Outline

- Introduction
- Background
- Distributed DBMS Architecture
- Distributed Database Design
- Distributed Query Processing
- Distributed Transaction Management
  - Transaction Concepts
- Building Distributed Database Systems (RAID)
- Mobile Database Systems
- Privacy, Trust, and Authentication
- Peer to Peer Systems

Useful References

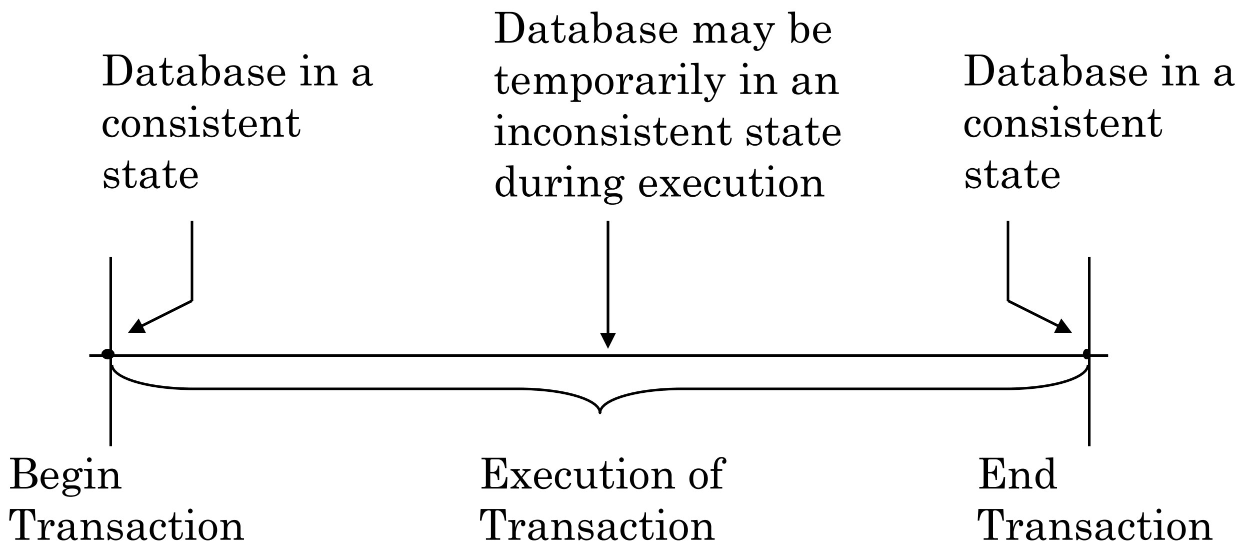
- C. Papadimitriou, *The serializability of concurrent database updates*, Journal of the ACM, 26(4), 1979.
- S. B. Davidson, *Optimism and consistency in partitioned distributed database systems*, ACM Transactions on Database Systems 9(3): 456-481, 1984.
- B. Bhargava and C. Hua. *A Causal Model for Analyzing Distributed Concurrency Control Algorithms*, IEEE Transactions on Software Engineering, SE-9, 470-486, 1983.
- Textbook *Principles of Distributed Database Systems*,

  Chapter 10.1, 11.1

Transaction

A transaction is a collection of actions that make consistent transformations of system states while preserving system consistency.

- concurrency transparency
- failure transparency



Formal Definitions and Models

Definition 1: A history is a quadruple $h = (n, \Pi, M, S)$ where

n is a positive integer,

Π is a permutation of the set

$\Sigma_n = \{R_1, W_1, R_2, W_2, \ldots, R_n, W_n\}$

equivalently a one-to-one function

$\Pi: \Sigma_n \rightarrow \{1, 2, \ldots, 2n\}$

that $\Pi(R_i) < \Pi(W_i)$ for $i = 1,2, \ldots, n$

M is a finite set of variables representing physical data items,

S is a function mapping Σ_n to 2^M

Set of all histories is denoted by M.

Definition 2: A transaction Ti is a pair (Ri, Wi). A transaction is a single execution of a program. This program may be a simple query statement expressed in a query language.

Definition 3: Read set of Ti is denoted by S (Ri) and Write set of Ti is denoted by S(Wi).

Formal Definitions and Models

Definition 4: A history $h = (n, \Pi, M, S)$ is serial if $\Pi(Wi) = \Pi(Ri) + 1$ for all $i = 1,2, \ldots, n$.

In other words, a history is serial if Ri immediately precedes Wi for $i = 1,2, \ldots, n$.

Definition 5: A history is serializable if there is some serial history hs such that the effect of the execution of h is equivalent to hs. Note serializability requires only that there exists some serial order equivalent to the actual interleaved execution history. There may in fact be several such equivalent serial orderings.

Definition 6: A history h is strongly serializable if in hs the following conditions hold true:

a) $\Pi(Wi) = \Pi(Ri) + 1$

b) $\Pi(R\ (i + 1)) = \Pi(Wi) + 1$

If $t(i + 1)$ is the next transaction that arrived and obtained the next time-stamp after Ti. In strongly serializable history, the following constraint must hold "If a transaction Ti is issued before a transaction Tj, then the total effect on the database should be equivalent to the effect that Ti was executed before Tj.

Note if Ti and Tj are independent, e.g., $\{S(Ri) \cup S(Wi)\} \cap \{S(Rj) \cup S(Wj)\} = \emptyset$ then the effect of execution TiTj or TjTi will be the same.

Formal Definitions and Models

history $h = (n, \pi, V_1 S)$

$$\bar{h} = (n+2, \bar{\pi}, V_1 \bar{S})$$

$$h = T_{n+1} \cdot h \cdot T_{n+2}$$

Live transaction (set can be found in $O(n \cdot |V|)$.

Two histories are equivalent ($\equiv$) if they have the same set of live transactions.

Equivalence can be determined $O(n \cdot |V|)$.

Theorem: Testing whether a history h is serializable is NP-complete even if h has no dead transactions.

- Polygraph: Pair of arcs between nodes
- Satisfiability: Problem of Boolean formulas in conjuctive normal forms with two-/three literals

  (SAT)

  (Non-circular)

Concatenation of histories:

$$h_1 = (n_1, \pi_1, V_1, S_1)$$

$$h_2 = (n_2, \pi_2, V_2, S_2)$$

$$h_1 \circ \quad - \quad n_2, \tau, V_1, P)$$

$$\tau(w_i) = \pi_1(w_i) \qquad i \leq n$$

$$\tau(w_i) = \pi_2(w_{i-n}) + 2n \quad \text{for} \quad i > n$$

same true for Ri

$$h_1 = R_1 W_1$$

$$h_2 = R_1 W_1$$

$$h_1 \circ \quad - \quad {}_1 R_2 W_2$$

Two-phase locking:

$h = (n, \pi, V, S)$ is 2PL

If $\exists$ distinct non-integer real numbers

$l_1, ..., l_n$ such that

(a) $\pi(R_i) < l_i < \pi(W_i)$ for i = 1, ..., n

(b) If $S(R_i) \cap S(W_j) \neq \emptyset$, $i \neq j$, and $\pi(R_i) < \pi(W_j)$, then $l_i < l_j$

(c) If $S(W_i) \cap S(W_j) \neq \emptyset$ and $\pi(W_i) < \pi(W_j)$, then $\pi(W_i) < l_j$

The Class DSR

$h_1 = (n, \pi, V, S)$ and $h_2 = (n, \pi', V, S)$ are histories.

$h_1 \sim h_2$ whenever $\pi(\sigma) = \pi'(\sigma)$ for all Σ_n except for two elements $\sigma_1, \sigma_2 \in \Sigma_n$ with

$\pi(\sigma_1) = \pi'(\sigma_2) = j$,

$\pi(\sigma_2) = \pi'(\sigma_1) = j + 1$

for some $1 \leq j \leq n - 1$, and

(a) $\sigma_1 = R_i, \sigma_2 = R_j$ for some $i, j \leq n$, or

(b) $\sigma_1 = R_i, \sigma_2 = W_j, i \neq j, i, j \leq n$, and $S(R_i) \cap S(W_j) = \emptyset$, or

(c) $\sigma_1 = W_i, \sigma_2 = W_j, i, j \leq n$, and $S(W_i) \cap S(W_j) = \emptyset$.

Let $\sim^*$ be reflexive-transitive closure of $\sim$.

The history h is D-serializable (DSR) if there is a serial history h_s such that $h \sim^* h_s$.

If a history is DSR, it is certanly SR.

Transaction Example – A Simple SQL Query

```
Transaction BUDGET_UPDATE
begin
    EXEC SQL UPDATE PROJ
             SET    BUDGET = BUDGET*1.1
             WHERE  PNAME = “CAD/CAM”
end.
```

Example Database

Consider an airline reservation example with the relations:

FLIGHT(<u>FNO, DATE</u>, SRC, DEST, STSOLD, CAP)
CUST(<u>CNAME</u>, ADDR, BAL)
FC(<u>FNO, DATE, CNAME</u>,SPECIAL)

Example Transaction – SQL Version

```
Begin_transaction Reservation
begin
    input(flight_no, date, customer_name);
    EXEC SQL  UPDATE  FLIGHT
              SET     STSOLD = STSOLD + 1
              WHERE   FNO = flight_no AND DATE = date;
    EXEC SQL  INSERT
              INTO    FC(FNO, DATE, CNAME, SPECIAL);
              VALUES  (flight_no, date, customer_name, null);
    output("reservation completed")
end . {Reservation}
```


Termination of Transactions

```
Begin_transaction Reservation
begin
    input(flight_no, date, customer_name);
    EXEC SQL SELECT   STSOLD,CAP
             INTO     temp1,temp2
             FROM     FLIGHT
             WHERE    FNO = flight_no AND DATE = date;
    if temp1 = temp2 then
        output("no free seats");
        Abort
    else
        EXEC SQL   UPDATE FLIGHT
                   SET    STSOLD = STSOLD + 1
                   WHERE  FNO = flight_no AND DATE = date;
        EXEC SQL   INSERT
                   INTO   FC(FNO, DATE, CNAME, SPECIAL);
                   VALUES (flight_no, date, customer_name, null);
      Commit
      output("reservation completed")
   endif
end . {Reservation}
```

Example Transaction – Reads & Writes

```
Begin_transaction Reservation
begin
        input(flight_no, date, customer_name);
        temp ← Read(flight_no(date).stsold);
        if temp = flight(date).cap then
        begin
            output("no free seats");
            Abort
        end
        else begin
            Write(flight(date).stsold, temp + 1);
            Write(flight(date).cname, customer_name);
            Write(flight(date).special, null);
            Commit;
            output("reservation completed")
        end
end. {Reservation}
```

Characterization

- Ti
  - Transaction i
- Read set (RS)
  - The set of data items that are read by a transaction
- Write set (WS)
  - The set of data items whose values are changed by this transaction
- Base set (BS)
  - RS $\cup$ WS

Formalization Based on Textbook

Let

- $O_{ij}(x)$ be some operation O_j of transaction T_i operating on entity x, where $O_j \in$ {read,write} and O_j is atomic
- $OS_i = \cup_j O_{ij}$
- $N_i \in$ {abort,commit}

Transaction T_i is a partial order $T_i = \{\Sigma_i, <_i\}$ where

- $\Sigma_i = OS_i \cup \{N_i\}$
- For any two operations O_{ij} , $O_{ik} \in OS_i$, if $O_{ij} = R(x)$ and $O_{ik} = W(x)$ for any data item x, then either $O_{ij} <_i O_{ik}$ or $O_{ik} <_i O_{ij}$
- $\forall O_{ij} \in OS_i, O_{ij} <_i N_i$

Example

Consider a transaction T:

Read(x)

Read(y)

$x \leftarrow x + y$

Write(x)

Commit

Then

$$\Sigma = \{R(x), R(y), W(x), C\}$$

$$< \, = \{(R(x), W(x)), (R(y), W(x)), (W(x), C), (R(x), C), (R(y), C)\}$$

DAG Representation

Assume

$$\prec = \{(R(x), W(x)), (R(y), W(x)), (R(x), C), (R(y), C), (W(x), C)\}$$