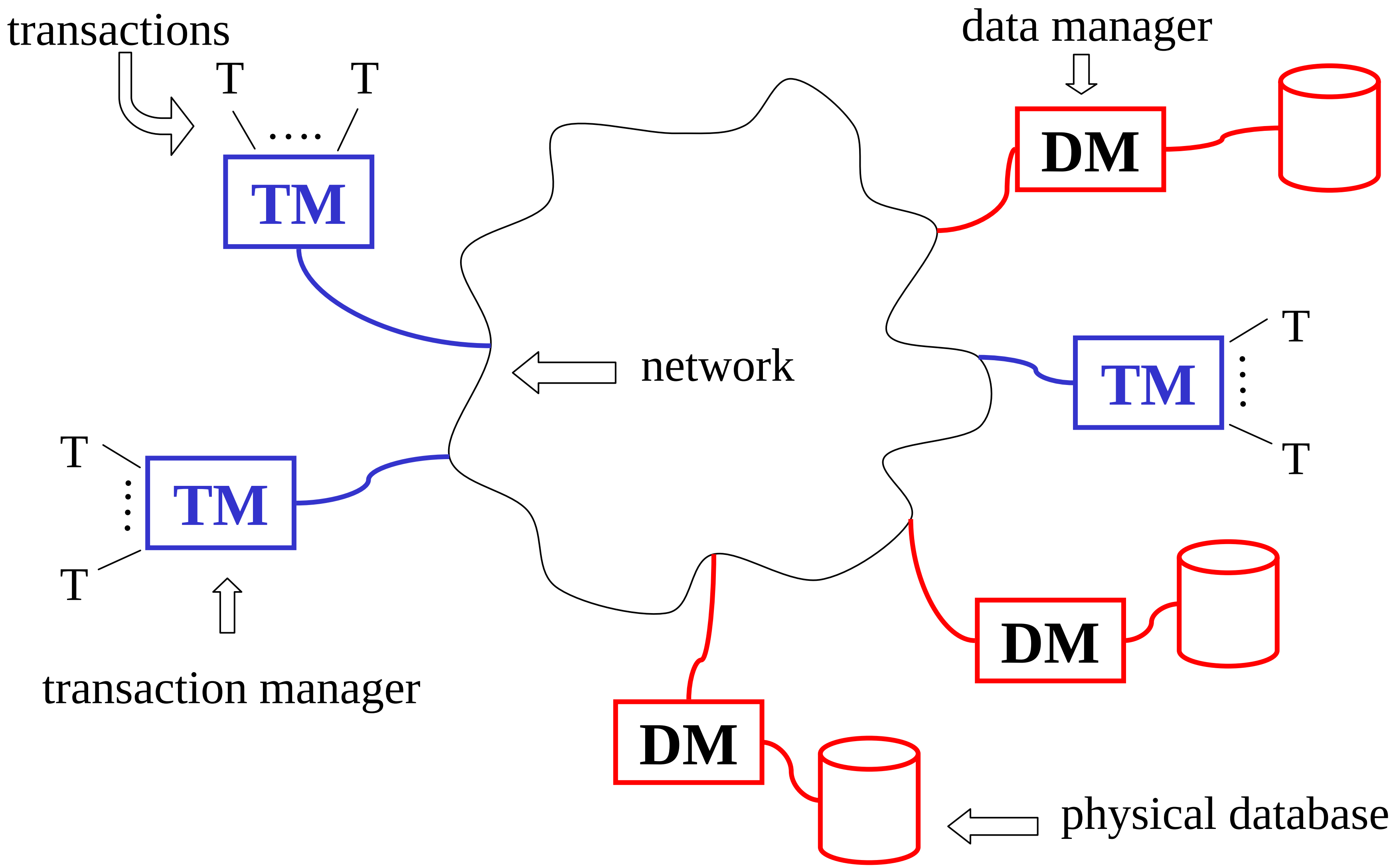
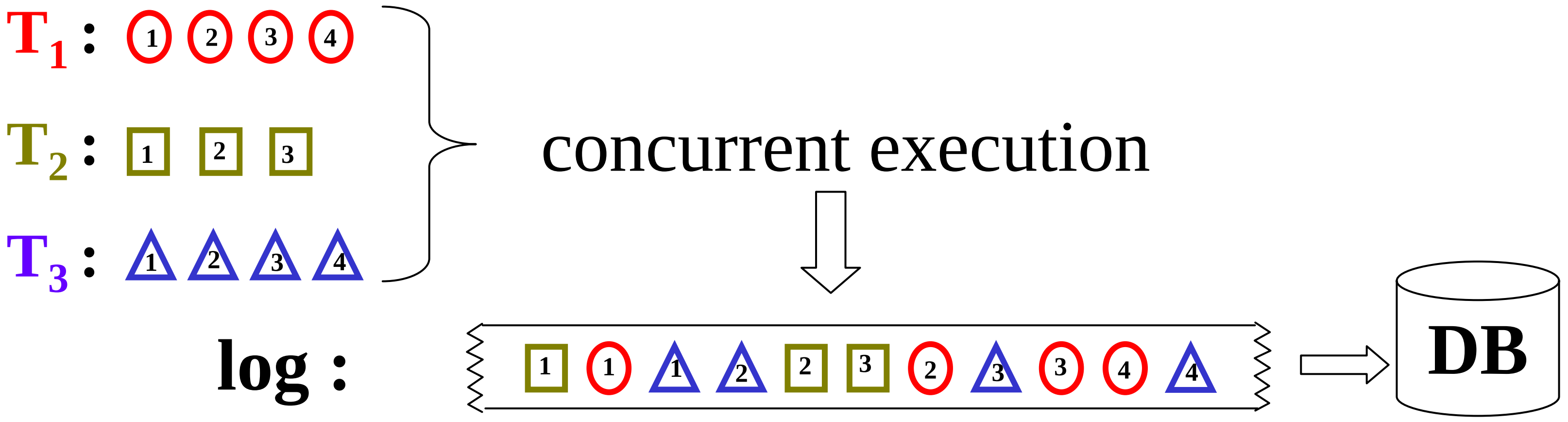
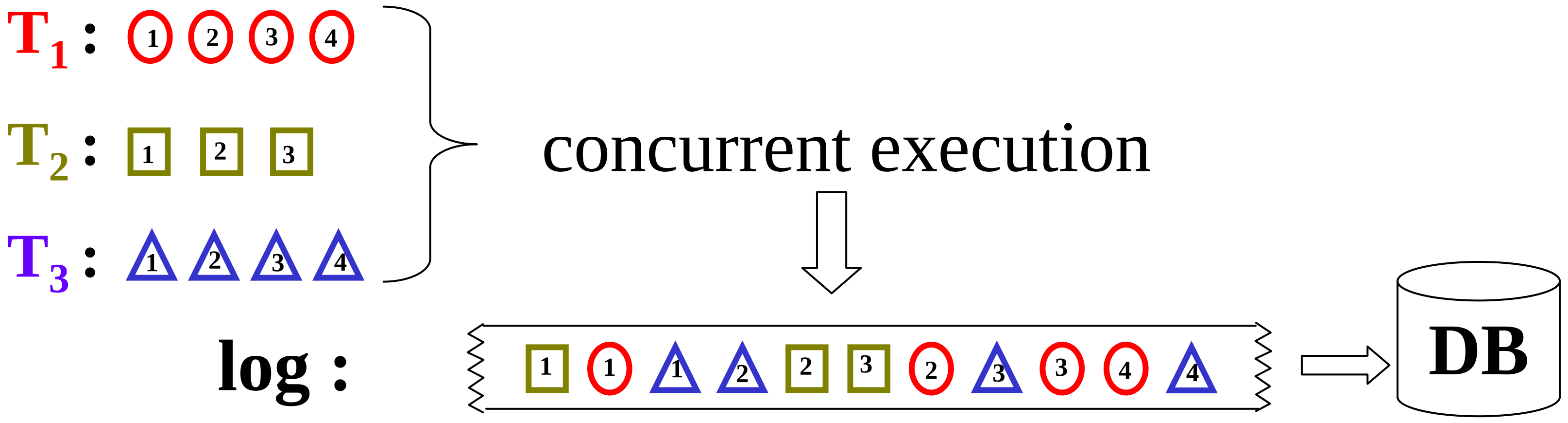
Distributed DBMS Model



Serialization



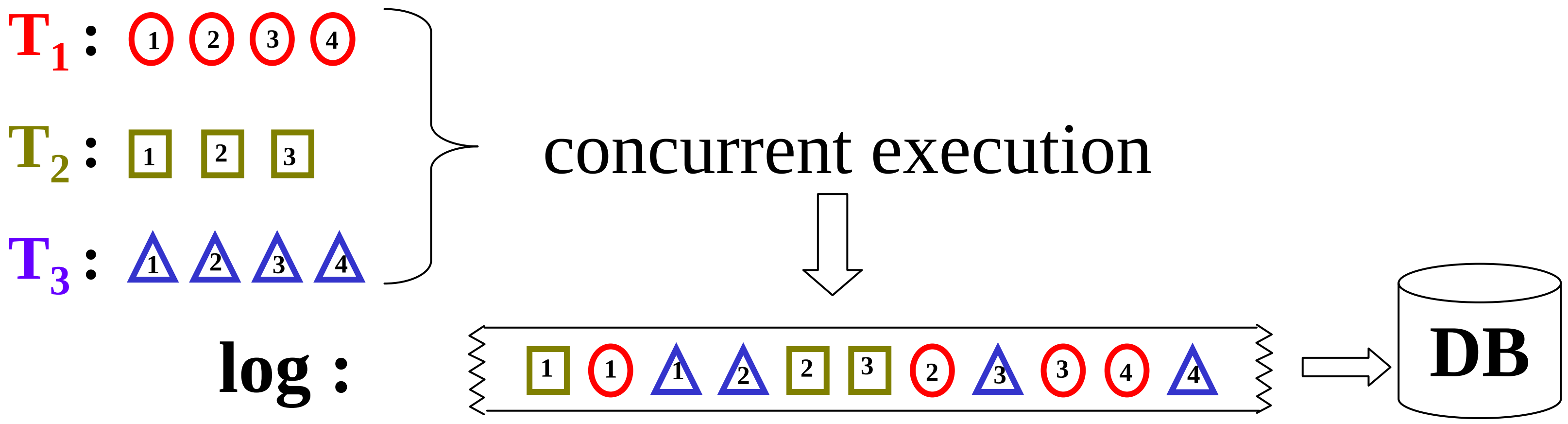
Serialization



OPERATIONS

READ(X):	read <u>any</u> <u>one</u> copy of X
	R_1 (X_3)
WRITE (Z):	write <u>all</u> copies of Z
	$W_3(Z_2)$ <u>and</u> W_3 (Z_3)

Serialization



DB is acceptable if it is <u>guaranteed</u> to have resulted from any <u>one</u> of:

T_1	T_2	T_3
T_2	T_1	T_3
T_2	T_3	T_1
T_1	T_3	T_2
T_3	T_1	T_2
T_3	T_2	T_1

Serialization

Consider two concurrent transactions executed at only one DM

LOG: $R_1(X)\ R_2(Y)\ R_1(Y)\ W_1(Z)\ W_1(X)\ W_2(X)\ R_2(Z)$

Serialization

Consider two concurrent transactions executed at only one DM

LOG: $R_1(X)\ R_2(Y)\ R_1(Y)\ W_1(Z)\ W_1(X)\ W_2(X)\ R_2(Z)$

Serial Order:

$R_2(Y)\ W_2(X)\ R_2(Z)\ ;\ R_1(X)\ R_1(Y)\ W_1(Z)\ W_1(X)$

Serialization

Consider two concurrent transactions executed at only one DM

$$\begin{tikzcd}
\textbf{LOG:} & R_1(X) & R_2(Y) & R_1(Y) & W_1(Z) \arrow[rrr, bend left, "②"] & W_1(X) & W_2(X) \arrow[drr, "①"] & R_2(Z) & \\
\textbf{Serial Order:} \arrow[rrr, bend left, "②"] & R_2(Y) & W_2(X) & R_2(Z) & ; & R_1(X) & R_1(Y) & W_1(Z) & W_1(X)
\end{tikzcd}$$

① **last write conflict**

② **read source conflict**

Serialization

Consider two concurrent transactions executed at only one DM

LOG: $R_1(X)\ R_2(Y)\ R_1(Y)\ W_1(Z)\ W_1(X)\ W_2(X)\ R_2(Z)$

Serial Order:

$R_1(X)\ R_1(Y)\ W_1(Z)\ W_1(X)\ ;\ R_2(Y)\ W_2(X)\ R_2(Z)$

Serialization

Consider two concurrent transactions executed at only one DM

$$\begin{tikzcd}
 & & {} \arrow[d, dotted] & & & & & \\
\text{LOG:} & R_1(X) & R_2(Y) & R_1(Y) & W_1(Z) \arrow[rrr, bend left] \arrow[ddl, dash] & W_1(X) & W_2(X) \arrow[ddr, dash] & R_2(Z) \\
 & {} \arrow[u, dotted] \arrow[d, dotted] & {} \arrow[d, dotted] & {} \arrow[u, dotted] & & & {} \arrow[d, dotted] & & \\
\text{Serial Order:} & R_1(X) & R_1(Y) & W_1(Z) \arrow[rrrrr, bend right] & W_1(X) & ; & R_2(Y) & W_2(X) & R_2(Z)
\end{tikzcd}$$

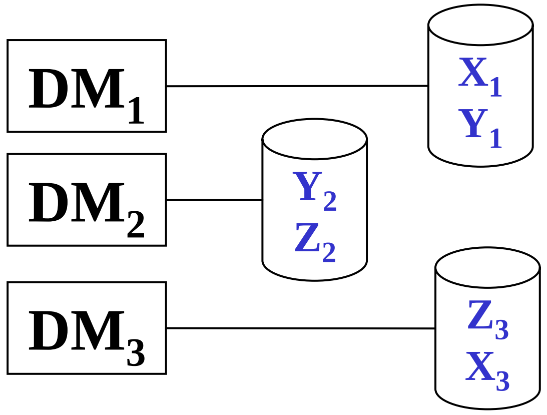
Distributed Transaction Processing

Transactions:

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

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

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



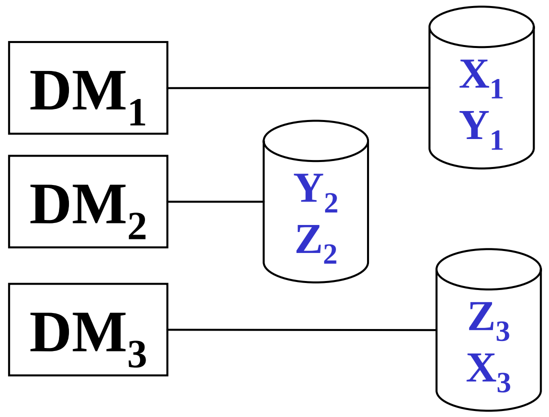
Distributed Transaction Processing

Transactions:

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

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

T_3 : READ(Z); 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 : $W_3(X_3)$ $W_2(Z_3)$

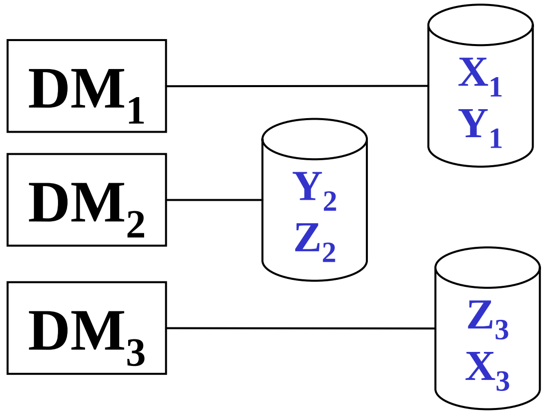
Distributed Transaction Processing

Transactions:

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

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

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



LOGS:

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

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

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

Question:

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

Serialization of Distributed Logs

<u>Conflict:</u> $P_j(A_X)$ 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

<u>Conflict:</u> $P_j(A_X)$ **and** $Q_i(B_Y)$ **conflict if**

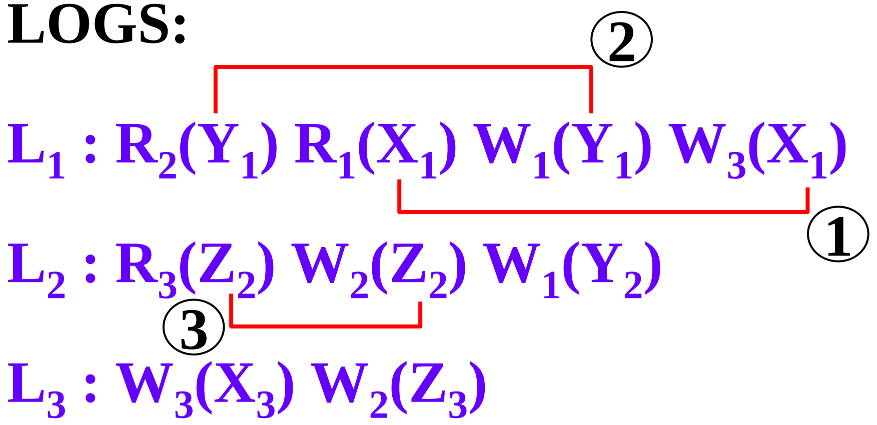
(1) P and Q are not both READ, and

(2) $A = B$

(3) $i \neq j$

(4) $X = Y$

LOGS:



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

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

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

$$\left.\begin{array}{ll} ① & \Rightarrow T_1 \rightarrow T_3 \\ ② & \Rightarrow T_2 \rightarrow T_1 \\ ③ & \Rightarrow T_3 \rightarrow T_2 \end{array}\right\}$$

Contradictory

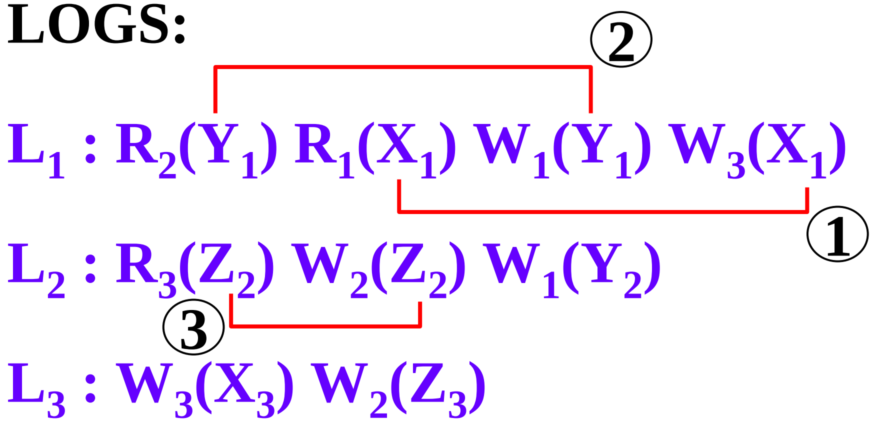
∴ No total order

∴ Not serializable

Serialization of Distributed Logs

<u>Theorem:</u> 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 => $T_1 \rightarrow T_3$

② => $T_2 \rightarrow T_1$

③ => $T_3 \rightarrow T_2$

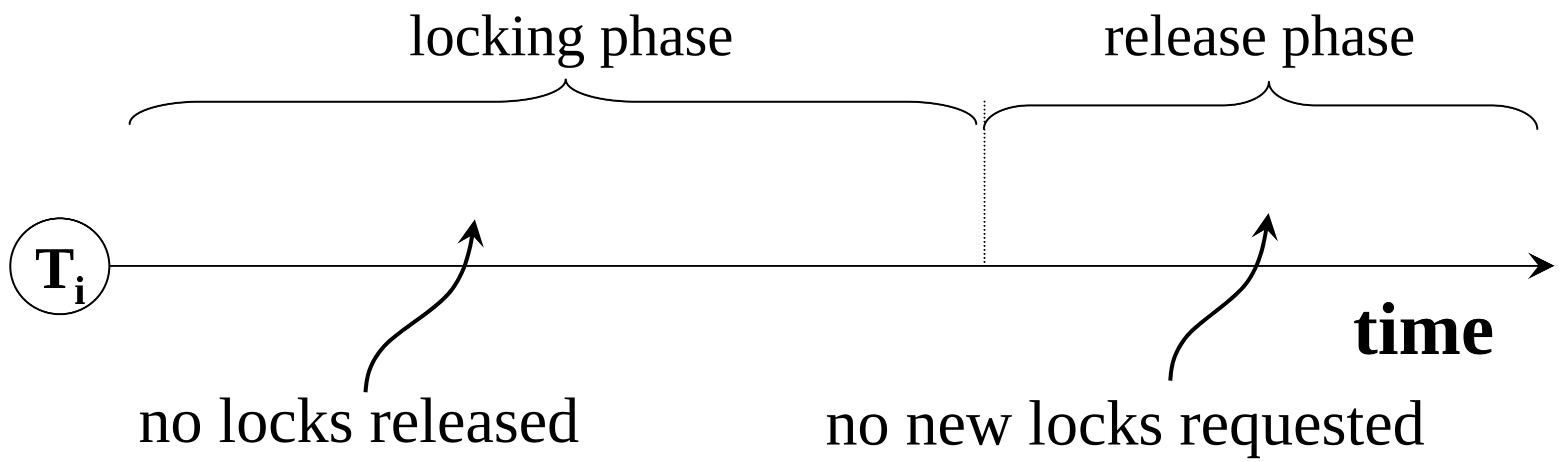
Contradictory

∴ No total order

∴ Not serializable

Locking

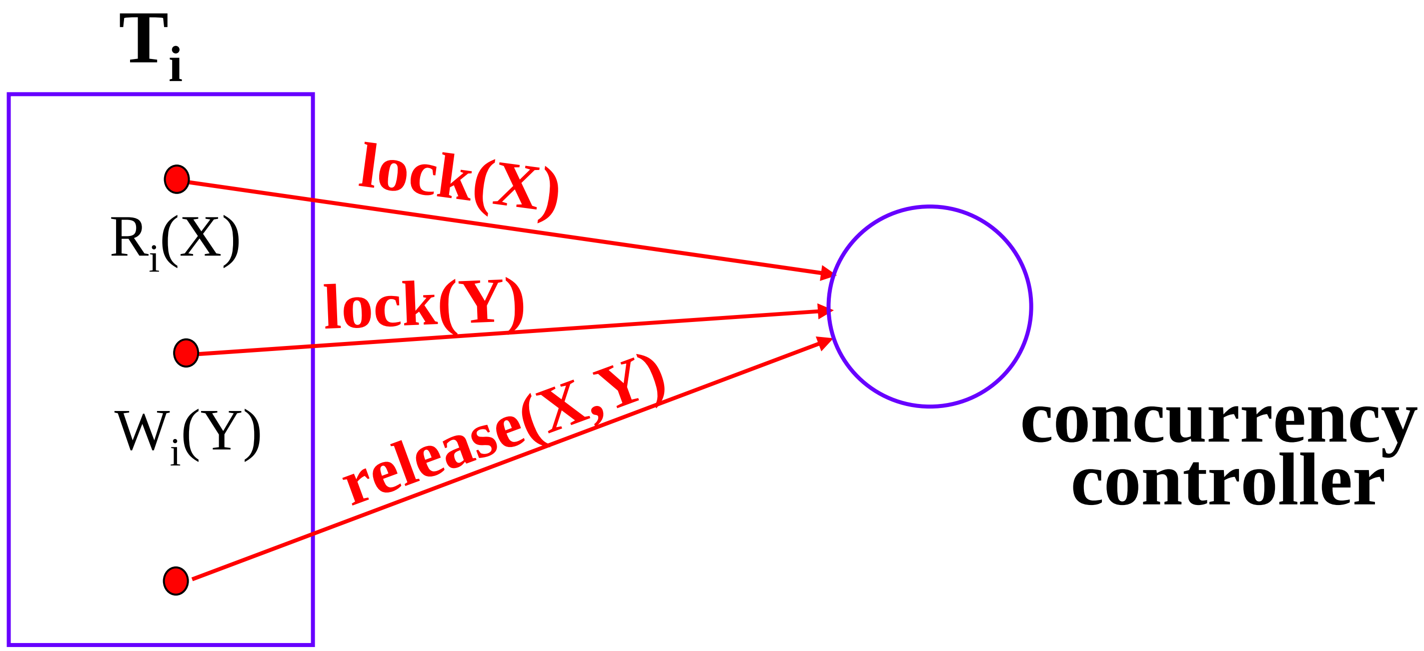
- **transactions must use Two Phase Locking (2PL)**



- **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]