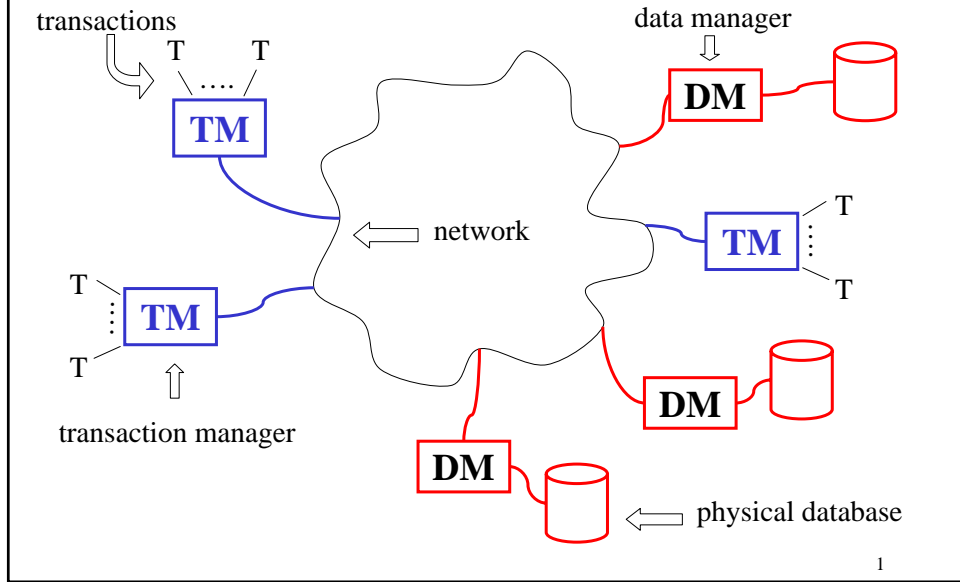
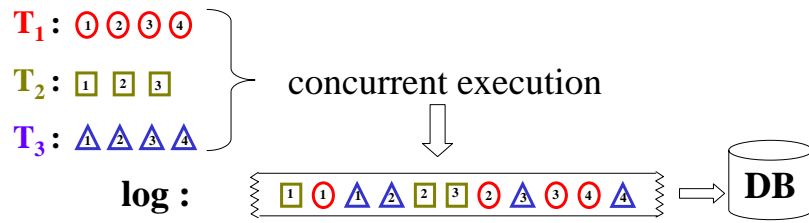


Distributed DBMS Model



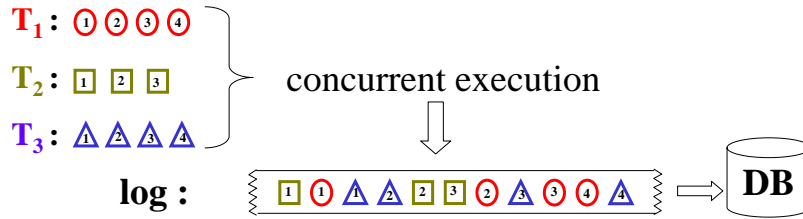
1

Serialization



2

Serialization

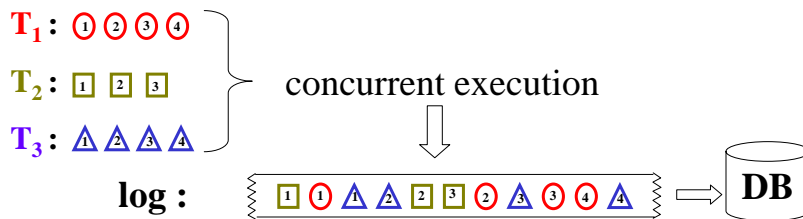


OPERATIONS

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

3

Serialization



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

4

Serialization

Consider two concurrent transactions executed at only one DM

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

5

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:

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

6


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



- ① last write conflict
- ② read source conflict

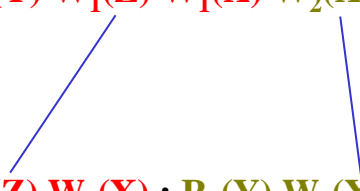
7

Serialization

Consider two concurrent transactions executed at only one DM

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

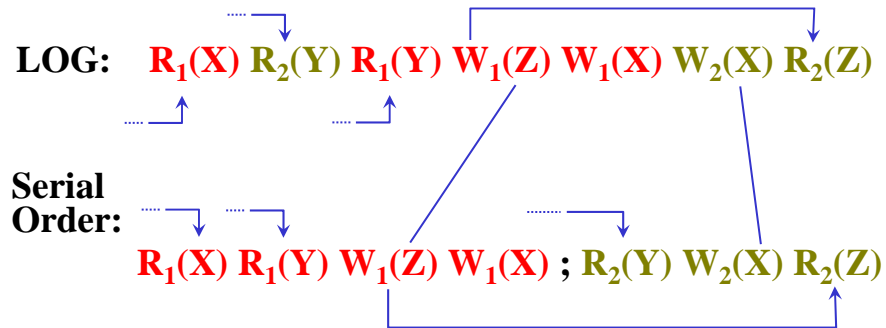
Serial Order: $R_1(X) R_1(Y) W_1(Z) W_1(X) ; R_2(Y) W_2(X) R_2(Z)$



8

Serialization

Consider two concurrent transactions executed at only one DM



9

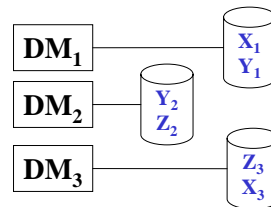
Distributed Transaction Processing

Transactions:

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

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

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



10

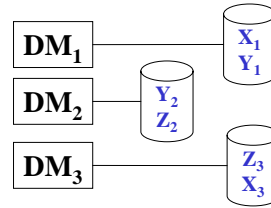
Distributed Transaction Processing

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

11

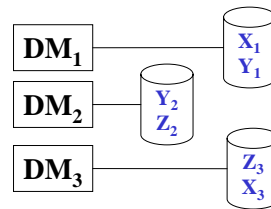
Distributed Transaction Processing

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?

12

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) $X = Y$

13

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:

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

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

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

14

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:

$L_1 : R_2(Y_1) \ R_1(X_1) \ W_1(Y_1) \ W_3(X_1)$ ②
 $L_2 : R_3(Z_2) \ W_2(Z_2) \ W_1(Y_2)$ ①
 $L_3 : W_3(X_3) \ W_2(Z_3)$ 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

15

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:

$L_1 : R_2(Y_1) \ R_1(X_1) \ W_1(Y_1) \ W_3(X_1)$ ②
 $L_2 : R_3(Z_2) \ W_2(Z_2) \ W_1(Y_2)$ ①
 $L_3 : W_3(X_3) \ W_2(Z_3)$ ③

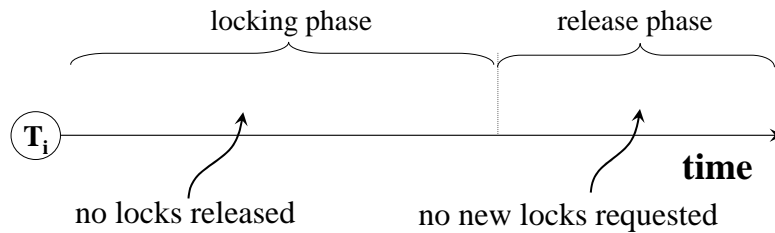
- ① $\Rightarrow T_1 \rightarrow T_3$
- ② $\Rightarrow T_2 \rightarrow T_1$
- ③ $\Rightarrow T_3 \rightarrow T_2$

Contradictory
 \therefore No total order
 \therefore Not serializable

16

Locking

- transactions must use Two Phase Locking (2PL)

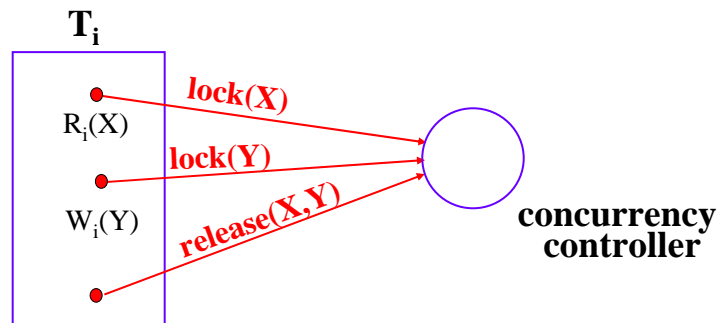


- only the following lock requests are granted

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

17

Locking



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

This guarantees serializability [ESWAREN]

18