A)
read (A)
unlock (A)
lock-S (B)
read (B)
unlock (B)
```

- but this process cuase siginificant problem : can't guarantee serializability
    - because, another transaction may lock-X (A) and write (A) and unlock (A) before lock-S (B) → this is violation of serializability

Two-Phase Locking Protocol(2PL)

- Devide transaction into two phases
    1. Growing Phase : transaction may obtain locks but not release any lock
    - can acquire new lock(S + X)
    - upgrade lock(S -> X)
    - can't release lock
    2. Shrinking Phase : transaction may release locks but not obtain any lock
    - can release lock(S + X)
    - downgrade lock(X -> S)
    - can't acquire new lock

Example

```
   T1          T2           T3
lock-X (A) |            |
read (A)   |            |
lock-S (B) |            |
read (B)   |            |
write (A)  |            |
unlock (A) |            |
           |lock-X (A)  |
           |read (A)    |
           |write (A)   |
           |unlock (A)  |
           |lock-X (A)  |
           |read (A)    |
           |write (A)   |
           |unlock (A)  |
           |            |lock-S (A)
           |            |read (A)
```

- T2 can't read (A) before T1 unlock (A)
- also T3 can't read (A) before T2 unlock (A)

Features of 2PL

- Guarantees (conflict)serializability
- Not ensure freedom from deadlock
    - starvation also possible
- Potential Cascading Rollback
    - if transaction T1 aborts, then all transaction that T1 has updated must be aborted
    - because, T1 may have updated data that T2 has read

Strict / Rigorous 2PL

- Strict 2PL : transaction hold write locks until commit or abort
    - Avoid Cascading Rollback
- Rigorous 2PL : transaction hold all locks until commit or abort
    - more strict than strict 2PL