Transaction Schedules

Friday, May 31, 2019 1:32 PM

- Modeling a transaction
  - Database = a collection of elements.
    - element { record (logical) / disk block (physical) }
  - Transaction = sequence of read/writes of elements

- Schedule: a sequence of interleared actions from all transaction.
  - Serial schedule: in which transactions are executed one after another
    - If use serial, nothing can go wrong.
  - Why not serial transaction: when read from disk, take long time.
    - A transaction does not wait for user.

- Serializable: A schedule is serializable if its equivalent to a serial schedule,

T1	T2
	READ(A,s)
	s := s*2
	WRITE(A,s)
	READ(B,s)
	s := s*2

WRITE(B,s)

READ(A, t)
t := t+100
WRITE(A, t)
READ(B, t)
t := t+100
WRITE(B,t)

T1	T2
READ(A, t)	
t := t+100	
WRITE(A, t)	
	READ(A,s)
	s := s*2
	WRITE(A,s)
READ(B, t)	
t := t+100	
WRITE(B,t)	
	READ(B,s)
	s := s*2
	WRITE(B,s)

This is a serializable schedule.
This is NOT a serial schedule

T1	T2
READ(A, t)	
t := t+100	
WRITE(A, t)	
	READ(A,s)
	s := s*2
	WRITE(A,s)
	READ(B,s)
	s := s*2
	WRITE(B,s)
READ(B, t)	
t := t+100	
WRITE(B,t)	

← this is not serializable.
Notice that they should first times 2, then +100

- Check if serializable:
  - focus on conflicting operation
  - **Conflict Serializability** (Swaps will change the program behavior)
    - Two actions by the same transaction T_i $r_i(X); w_i(Y)$
    - Two writes by T_i and T_j to same element $w_i(X); w_j(X)$.
    - Read/Write by T_i, T_j to same element $\begin{cases} w_i(X); r_j(X) \\ r_i(X); w_j(X) \end{cases}$
  - Every conflict serializable schedule is serializable.
    - The previous example is not conflict serializable.

- Testing for conflict-serializability
  - **Precedence graph**
    - A node for each transaction T_i.
    - A edge to T_i to T_j whenever an action T_i conflicts with, and comes before an action in T_j.
    - The schedule is conflict-serializable $\Leftrightarrow$ the precedence graph is **acyclic**.

e.g.1. $r_2(A); r_1(B); w_2(A); r_3(A); w_1(B); w_3(A); r_2(B); w_2(B)$.

Conflict-serializable

e.g.2. $r_2(A)$; $r_1(B)$; $w_2(A)$; $r_2(B)$; $r_3(A)$; $w_1(B)$; $w_3(A)$; $w_2(B)$.

Not conflict-serializable