Transactions / Serializability

abstraction of concurrent programming.

```
lock (mtx_a)
lock (mtx_b)
operate (a)
operate (b)
unlock (a)
unlock (b)
```

→ manual synchronization

```
lock (mtx_a)
operate (a)
unlock (mtx_a)
lock (mtx_b)
- - - (b)
unlock (b)
```

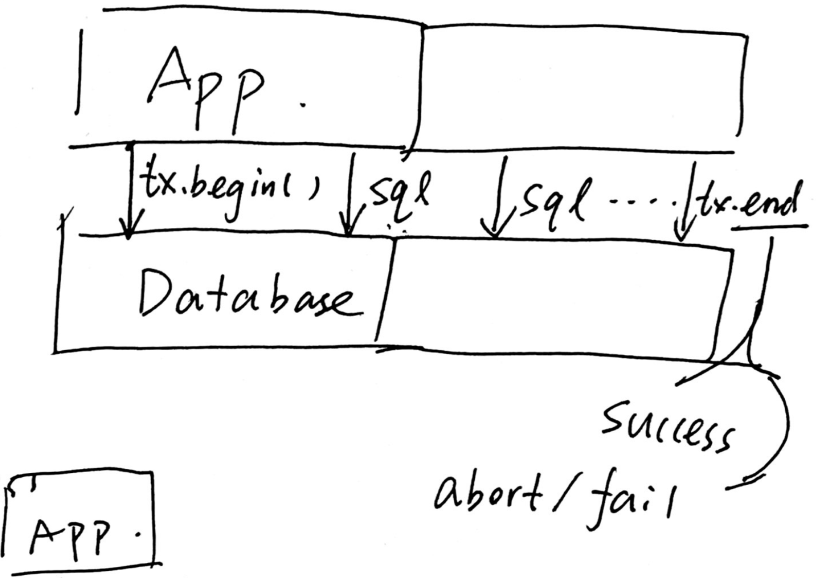
Dynamic / Static

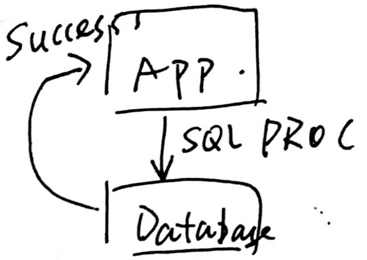
```
Transaction {

    a....

    b....

}
```



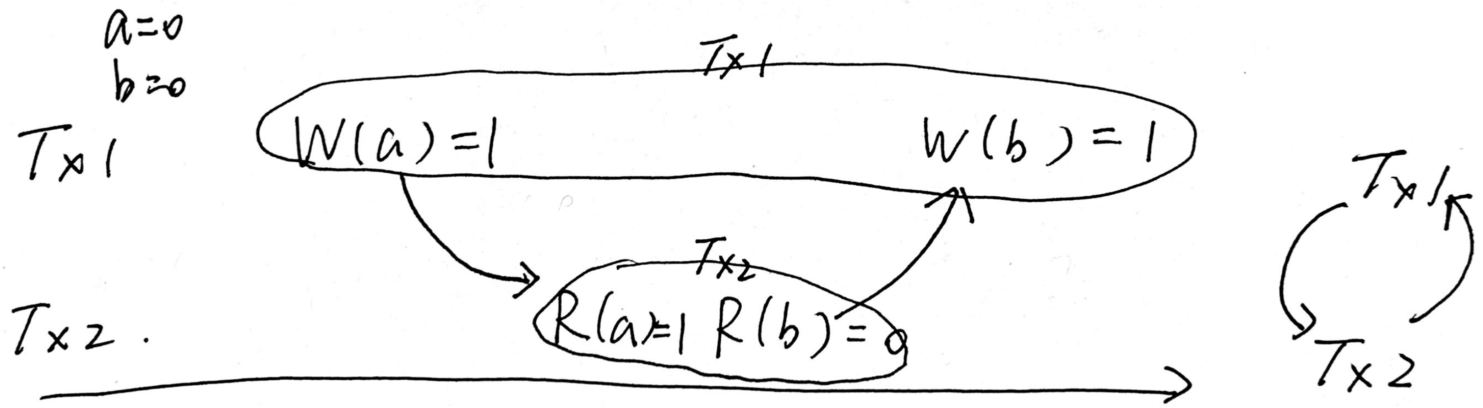


Serializability.

- Multiple object
- Schedule
- Global order.

Linearizability .: completion to issuing.

Strit-serializability



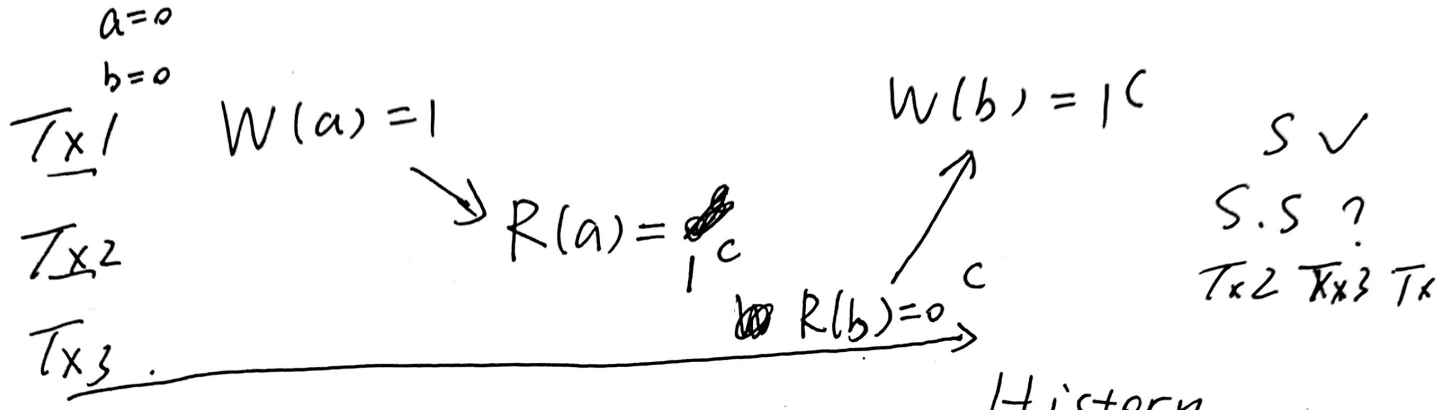
conflict graph.

serializable. - - -

precedence — -

History vs. Schedule.

Client — system ..



S ✓

S.S ?

Tx2 Tx3 Tx1

History.

Tx3 Tx1 Tx2

Implementing Serializability.

lock.
- entire database.
- lock. dataset
  - read-set
  - write-set

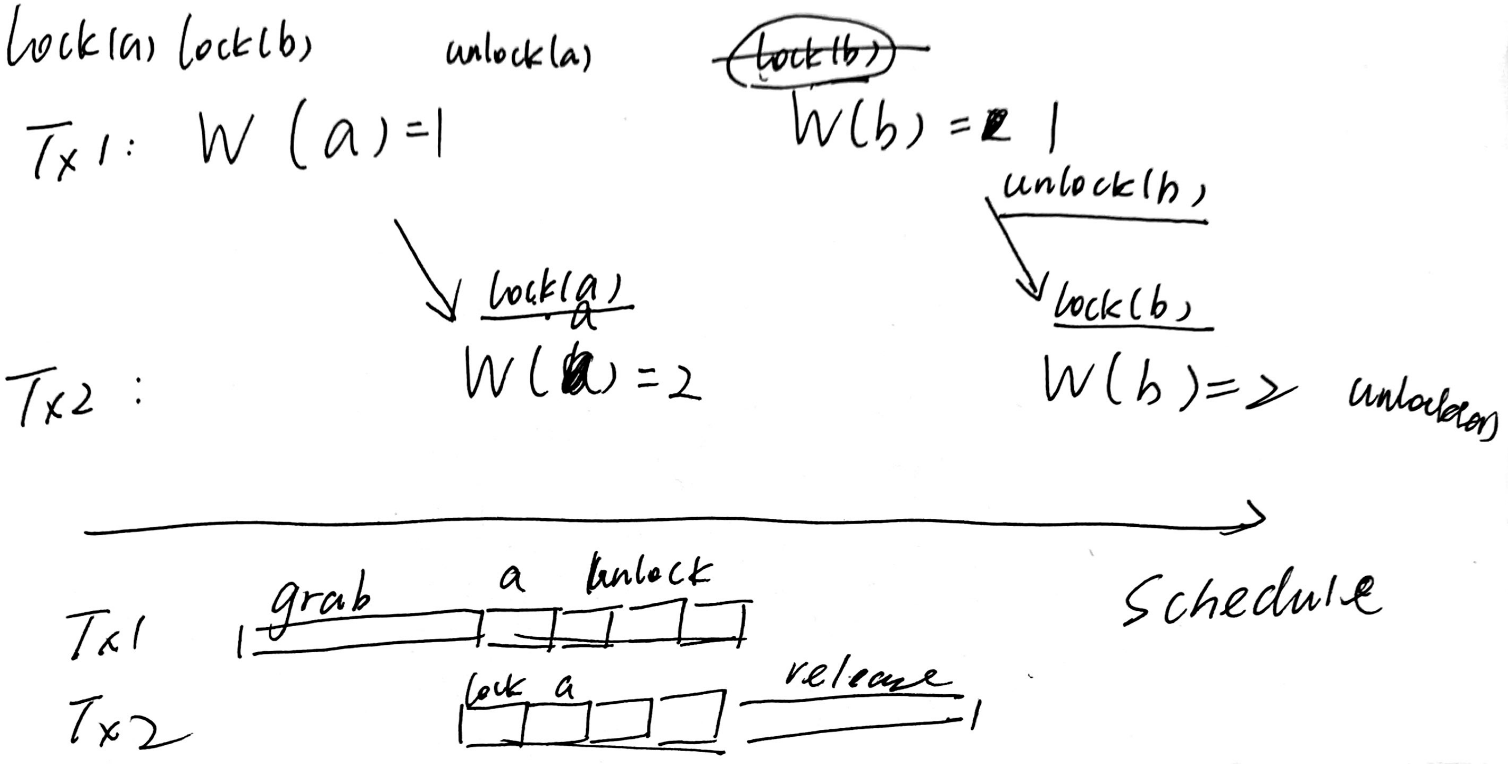
deadlocks.
- grab locks
- access.
- release.

① lock a access a lock b .. b ..

② access a release a . release b.

③ Both?

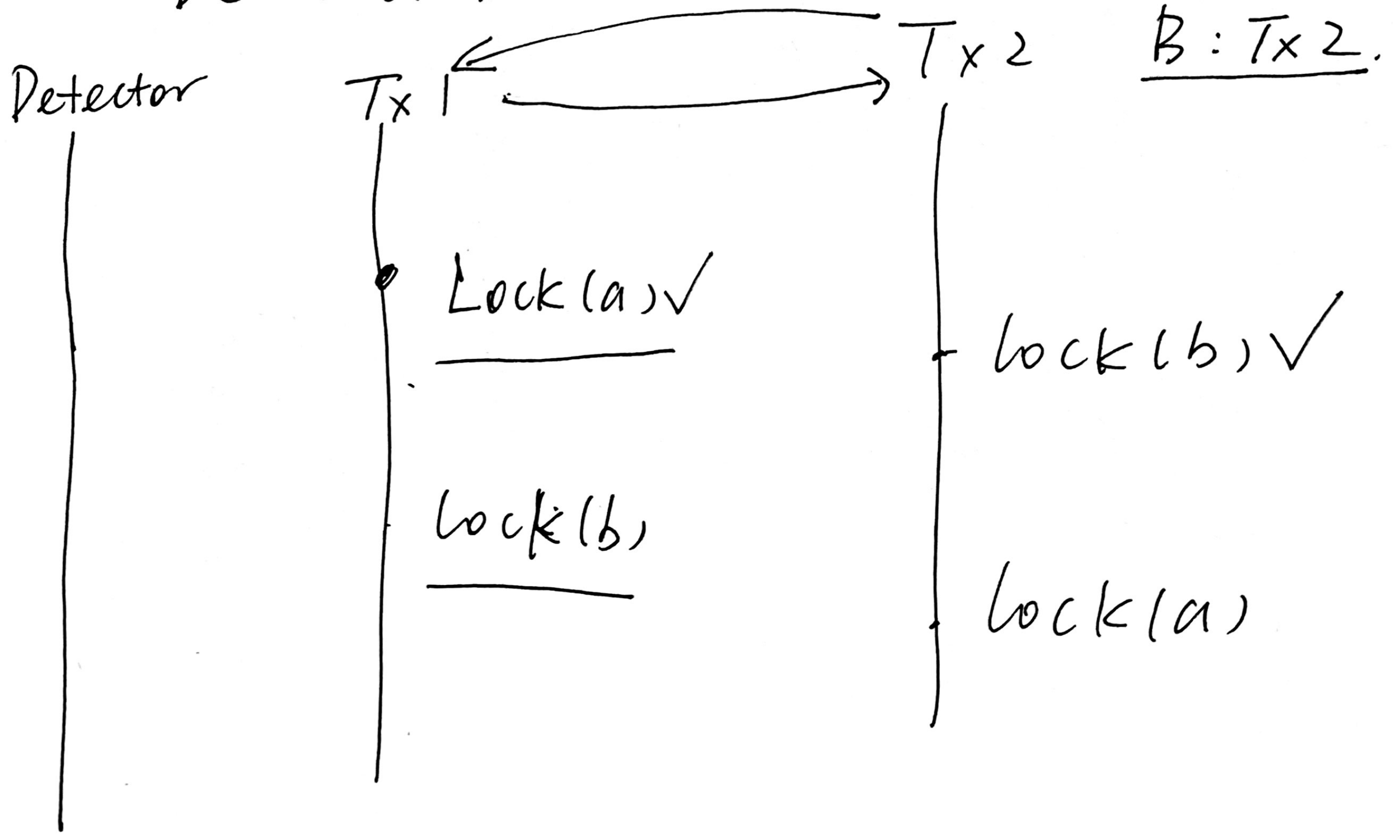
2PL enforcing serial-----

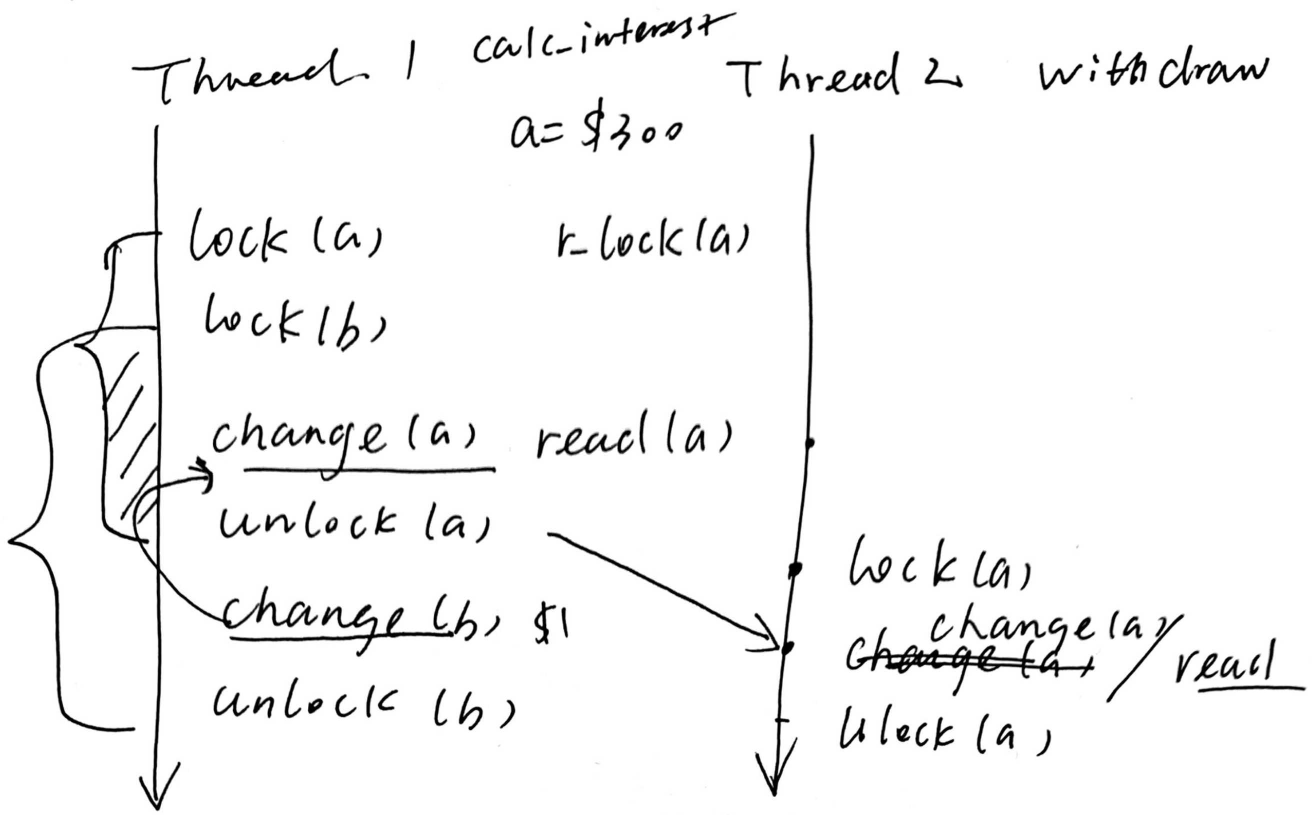


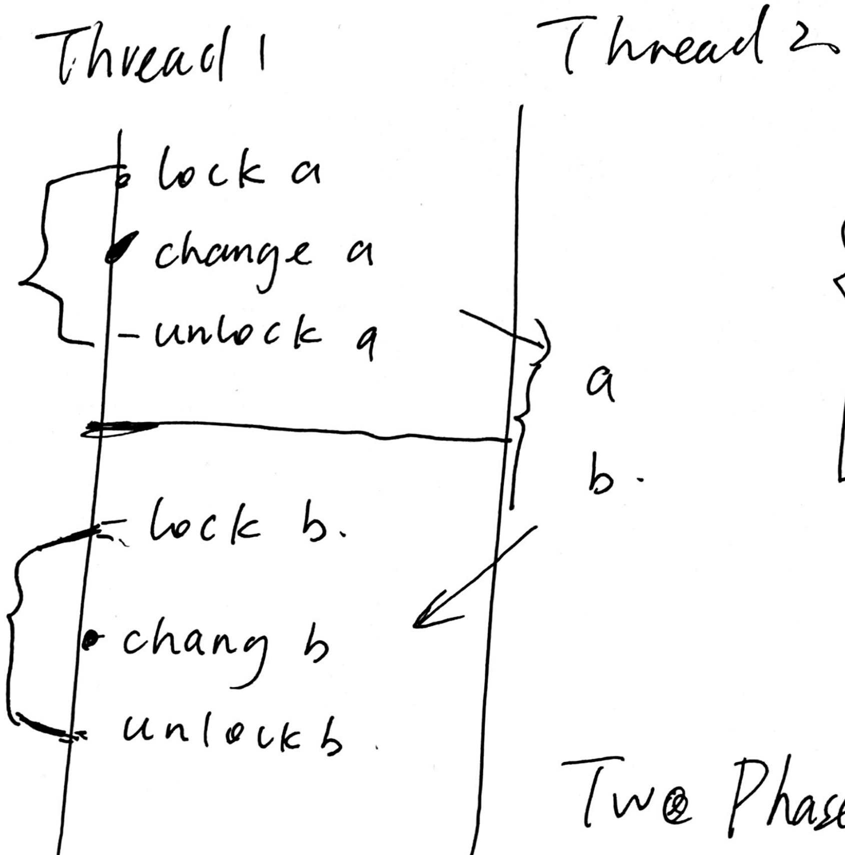
Deadlock.

A: Tx1

B: Tx2.







1° Grab lock in the future

⇒ cannot release lock

2° Release lock in the past

⇒ cannot grab lock in the future

Two Phases

1. locking.

2. release.