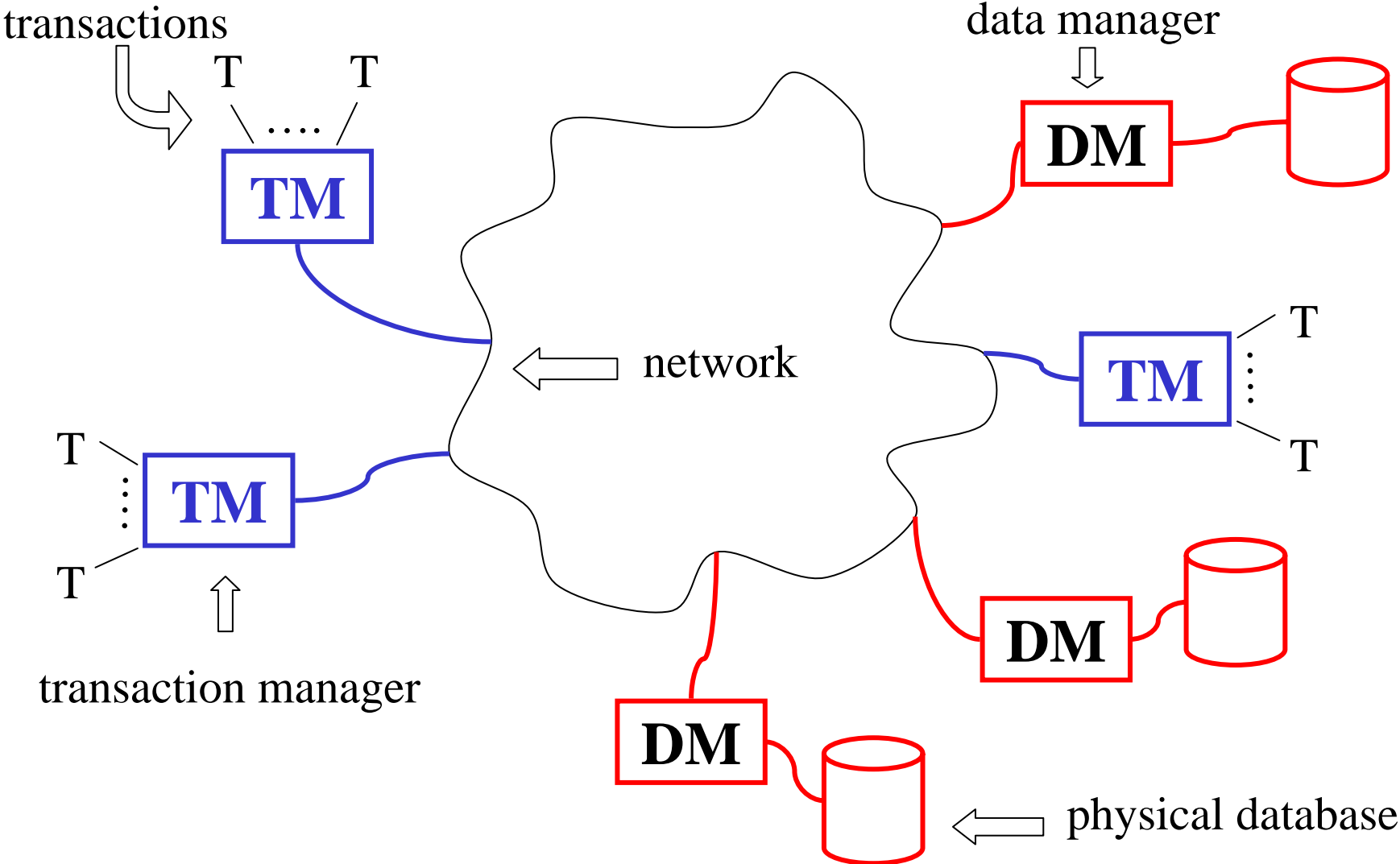


# Distributed DBMS Model



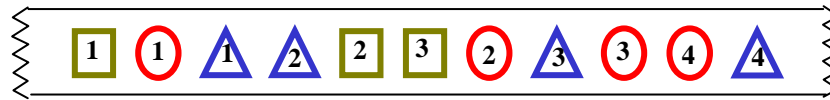
# Serialization

**T<sub>1</sub>** : ○ 1 ○ 2 ○ 3 ○ 4

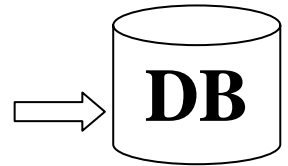
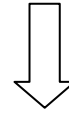
**T<sub>2</sub>** : □ 1 □ 2 □ 3

**T<sub>3</sub>** : △ 1 △ 2 △ 3 △ 4

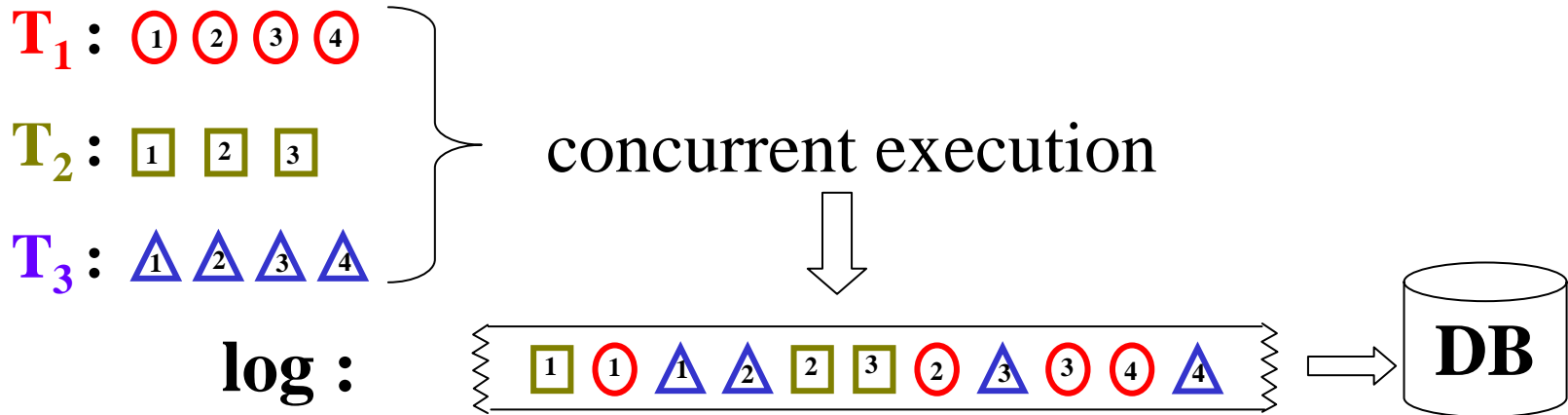
**log :**



concurrent execution



# Serialization



## OPERATIONS

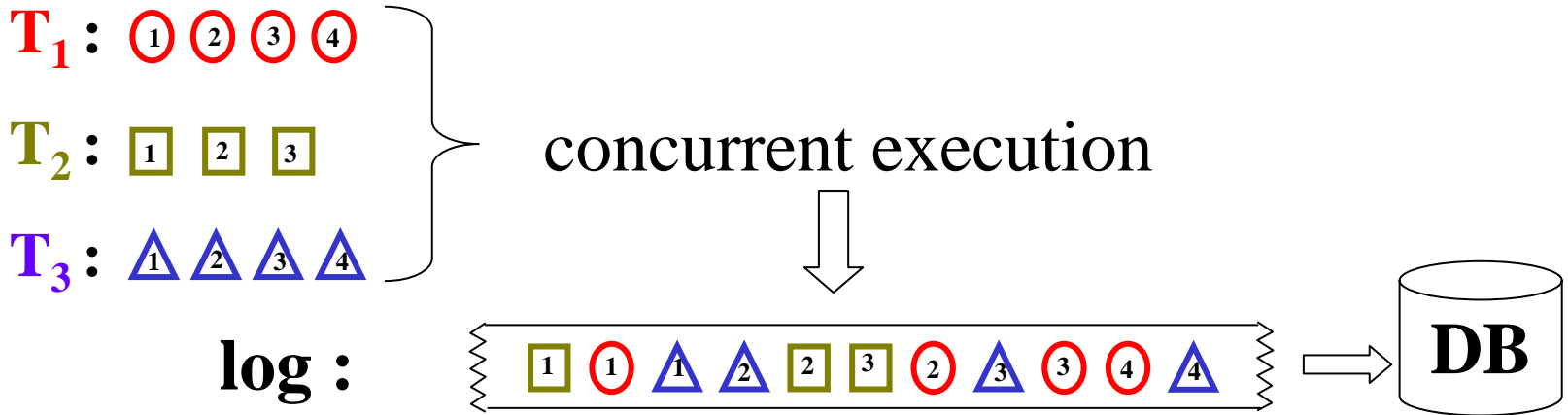
READ(X): read any one copy of X

$R_1(X_3)$

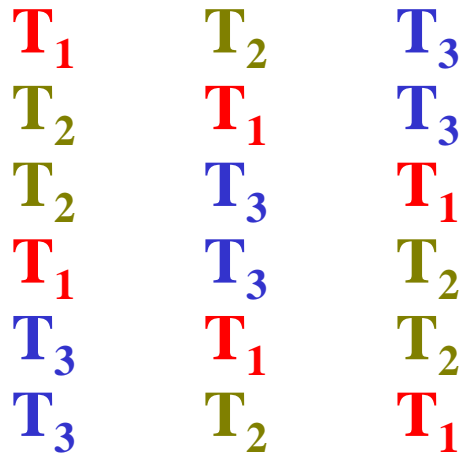
WRITE (Z): write all copies of Z

$W_3(Z_2)$  and  $W_3(Z_3)$

# Serialization



DB is acceptable if it is guaranteed to have resulted from any one of:



# Serialization

Consider two concurrent transactions executed at only one DM

**LOG:**    **R<sub>1</sub>(X)** **R<sub>2</sub>(Y)** **R<sub>1</sub>(Y)** **W<sub>1</sub>(Z)** **W<sub>1</sub>(X)** **W<sub>2</sub>(X)** **R<sub>2</sub>(Z)**

# Serialization

Consider two concurrent transactions executed at only one DM

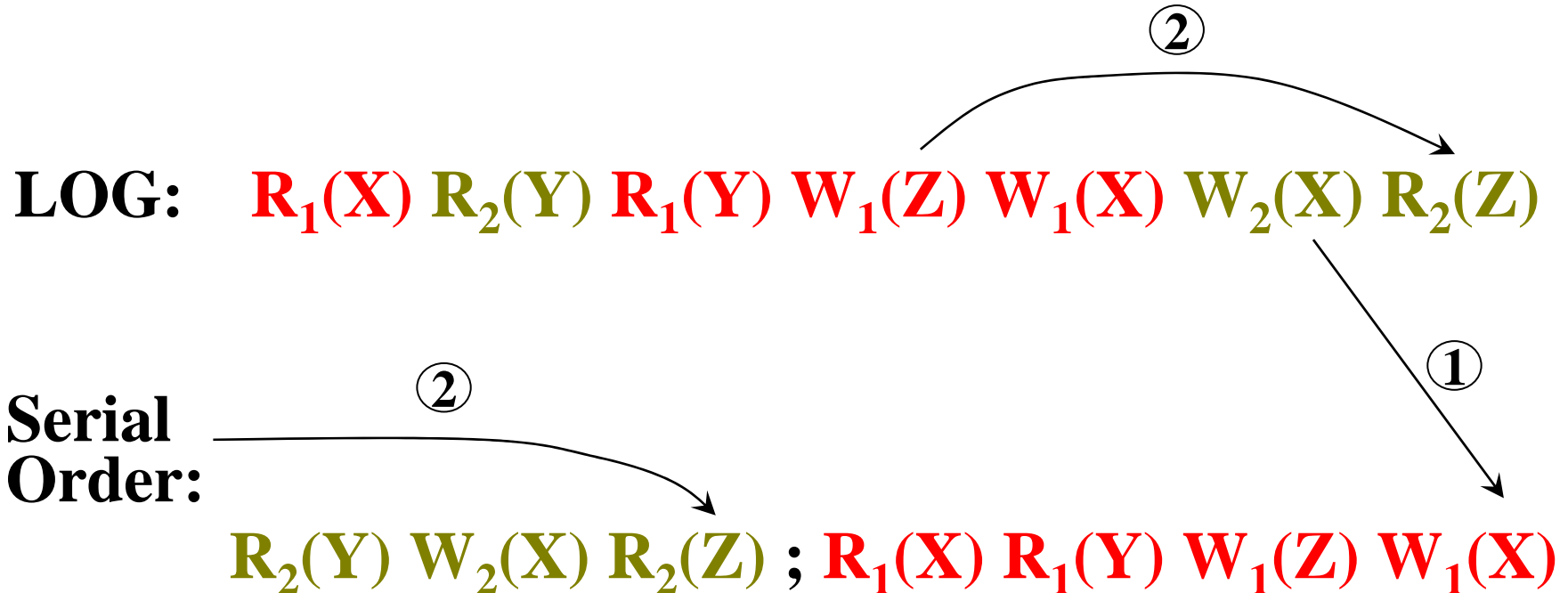
**LOG:**    **R<sub>1</sub>(X)** **R<sub>2</sub>(Y)** **R<sub>1</sub>(Y)** **W<sub>1</sub>(Z)** **W<sub>1</sub>(X)** **W<sub>2</sub>(X)** **R<sub>2</sub>(Z)**

**Serial  
Order:**

**R<sub>2</sub>(Y)** **W<sub>2</sub>(X)** **R<sub>2</sub>(Z)** ; **R<sub>1</sub>(X)** **R<sub>1</sub>(Y)** **W<sub>1</sub>(Z)** **W<sub>1</sub>(X)**

# Serialization

Consider two concurrent transactions executed at only one DM



① last write conflict

② read source conflict

# Serialization

Consider two concurrent transactions executed at only one DM

**LOG:**    **R<sub>1</sub>(X)** **R<sub>2</sub>(Y)** **R<sub>1</sub>(Y)** **W<sub>1</sub>(Z)** **W<sub>1</sub>(X)** **W<sub>2</sub>(X)** **R<sub>2</sub>(Z)**

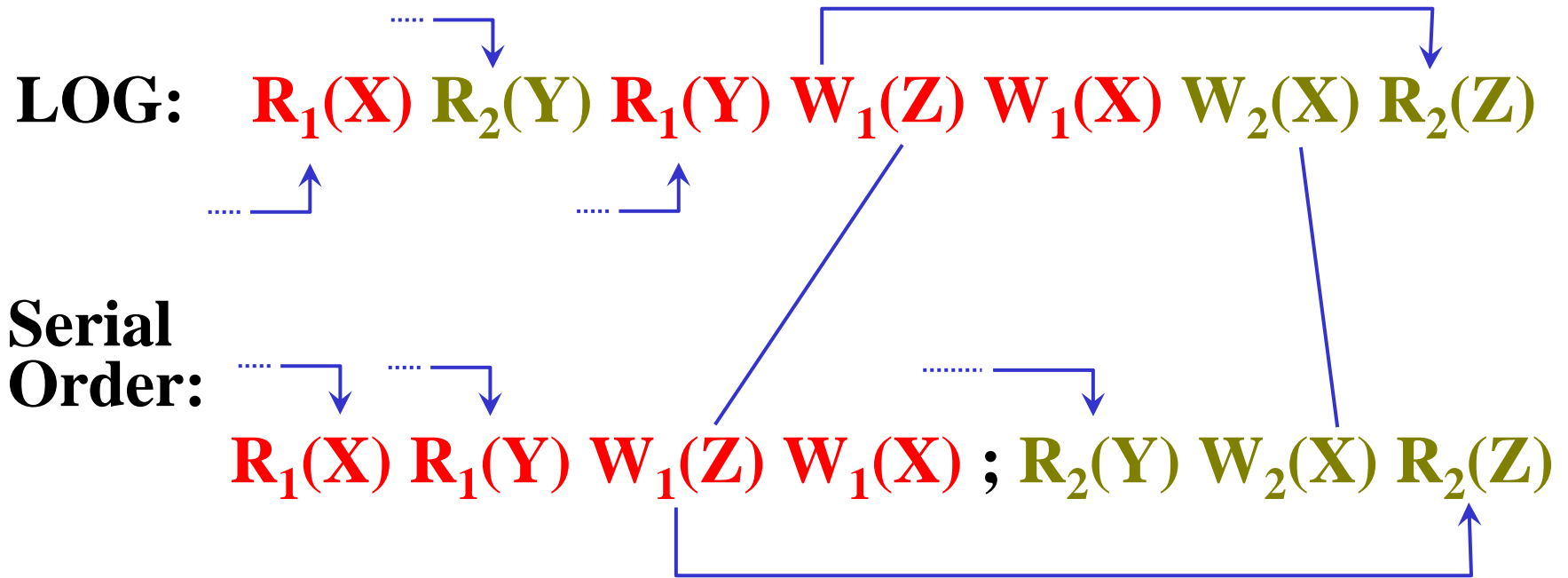
**Serial  
Order:**

**R<sub>1</sub>(X) R<sub>1</sub>(Y) W<sub>1</sub>(Z) W<sub>1</sub>(X) ; R<sub>2</sub>(Y) W<sub>2</sub>(X) R<sub>2</sub>(Z)**



# Serialization

Consider two concurrent transactions executed at only one DM



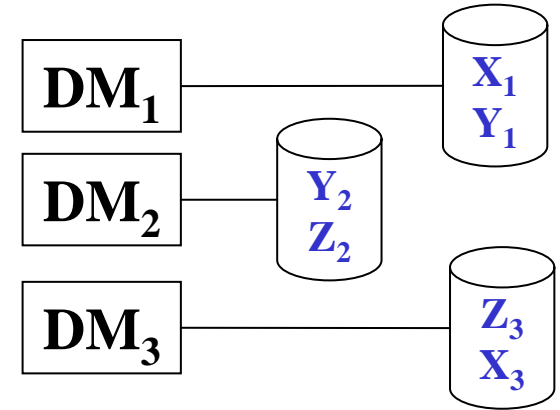
# Distributed Transaction Processing

## Transactions:

$T_1$  : READ(X); WRITE(Y);

$T_2$  : READ(Y); WRITE(Z);

$T_3$  : READ(Z); WRITE(X);



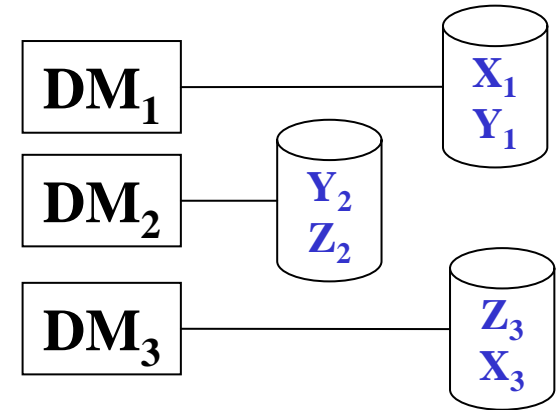
# Distributed Transaction Processing

## Transactions:

$T_1 : \text{READ}(X); \text{WRITE}(Y);$

$T_2 : \text{READ}(Y); \text{WRITE}(Z);$

$T_3 : \text{READ}(Z); \text{WRITE}(X);$



## LOGS:

$L_1 : R_2(Y_1) R_1(X_1) W_1(Y_1) W_3(X_1)$

$L_2 : R_3(Z_2) W_2(Z_2) W_1(Y_2)$

$L_3 : W_3(X_3) W_2(Z_3)$

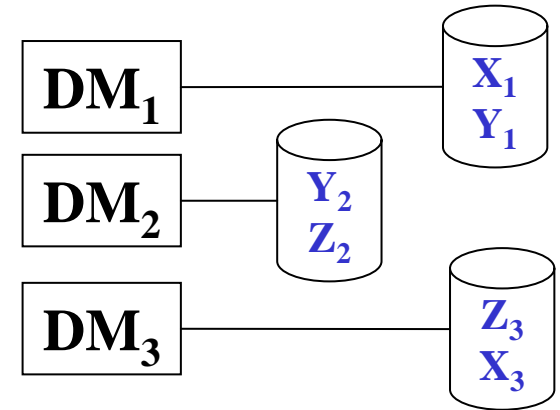
# Distributed Transaction Processing

## Transactions:

$T_1$  : READ(X); WRITE(Y);

$T_2$  : READ(Y); WRITE(Z);

$T_3$  : READ(Z); WRITE(X);



## LOGS:

$L_1$  : R<sub>2</sub>(Y<sub>1</sub>) R<sub>1</sub>(X<sub>1</sub>) W<sub>1</sub>(Y<sub>1</sub>) W<sub>3</sub>(X<sub>1</sub>)

$L_2$  : R<sub>3</sub>(Z<sub>2</sub>) W<sub>2</sub>(Z<sub>2</sub>) W<sub>1</sub>(Y<sub>2</sub>)

$L_3$  : W<sub>3</sub>(X<sub>3</sub>) W<sub>2</sub>(Z<sub>3</sub>)

## Question:

Are these logs equivalent to some serial execution of the transactions?

# Serialization of Distributed Logs

**Conflict:  $P_j(A_X)$  and  $Q_i(B_Y)$  conflict if**

- (1) P and Q are not both READ, and
- (2)  $A = B$
- (3)  $i \neq j$
- (4)  $X = Y$

# Serialization of Distributed Logs

**Conflict:  $P_j(A_X)$  and  $Q_i(B_Y)$  conflict if**

- (1) P and Q are not both READ, and
- (2)  $A = B$
- (3)  $i \neq j$
- (4)  $X = Y$

**LOGS:**

$L_1 : R_2(Y_1) R_1(X_1) W_1(Y_1) W_3(X_1)$

$L_2 : R_3(Z_2) W_2(Z_2) W_1(Y_2)$

$L_3 : W_3(X_3) W_2(Z_3)$

# Serialization of Distributed Logs

**Conflict:  $P_j(A_X)$  and  $Q_i(B_Y)$  conflict if**

- (1) P and Q are not both READ, and
- (2)  $A = B$
- (3)  $i \neq j$
- (4)  $X = Y$

**LOGS:**

$L_1 : R_2(Y_1) R_1(X_1) W_1(Y_1) W_3(X_1)$

$L_2 : R_3(Z_2) W_2(Z_2) W_1(Y_2)$

$L_3 : W_3(X_3) W_2(Z_3)$

②

①

③

①  $\Rightarrow T_1 \rightarrow T_3$

②  $\Rightarrow T_2 \rightarrow T_1$

③  $\Rightarrow T_3 \rightarrow T_2$

**Contradictory**

$\therefore$  No total order

$\therefore$  Not serializable

# Serialization of Distributed Logs

**Theorem: Distributed logs are serializable if there exists a total ordering of the transactions such that for conflicting operations  $P_j$  and  $Q_i$   $P_j \rightarrow Q_i$  in a LOG only if  $T_j \rightarrow T_i$**

**LOGS:**

$L_1 : R_2(Y_1) R_1(X_1) W_1(Y_1) W_3(X_1)$   
 $L_2 : R_3(Z_2) W_2(Z_2) W_1(Y_2)$   
 $L_3 : W_3(X_3) W_2(Z_3)$

①  $\Rightarrow T_1 \rightarrow T_3$

②  $\Rightarrow T_2 \rightarrow T_1$

③  $\Rightarrow T_3 \rightarrow T_2$

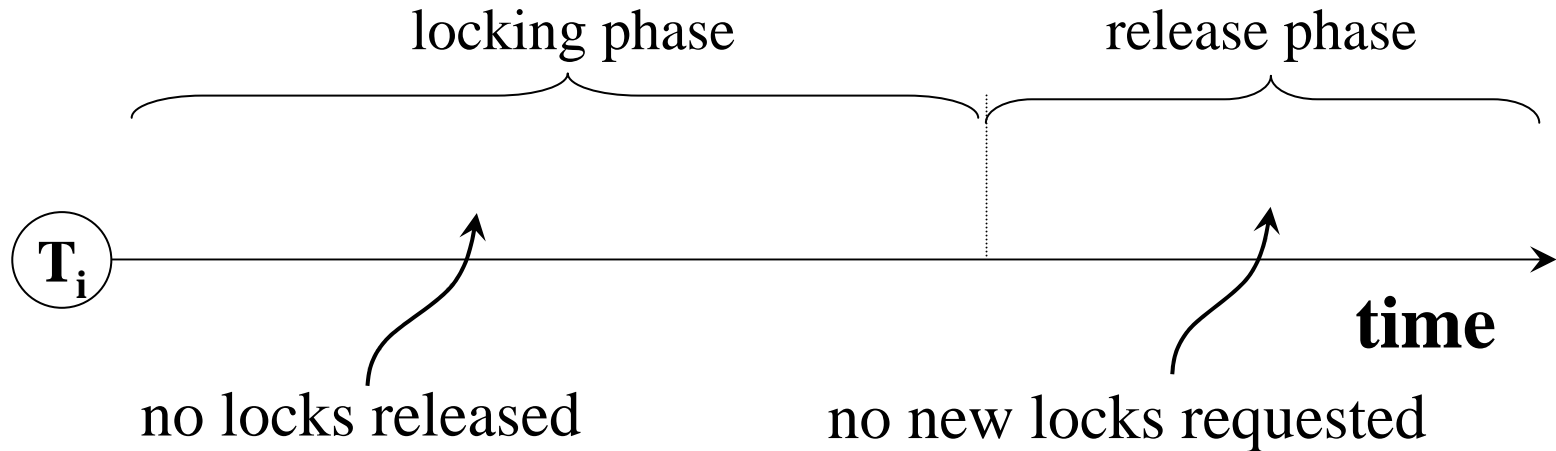
**Contradictory**

$\therefore$  No total order

$\therefore$  Not serializable

# Locking

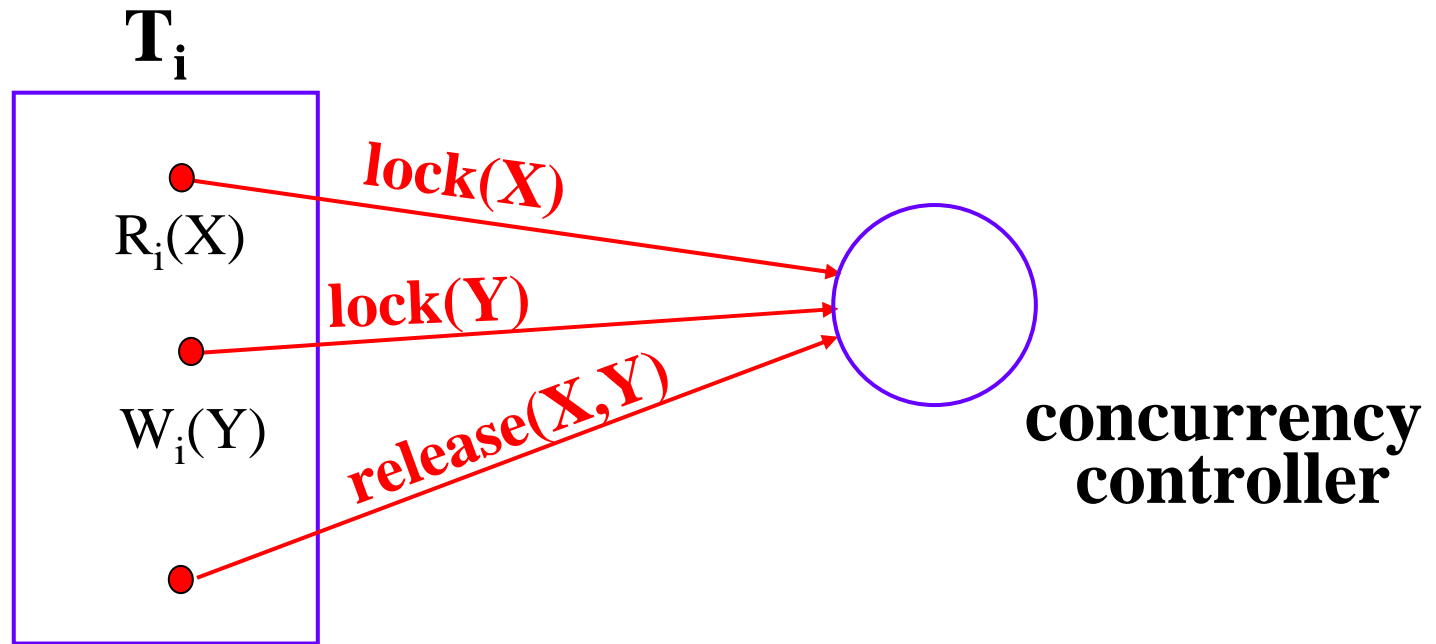
- transactions must use Two Phase Locking (2PL)



- only the following lock requests are granted

	current lock state		
lock request	not locked	READ locked	WRITE locked
READ	OK	OK	DENY
WRITE	OK	DENY	DENY

# Locking



- request lock before accessing a data item
- release all locks at the end of transaction

**This guarantees serializability [ESWAREN]**