Transactions / Serializability abstraction of coconcurrent programming. lock (at too mtx-a) > manual. lock (mtx-b) synchronization roperate (a) Lock (mtx-a) operate (b) operate (a) unlock (a) unlock (mtx-a) (ock (mtx-b) unlock (b) unlock (b) Dynamic / Static Transaction ? tx.begin() | Sql | Sql -- . ftx.end Database Success abort/fail J. SQL PROC

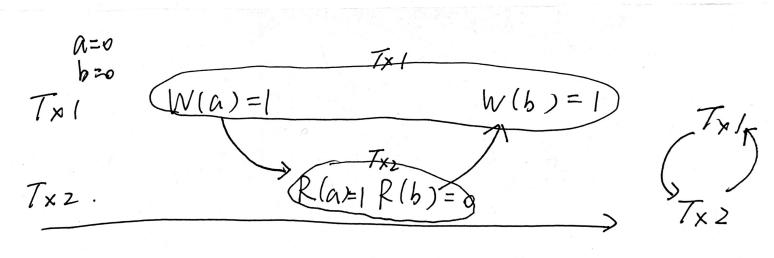
Serializability.

Multiple object

Schedule

Global order.

Linearizability: completion to issuing. Strit-serializability



Conflict graph.

Serializable. - -
precedence --

History vs. Schedule	•
Client Systen	<u>1</u>
$ \begin{array}{ccc} \alpha = 0 \\ b = 0 \end{array} $ $ \begin{array}{ccc} T \times 1 & W(\alpha) = 1 \end{array} $ $ \begin{array}{ccc} T \times 2 & P(\alpha) = 1 \end{array} $ $ \begin{array}{ccc} T \times 3 & . \end{array} $	$W(b) = 1^{C}$ $S = 1^{C}$ $T_{K2} = T_{K3} = T_{K1}$ $H \text{ is tory}$
	TN3 TX # TX2
Implementing Serial	izahility.
bock. < entire data	base set read-set write-set
deadlocks of grab locks. access. release.	1 Lock a accessa, lock b b 2 access a release a release b.
	(3) Both?

2PL enforcing serial ---. Lockia, lockib, (bock 16) unlock(a) W(b) = 2 1 Tx1: W (a)=1 unlock(h) Y Lock(a) bock (b) W(b) = 2 unlocker W(b)=2 Tx2: bulock Schedule 7×2 A: Tx1 Deadlock. Tx2 B:Tx2. TX Detector Lock (a) lock (b) lock(a)

Thread 1 calcinterest
Thread 2 with cham

a=\$300

lock(a) relock(a)

bock(b)

change(a) read(a)

unlock(a)

change(b) \$1

change(b) \$1

unlock(a)

lock(a)

change(a)

change(a)

change(a)

lock(a)

lock(a)

lock(a)

