$r_1(X)$

## conflicts

- access same data
- different transactions
- one must be a write

## Correct Execution

- serial schedule (no interleaving)
- serializable schedule is equiv. to some serial sched.

---

Equivalent?
- same results
- conflict equiv.
  - order of two conflicting ops. is the same in both sched.

**Testing for conflict serializability (Algorithm 21.1):**

1. Create a directed graph with a node for each transaction T participating in S.

2. For each conflict between an operation in T_i and an operation in T_j, add a directed edge from T_i to T_j if the conflicting operation comes first in T_i, otherwise add an edge from T_j to T_i.

3. The schedule is serializable iff the graph has no cycles.

$T_1$ : $r_1(x)\ w_1(x)$

$T_2$ : $r_2(y)\ w_2(y)\ r_2(x)\ w_2(x)$

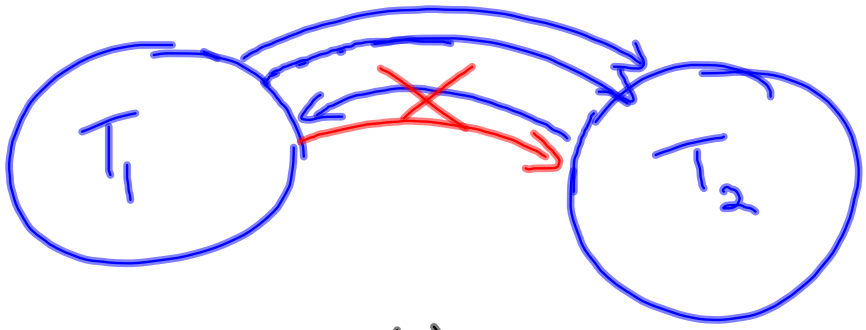
conflicts

$r_1(x)$ , $w_2(x)$

$r_2(x)$ , $w_1(x)$

$w_1(x)$ , $w_2(x)$

$S_1$ : $r_1(x)\ r_2(y)\ w_2(y)\ r_2(x)\ w_1(x)\ w_2(x)$




$S_1'$ : $r_1(x)\ r_2(y)\ w_2(y)\ w_1(x)\ r_2(x)\ w_2(x)$

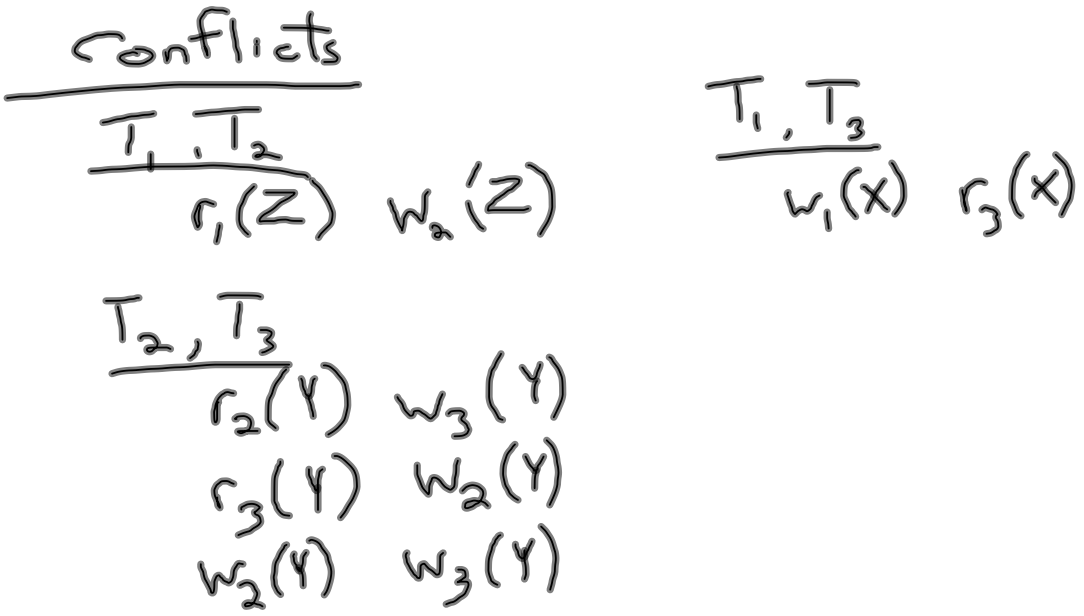
# Testing for conflict serializability (Algorithm 21.1):

1. Create a directed graph with a node for each transaction T participating in S.
2. For each conflict between an operation in T_i and an operation in T_j, add a directed edge from T_i to T_j if the conflicting operation comes first in S, otherwise add an edge from T_j to T_i.
3. The schedule is serializable iff the graph has no cycles.

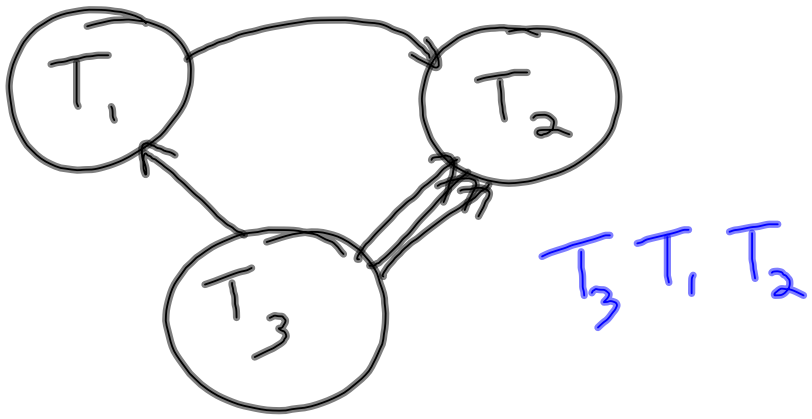
Transactions:
T_1: r_1(X); r_1(Z); w_1(X);
T_2: r_2(Z); r_2(Y); w_2(Z); w_2(Y);
T_3: r_3(X); r_3(Y); w_3(Y);

Schedules:
S_1: r_1(X); r_2(Z); r_1(Z); r_3(X); r_3(Y); w_1(X); w_3(Y); r_2(Y); w_2(Z); w_2(Y);
S_2: r_1(X); r_2(Z); r_3(X); r_1(Z); r_2(Y); r_3(Y); w_1(X); w_2(Z); w_3(Y); w_2(Y);
S_3: r_2(Z); r_1(X); r_2(Y); w_2(Z); r_1(Z); w_1(X); r_3(X); w_2(Y); r_3(Y); w_3(Y);

1. Identify the conflicts between all transactions.
2. Determine if each schedule is serializable.
3. If the schedule is serializable, what is an equivalent serial schedule?




S_1: r_1(X); r_2(Z); r_1(Z); r_3(X); r_3(Y); w_1(X); w_3(Y); r_2(Y); w_2(Z); w_2(Y);



SQL

- START TRANSACTION
- BEGIN
- COMMIT
- ROLLBACK