CS 245: Database System Principles

Notes 11: View Serializability

Hector Garcia-Molina

View Serializability

Conflict equivalent	View equivalent
Conflict serializable	View serializable

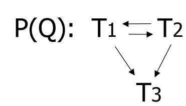
Motivating example

Schedule Q

T_1	T_2	T_3
Read(A)		
	Write(A)	
Write(A)		
		Write(A)

Same as

$Q = r_1(A)\ w_2(A)\ w_1(A)\ w_3(A)$



➠ Not conflict serializable!

But now compare Q to Ss, a serial schedule:

Q	T_1	T_2	T_3
	Read(A)		
		Write(A)	
	Write(A)		
			Write(A)

Ss	T_1	T_2	T_3
	Read(A)		
	Write(A)		
		Write(A)	
			Write(A)

- T_1 reads same thing in Q, Ss
- T_2, T_3 read samething (nothing?)
- After Q or Ss, DB is left in same state

➠ So what is wrong with Q?

Definition Schedules S_1,S_2 are View Equivalent if:

(1) If in S_1: $w_j(A) \Rightarrow r_i(A)$ then in S_2: $w_j(A) \Rightarrow r_i(A)$

$\Rightarrow$ means "reads value produced"

(2) If in S_1: $r_i(A)$ reads initial DB value, then in S_2: $r_i(A)$ also reads initial DB value

(3) If in S_1: T_i does last write on A, then in S_2: T_i also does last write on A

Definition

Schedule S_1 is View Serializable if it is view equivalent to some serial schedule

View Serializable $\longleftarrow$? $\longrightarrow$ Conflict Serializable

- View Serializable $\not\Rightarrow$ Conflict Serializable
  e.g., See Schedule Q
- Conflict Serializable $\overset{?}{\Rightarrow}$ View Serializable

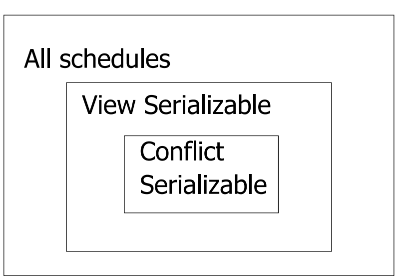
Lemma

Conflict Serializable $\Rightarrow$ View Serializable

Proof:

__Swapping non-conflicting actions does not change what transactions read nor final DB state

Venn Diagram



Note: All view serializable schedules that are not conflict serializable, involve useless write

S = $W_2(A)$ … $W_3(A)$…..

no reads

FALSE: Counterexample (Sorav Bansal): $w_3(Y)\ r_2(Y)\ w_1(X)\ r_2(X)\ w_3(X)\ r_4(X)\ w_5(X)$

How do we test for view-serializability?

- P(S) not good enough…
  (see schedule Q)

- One problem: some swaps involving conflicting actions are OK… e.g.:

$S = \ldots w_2(A) \ldots\ldots r_1(A) \ldots\ w_3(A) \ldots \quad w_4(A)$

this action can move
if this write exists

- Another problem: useless writes

$S = \ldots\ldots W_2(A) \ldots\ldots\ldots W_1(A) \ldots\ldots$

no A reads

To check if S is View Serializable

(1) Add final transaction T_f that reads all DB
(eliminates condition 3 of V-S definition)

E.g.: $S = \ldots\ldots W_1(A) \ldots\ldots\ldots W_2(A) \ldots \quad r_f(A)$

?

Last A write   add

(2) Add initial transaction T_b
that writes all DB
(eliminates condition 2 of V-S definition)

E.g.: $S = w_b(A) \ldots r_1(A) \ldots w_2(A) \ldots$

?

add

(3) Create labeled precedence graph of S:

(3a) If $w_i(A) \Rightarrow r_j(A)$ in S, add $T_i \xrightarrow{0} T_j$

(3b) For each $w_i(A) \Rightarrow r_j(A)$ do
consider each $w_k(A)$: $[T_k \neq T_b]$

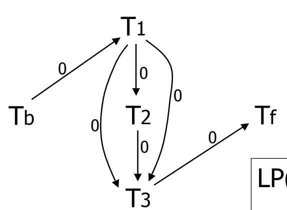
- If $T_i \neq T_b \wedge T_j \neq T_f$ then insert
  $\begin{cases} T_k \xrightarrow{p} T_i \\ T_j \xrightarrow{p} T_k \end{cases}$ some new p
- If $T_i = T_b \wedge T_j \neq T_f$ then insert
  $T_j \xrightarrow{0} T_k$
- If $T_i \neq T_b \wedge T_j = T_f$ then insert
  $T_k \xrightarrow{0} T_i$

(4) Check if LP(S) is "acyclic" (if so, S is V-S)
- For each pair of "p" arcs ($p \neq 0$), choose one

<u>Example:</u> check if Q is V-S:

$Q = r_1(A)\ w_2(A)\ w_1(A)\ w_3(A)$

$Q' = w_b(A) \Rightarrow r_1(A)\ w_2(A)\ w_1(A)\ w_3(A) \Rightarrow r_f(A)$

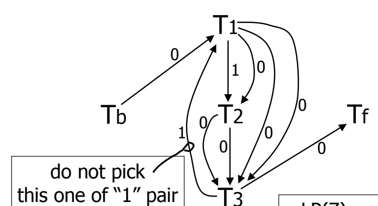


rule 3(a)
rule 3(b)
rule 3(b)

LP(S) acyclic!!
S is V-S

Another example:

$Z = w_b(A) \Rightarrow r_1(A)\ w_2(A) \Rightarrow r_3(A)\ w_1(A)\ w_3(A) \Rightarrow r_f(A)$



LP(Z) acyclic, so Z is V-S
(equivalent to $T_b\ T_1\ T_2\ T_3\ T_f$)

$S_s = w_b(A)\underbrace{r_1(A)w_1(A)}_{T_1}\underbrace{w_2(A)}_{T_2}\underbrace{r_3(A)w_3(A)}_{T_3}r_f(A)$

Z + S_s indeed do same thing

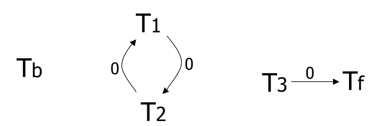
- Checking view serializability is expensive
- Still, V-S useful in some cases...

Example on useless transactions:

$S = w_1(A)\ r_2(A)\ w_2(B)\ r_1(B)\ w_3(A)\ w_3(B)$

$S' =$

$T_b\ w_1(A) \Rightarrow r_2(A) w_2(B) \Rightarrow r_1(B)\ w_3(A) w_3(B) \Rightarrow T_f$



- If we only care about final state ➽ remove T_1, T_2; i.e., remove useless transactions
- If we care what T_1, T_2 read (view equivalence), then do not remove useless transactions

- If all transactions read what they write, (I.e., $T_j = \ldots R_j(A) \ldots W_j(A) \ldots$) then view serializability = conf. serializability

[Another way of saying: blind writes appear in any view-serializable schedule that is not conflict serializable]

Proof(?): say S_1 is view-ser. and no blind writes. S_1 V-equiv to S_s, serial schedule.

(1) Goal: Show that
$T_1 \rightarrow T_2$ in $P(S_1) \Rightarrow T_1 <_{ss} T_2$
(2) Assume $T_1 \rightarrow T_2$
if $S_1 = \ldots w_1(A) \ldots r_2(A) \ldots$
(direct read) clearly $T_1 <_{ss} T_2$
if $S_1 = \ldots w_1(A) \ldots r_3(A)\ w_3(A) \ldots r_2(A) \ldots$
also $T_1 <_{ss} T_2$
if $S_1 = \ldots r_1(A)\ r_3(A) \ldots w_1(A) \ldots w_3(A) \ldots r_2(A)$
not possible: T_1, T_3 not serializable
Other cases similar...

Implications:

If no blind writes, view-ser $\Longleftrightarrow$ conf-ser

$P(S)$ acyclic $\Rightarrow$ all transactions read the same as in a serial schedule