Concurrency Control, Ch 18

Objectives:
- Serializble Schedules
- Serializability Theorem

Modified from Hector Garcia-Molina slides

2 Lecture Sequence

1. All concept – *nothing about implementation*
2. Then, how to integrate locks to guarantee the serializability property

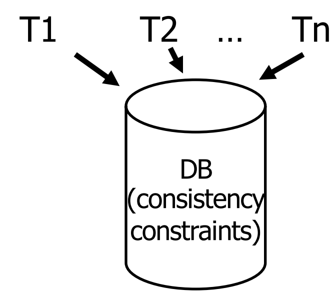
Correctness

This text' s authors include "transaction people". Introduce a broader notion of correctness

> Correctness principle wrt a set of transactions reaching a serializable state

- Syntactic – conflict serializability
- Semantic – more general

Chapter 18__Concurrency Control



Notation

- Transactions, Ti
- Transactions
  - Read(X) // x identifies an object in memory
  - Write(X) // disk read/writes - outside our interests
- Arithmetics I.e. X = 2x
  - used only for pedagogical convenience
  - algorithms we are concerned with, oblivious to internal behavior or a transaction, (why?)

Example:
Constraint: A=B – correctness invariant

T1:		T2:	
	Read(A)		Read(A)
	$A \leftarrow A+100$		$A \leftarrow A\times 2$
	Write(A)		Write(A)
	Read(B)		Read(B)
	$B \leftarrow B+100$		$B \leftarrow B\times 2$
	Write(B)		Write(B)

Schedule A

T1	T2	A	B
		25	25
Read(A); $A \leftarrow A+100$			
Write(A);		125	
Read(B); $B \leftarrow B+100$;			
Write(B);			125
	Read(A);$A \leftarrow A\times 2$;		
	Write(A);	250	
	Read(B);$B \leftarrow B\times 2$;		
	Write(B);		250
		250	250

Schedule B

T1	T2	A	B
		25	25
	Read(A);$A \leftarrow A\times 2$;		
	Write(A);	50	
	Read(B);$B \leftarrow B\times 2$;		
	Write(B);		50
Read(A); $A \leftarrow A+100$			
Write(A);		150	
Read(B); $B \leftarrow B+100$;			
Write(B);			150
		150	150

Schedule A & B: Serial Schedules

- Notice,
  - final results are different
  - which is correct? ans. both

Schedule C - now we have concurrency

T1	T2	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
	Read(A);A ← A×2;		
	Write(A);	250	
Read(B); B ← B+100;			
Write(B);			125
	Read(B);B ← B×2;		
	Write(B);		250
		250	250

and a correct answer

Schedule D

T1	T2	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
	Read(A);A ← A×2;		
	Write(A);	250	
	Read(B);B ← B×2;		
	Write(B);		50
Read(B); B ← B+100;			
Write(B);			150
		250	150

oops

Schedule E

Same as Schedule D
but with new T2'
- text authors research interest

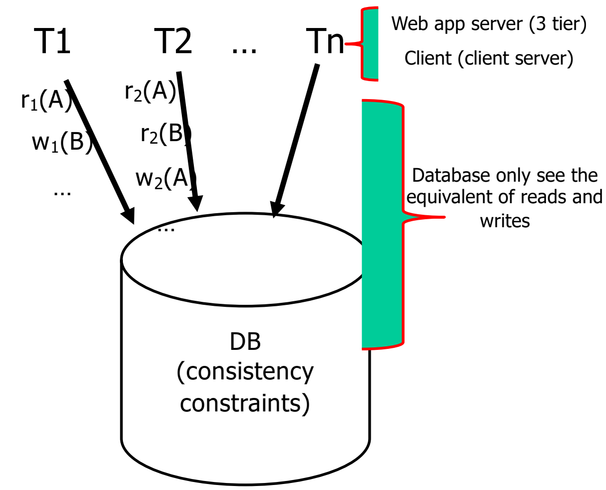
T1	T2'	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
	Read(A);A ← A×1;		
	Write(A);	125	
	Read(B);B ← B×1;		
	Write(B);		25
Read(B); B ← B+100;			
Write(B);			125
		125	125

- Want schedules that are "good", regardless of
  - initial state and
  - transaction semantics
- Only look at order of read and writes

Notation: For transaction Ti

$r_i(A)$,

$w_i(B)$,



Schedule:

A schedule is an ordered sequence of operations taken by a set of transactions

Example:

$Sc=r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$

Serial schedule:

Serial schedule: no interleaving of actions or transactions

$$Sa=\underbrace{r_1(A)w_1(A)\ r_1(B)w_1(B)}_{T_1}\underbrace{r_2(A)w_2(A)r_2(B)w_2(B)}_{T_2}$$

The Swapping Game

Example:

$Sc=r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$

$Sa=\underbrace{r_1(A)w_1(A)\ r_1(B)w_1(B)}_{T_1}\underbrace{r_2(A)w_2(A)r_2(B)w_2(B)}_{T_2}$

General proof element – the ability to swap sections of a schedule

Bernstein Conditions

Transaction: sequence of $r_i(x)$, $w_i(x)$ actions

Conflicting actions: $r_1(A) < w_2(A)$ $\quad w_2(A) < r_1(A)$ $\quad w_1(A) < w_2(A)$

(a.k.a. Bernstein Conditions, write-after-read, r.a.w., w.a.w.)

Schedule: represents chronological order in which actions are executed

Serial schedule: no interleaving of actions or transactions

However, for Sd:

$Sd=r_1(A)w_1(A)r_2(A)w_2(A)\ r_2(B)w_2(B)r_1(B)w_1(B)$

- as a matter of fact, T_2 must precede T_1 in any equivalent schedule, i.e., $T_2 \rightarrow T_1$

- $T_2 \rightarrow T_1$
- Also, $T_1 \rightarrow T_2$

$$\begin{tikzcd} T_1 \arrow[r, bend left] & T_2 \arrow[l, bend left] \end{tikzcd}$$

⇨ Sd cannot be rearranged into a serial schedule

⇨ Sd is not "equivalent" to any serial schedule

⇨ Sd is "bad"

Returning to Sc

$$Sc = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$$

$T_1 \rightarrow T_2$ $T_1 \rightarrow T_2$

☛ no cycles $\Rightarrow$ Sc is "equivalent" to a serial schedule (in this case T_1, T_2)

<u>Definition</u>

S_1, S_2 are <u>conflict equivalent</u> schedules if S_1 can be transformed into S_2 by a series of swaps on non-conflicting actions.

<u>Definition</u>

A schedule is <u>conflict serializable</u> if it is conflict equivalent to some serial schedule.

Precedence graph P(S) (S is schedule)

Nodes: transactions in S

Arcs: Ti → Tj whenever

- $p_i(A)$, $q_j(A)$ are actions in S
- $p_i(A) <_S q_j(A)$
- at least one of p_i, q_j is a write

Exercise:

- What is P(S) for
  $S = w_3(A)\ w_2(C)\ r_1(A)\ w_1(B)\ r_1(C)\ w_2(A)\ r_4(A)\ w_4(D)$
- Is S serializable?

Lemma

S_1, S_2 conflict equivalent $\Rightarrow P(S_1)=P(S_2)$

Proof:

Assume $P(S_1) \neq P(S_2)$

$\Rightarrow \exists\ T_i: T_i \rightarrow T_j$ in S_1 and not in S_2

$$\left.\begin{array}{l}\Rightarrow S_1 = \ldots p_i(A) \ldots q_j(A) \ldots \\ \quad S_2 = \ldots q_j(A) \ldots p_i(A) \ldots\end{array}\right\}\ \begin{array}{l}p_i, q_j \\ \text{conflict}\end{array}$$

$\Rightarrow S_1, S_2$ not conflict equivalent

Note: $P(S_1)=P(S_2) \not\Rightarrow S_1, S_2$ conflict equivalent

Counter example:

$S_1=w_1(A)\ r_2(A) \quad w_2(B)\ r_1(B)$

$S_2=r_2(A)\ w_1(A) \quad r_1(B)\ w_2(B)$

Serializability Theorem

$P(S_1)$ acyclic $\Longleftrightarrow$ S_1 conflict serializable

($\Leftarrow$) Assume S_1 is conflict serializable

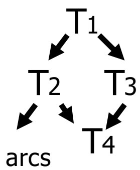
$\Rightarrow \exists\, S_s$: S_s, S_1 conflict equivalent // by def.

$\Rightarrow P(S_s) = P(S_1)$ // from lemma

$\Rightarrow$ // what does P(Ss) look like?

Theorem

$P(S_1)$ acyclic $\Longleftrightarrow$ S_1 conflict serializable



($\Rightarrow$) Assume $P(S_1)$ is acyclic

Transform S_1 as follows:

(1) Take T_1 to be transaction with no incident arcs

(2) Move all T_1 actions to the front

$S_1 = \ldots\ldots\; q_j(A)\ldots\ldots p_1(A)\ldots\ldots$

(3) we now have S_1 = < T_1 actions >< ... rest ... >

(4) repeat above steps to serialize rest!