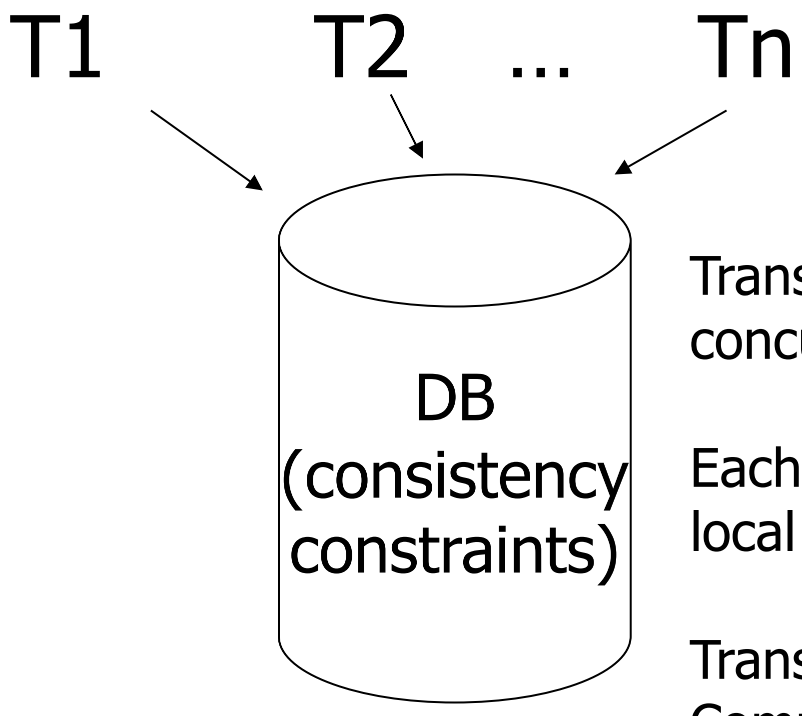
Concurrency Control

Based on notes by Hector Garcia-Molina

Concurrency Control



Transactions execute concurrently

Each transaction has own local memory

Transactions do not Communicate

Example:

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

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

Constraint: $A=B$

Schedule A

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

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 C

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

Schedule C

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

Schedule D

T1	T2
Read(A); $A \leftarrow A+100$	
Write(A);	
	Read(A);$A \leftarrow A\times2$;
	Write(A);
	Read(B);$B \leftarrow B\times2$;
	Write(B);
Read(B); $B \leftarrow B+100$;	
Write(B);	

Schedule D

T1	T2	A	B
		25	25
Read(A); A $\leftarrow$ A+100			
Write(A);		125	
	Read(A);A $\leftarrow$ A$\times$2;		
	Write(A);	250	
	Read(B);B $\leftarrow$ B$\times$2;		
	Write(B);		50
Read(B); B $\leftarrow$ B+100;			
Write(B);			150
		250	150

Schedule E

Same as Schedule D but with new T2′

T1	T2′
Read(A); $A \leftarrow A+100$	
Write(A);	
	Read(A);$A \leftarrow A \times 1$;
	Write(A);
	Read(B);$B \leftarrow B \times 1$;
	Write(B);
Read(B); $B \leftarrow B+100$;	
Write(B);	

Schedule E

Same as Schedule D but with new T2′

T1	T2′	A	B
		25	25
Read(A); $A \leftarrow A+100$			
Write(A);		125	
	Read(A); $A \leftarrow A \times 1$;		
	Write(A);	125	
	Read(B); $B \leftarrow B \times 1$;		
	Write(B);		25
Read(B); $B \leftarrow B+100$;			
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

Example:

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

Example:

$$Sc=r_1(A)w_1(A)\underbrace{r_2(A)w_2(A)}\underbrace{r_1(B)w_1(B)}r_2(B)w_2(B)$$

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

However, for Sd:

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

Concepts

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

Conflicting actions: $r_i(A)$ / $w_j(A)$ $w_i(A)$ / $r_j(A)$ $w_i(A)$ / $w_j(A)$ $i \neq j$

Schedule: represents chronological order in which actions are executed

Serial schedule: no interleaving of actions or transactions

Definition

S_1, S_2 are conflict equivalent schedules

if S_1 can be transformed into S_2 by a series of swaps on non-conflicting actions.

Definition

S_1 is serializable if there is a serial schedule S_s such that for every initial database state, the effects of S_1 and S_s are the same

Definition

A schedule is conflict serializable if it is conflict equivalent to some serial schedule.

Example: Sc is conflict serializable

$$Sc = r_1(A)w_1(A)\underbrace{r_2(A)w_2(A)}\underbrace{r_1(B)w_1(B)}r_2(B)w_2(B)$$

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

Sd is not conflict serializable:

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

Precedence graph P(S) (S is schedule)

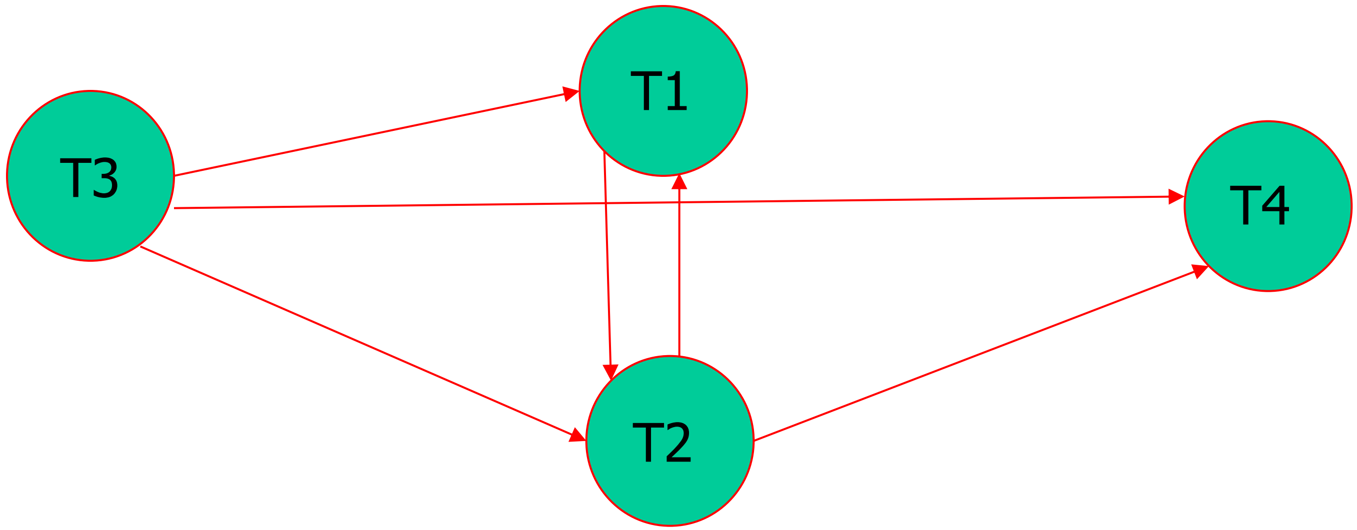
Nodes: transactions in S

Arcs: $T_i \rightarrow T_j$ 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:

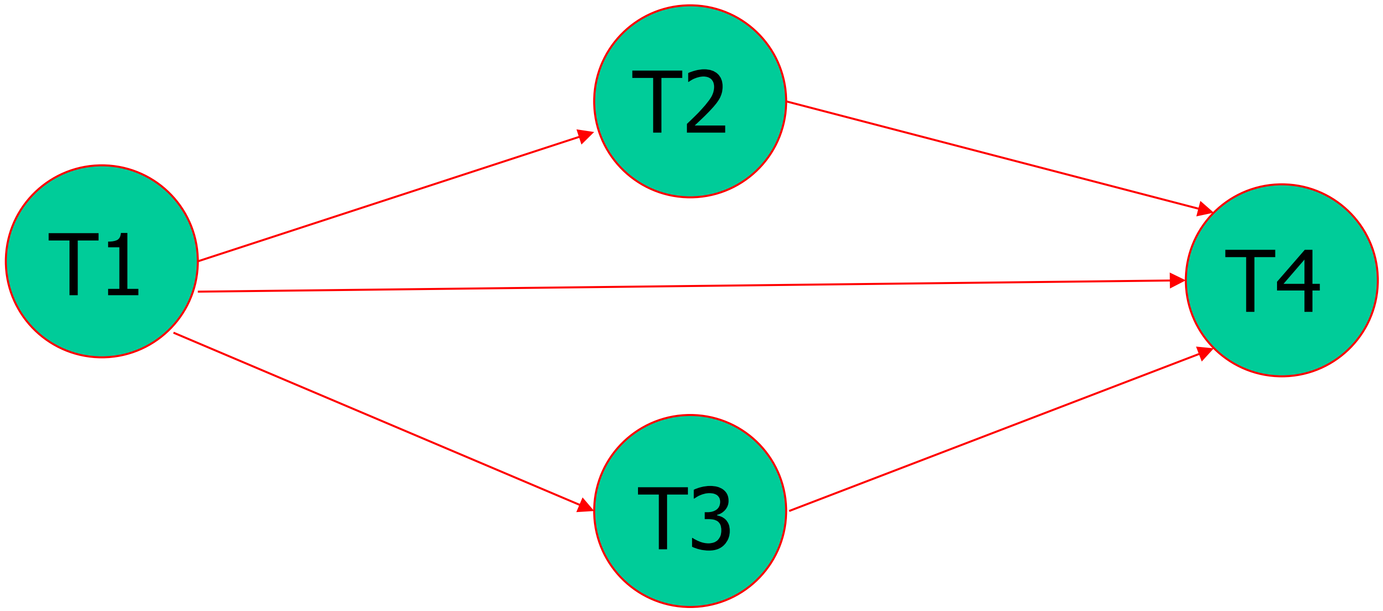
- What is P(S) for
  $S = w_3(A)\ w_2(C)\ r_1(A)\ w_1(B)\ r_1(C)\ w_2(A)\ r_4(A)\ w_4(D)$



- Is S serializable?

Another Exercise:

- What is P(S) for
  $S = w_1(A)\ r_2(A)\ r_3(A)\ w_4(A)$?



Lemma

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

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

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

$\Rightarrow S_1, S_2$ not conflict equivalent

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

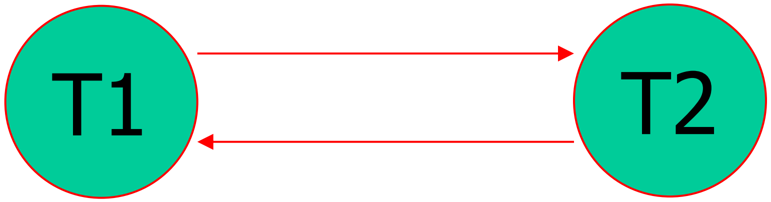
Note: $P(S_1)=P(S_2) \not\Rightarrow S_1, S_2$ conflict equivalent

<u>Counter example:</u>

T1 := w1(A) r1(B)   T2 := r2(A) w2(B)

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

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



$P(S_1) = P(S_2)$

Serialization Theorem

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

Serialization Theorem

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

($\Leftarrow$) Assume S_1 is conflict serializable

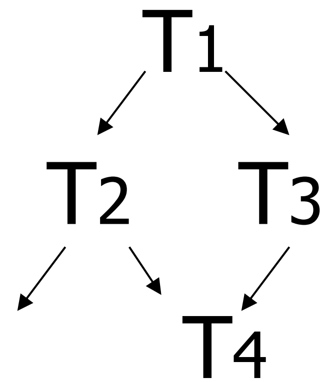
$\Rightarrow \exists\ S_s$: S_s, S_1 conflict equivalent

$\Rightarrow P(S_s) = P(S_1)$

$\Rightarrow P(S_1)$ acyclic since $P(S_s)$ is acyclic

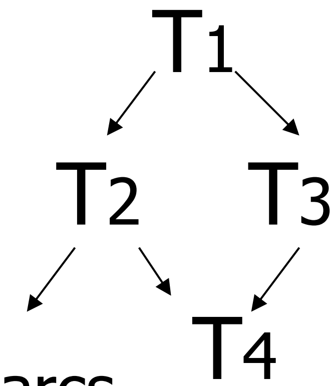
Serialization Theorem

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



Serialization 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

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

(3) we now have $S_1 = <T_1 \text{ actions} ><\ldots \text{ rest } \ldots>$

(4) repeat above steps to serialize rest!

Serialization Theorem

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

How to enforce serializable schedules?

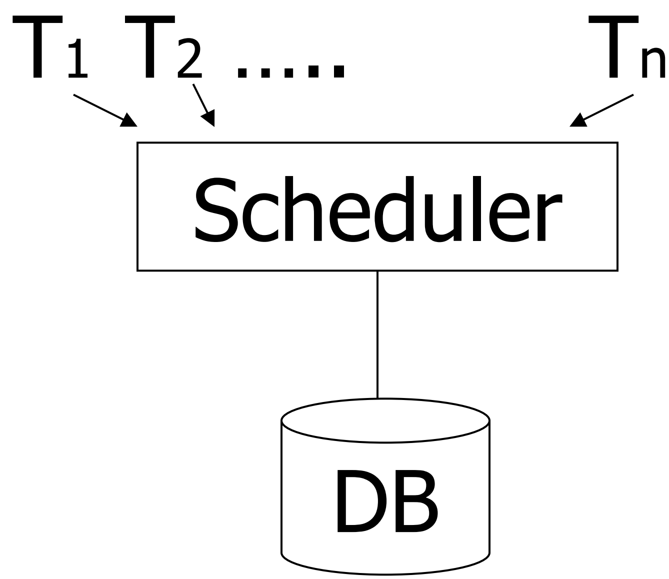
Option 1: run system, recording P(S);
after some time,

check for P(S) cycles and declare

if execution was good

How to enforce serializable schedules?

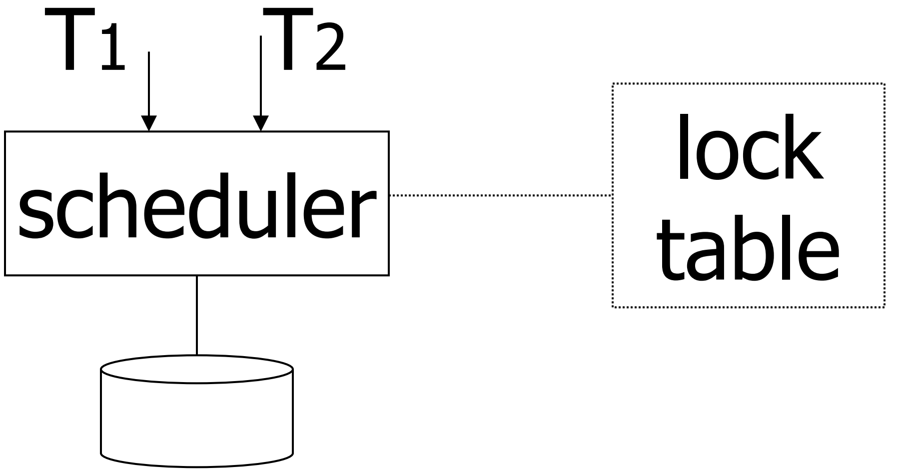
Option 2: prevent P(S) cycles from occurring



A locking protocol

Two new actions:

- lock (exclusive): l_i (A)
- unlock: u_i (A)



Rule #1: Well-formed transactions

T_i: … $l_i(A)$ … $p_i(A)$ … $u_i(A)$ …

A transaction can only read or write if it first as been granted a lock.

If a transaction locks an element, it must later unlock it.

Rule #2 Legal scheduler

$S = \ldots\ldots\ldots\; l_i(A)\; \ldots\ldots\ldots\ldots\; u_i(A)\; \ldots\ldots\ldots$

no $l_j(A)$

Locks must have their intended semantics: no 2 transactions may have locked the same element without one having first released the lock.

Exercise:

- What schedules are legal?
  What transactions are well-formed?

$S1 = l_1(A)l_1(B)r_1(A)w_1(B)l_2(B)u_1(A)u_1(B)$
$r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)$

$S2 = l_1(A)r_1(A)w_1(B)u_1(A)u_1(B)$
$l_2(B)r_2(B)w_2(B)l_3(B)r_3(B)u_3(B)$

$S3 = l_1(A)r_1(A)u_1(A)l_1(B)w_1(B)u_1(B)$
$l_2(B)r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)$

Exercise:

- What schedules are legal?
  What transactions are well-formed?

  S1 = $l_1(A)l_1(B)r_1(A)w_1(B)l_2(B)u_1(A)u_1(B)$
  $r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)$

  S2 = $l_1(A)r_1(A)w_1(B)u_1(A)u_1(B)$
  $l_2(B)r_2(B)w_2(B)l_3(B)r_3(B)u_3(B)$ $u_2(B)$?

  S3 = $l_1(A)r_1(A)u_1(A)l_1(B)w_1(B)u_1(B)$
  $l_2(B)r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)$

Do rules 1 and 2 guarantee serializability?

Schedule F

T1	T2
$l_1(A)$;Read(A)	
$A \leftarrow A+100$;Write(A);$u_1(A)$	
	$l_2(A)$;Read(A)
	$A \leftarrow Ax2$;Write(A);$u_2(A)$
	$l_2(B)$;Read(B)
	$B \leftarrow Bx2$;Write(B);$u_2(B)$
$l_1(B)$;Read(B)	
$B \leftarrow B+100$;Write(B);$u_1(B)$	

Schedule F

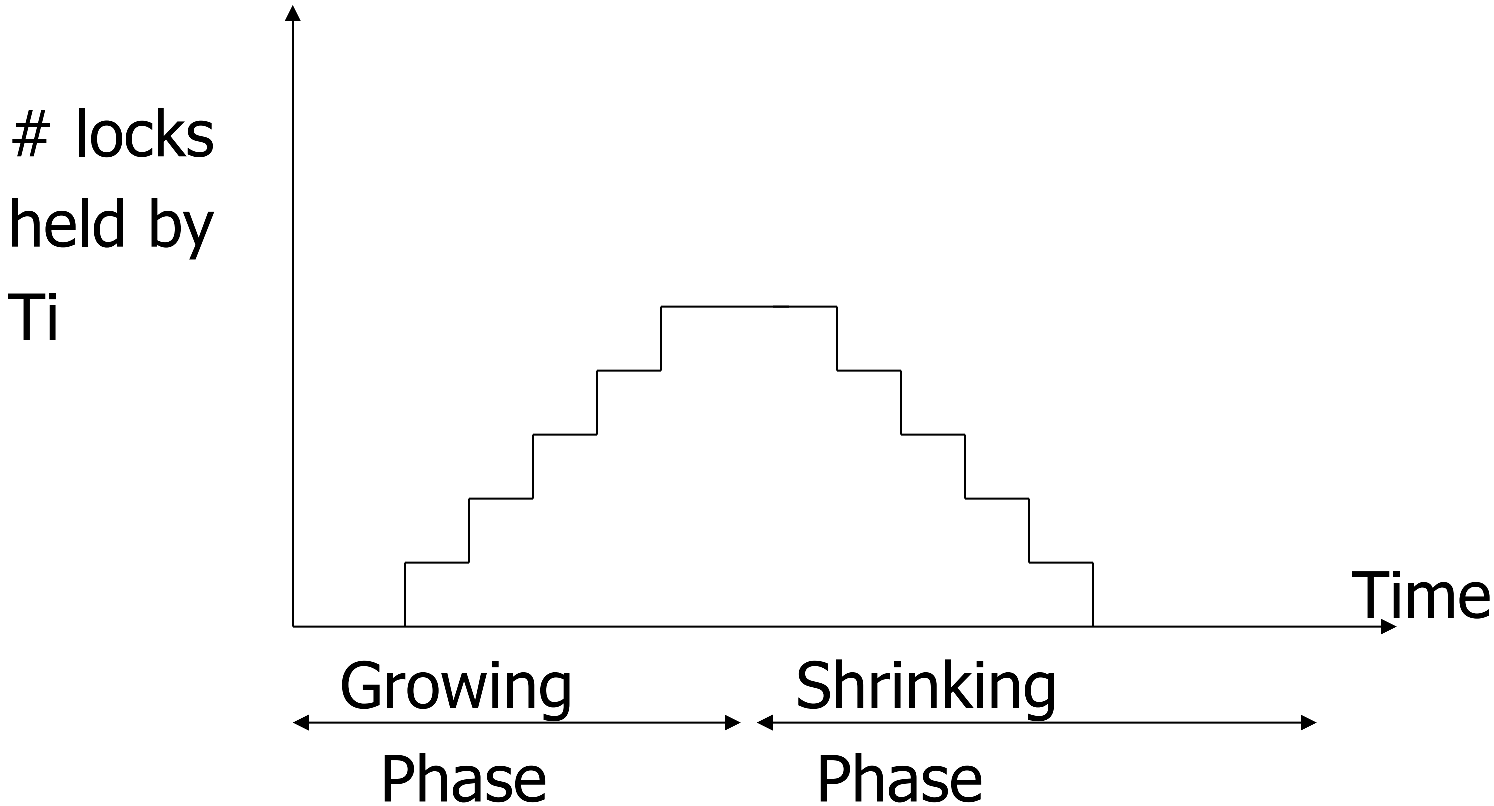
		A	B
T1	T2	25	25
$l_1(A)$;Read(A)			
$A \leftarrow A+100$;Write(A);$u_1(A)$		125	
	$l_2(A)$;Read(A)		
	$A \leftarrow A\text{x}2$;Write(A);$u_2(A)$	250	
	$l_2(B)$;Read(B)		
	$B \leftarrow B\text{x}2$;Write(B);$u_2(B)$		50
$l_1(B)$;Read(B)			
$B \leftarrow B+100$;Write(B);$u_1(B)$			150
		250	150

Rule #3 Two phase locking (2PL)

for transactions

T_i = ... $l_i(A)$ .... $u_i(A)$ ....

no unlocks   no locks


locks
held by
Ti
Time
Growing
Phase
Shrinking
Phase

Schedule G

T1	T2
$l_1(A)$;Read(A)	
$A \leftarrow A+100$;Write(A)	
$l_1(B)$; $u_1(A)$	
	delayed
	$l_2(A)$;Read(A)
	$A \leftarrow Ax2$;Write(A); $l_2(B)$

Schedule G

T1	T2
$l_1(A)$;Read(A)	
$A \leftarrow A+100$;Write(A)	
$l_1(B)$; $u_1(A)$	
	$l_2(A)$;Read(A)
	$A \leftarrow Ax2$;Write(A);$l_2(B)$
Read(B);$B \leftarrow B+100$	
Write(B); $u_1(B)$	

delayed

Schedule G

T1	T2
$l_1(A)$;Read(A)	
$A \leftarrow A+100$;Write(A)	
$l_1(B)$; $u_1(A)$	
	$l_2(A)$;Read(A)
	$A \leftarrow Ax2$;Write(A);$l_2(B)$ delayed
Read(B);$B \leftarrow B+100$	
Write(B); $u_1(B)$	
	$l_2(B)$; $u_2(A)$;Read(B)
	$B \leftarrow Bx2$;Write(B);$u_2(B)$;

Schedule H (T2 reversed)

T1	T2
$l_1(A)$; Read(A)	$l_2(B)$;Read(B)
$A \leftarrow A+100$;Write(A)	$B \leftarrow Bx2$;Write(B)
$l_1(B)$	$l_2(A)$
delayed	delayed

Schedule is deadlocked since neither T1 nor T2 can proceed!

2PL guarantees serializability

Show that rules #1,2,3 $\Rightarrow$ conflict-serializable schedules

Conflict rules for $l_i(A)$, $u_i(A)$:

- $l_i(A)$, $l_j(A)$ conflict
- $l_i(A)$, $u_j(A)$ conflict

Note: no conflict $< u_i(A), u_j(A)>$, $< l_i(A), r_j(A)>$,...

Theorem Rules #1,2,3 $\Longrightarrow$ conflict
(2PL) serializable
schedule

Theorem Rules #1,2,3 $\Rightarrow$ conflict
(2PL) serializable
schedule

To help in proof:

Definition Shrink(Ti) = SH(Ti) =
first unlock action of Ti

Lemma

$Ti \rightarrow Tj$ in $S \Rightarrow SH(Ti) <_S SH(Tj)$

Lemma

$T_i \rightarrow T_j$ in $S \Rightarrow SH(T_i) <_S SH(T_j)$

Proof of lemma:

$T_i \rightarrow T_j$ means that

$S = \ldots\ p_i(A) \ldots\ q_j(A) \ldots;\quad p,q$ conflict

By rules 1,2:

$S = \ldots\ p_i(A) \ldots\ u_i(A) \ldots\ l_j(A) \ldots\ q_j(A) \ldots$

Lemma

$Ti \rightarrow Tj$ in $S \Rightarrow SH(Ti) <_S SH(Tj)$

Proof of lemma:

$Ti \rightarrow Tj$ means that

$S = \ldots p_i(A) \ldots q_j(A) \ldots;$ p,q conflict

By rules 1,2:

$S = \ldots p_i(A) \ldots u_i(A) \ldots l_j(A) \ldots q_j(A) \ldots$

By rule 3: SH(Ti) SH(Tj)

So, $SH(Ti) <_S SH(Tj)$

Theorem Rules #1,2,3 $\Longrightarrow$ conflict
(2PL) serializable
schedule

Proof:

(1) Assume P(S) has cycle

$$T_1 \rightarrow T_2 \rightarrow \ldots T_n \rightarrow T_1$$

(2) By lemma: $SH(T_1) < SH(T_2) < \ldots < SH(T_1)$

(3) Impossible, so P(S) acyclic

(4) $\Rightarrow$ S is conflict serializable