Distributed DBMS Model

Serialization

T_1: OOOO
T_2: D D D D
T_3: OOOO

concurrent execution

log:

DB

OPERATIONS

READ(X): read any one copy of X
R_1(X_1)
WRITE(Z): write all copies of Z
W_1(Z_2) and W_1(Z_3)
W_2(Z_1)

Consider two concurrent transactions executed at only one DM

LOG:

R_1(X) R_2(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

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

Serial Order:

\[ R_1(Y) W_2(X) R_1(Z) ; R_1(X) R_1(Y) W_1(Z) W_1(X) \]

1. last write conflict
2. read source conflict

---

**Distributed Transaction Processing**

Transactions:

- \( T_1 : \text{READ}(X); \text{WRITE}(X); \)
- \( T_2 : \text{READ}(Y); \text{WRITE}(Z); \)
- \( T_3 : \text{READ}(Z); \text{WRITE}(X); \)

LOGS:

- \( L_1 : R_1(Y_1) R_2(X_1) W_1(Y_1) W_2(X_1) \)
- \( L_2 : R_2(Z_2) W_2(Z_2) W_1(Y_2) \)
- \( L_3 : W_1(X_3) W_2(Z_2) \)

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. \( X = Y \)

**LOGS:**

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

Contradictory
∴ No total order
∴ Not serializable

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:**

\[
L_1 : R_2(Y_1) \ R_2(X_1) \ W_1(Y_1) \ W_3(X_1) \\
L_2 : R_3(Z_2) \ W_2(Z_2) \ W_1(Y_2) \\
L_3 : W_3(X_3) \ W_2(Z_3)
\]

Contradictory
∴ No total order
∴ Not serializable

Locking

- transactions must use Two Phase Locking (2PL)
- request lock before accessing a data item
- release all locks at the end of transaction

This guarantees serializability [ESWAREN]