9. Concurrency Control

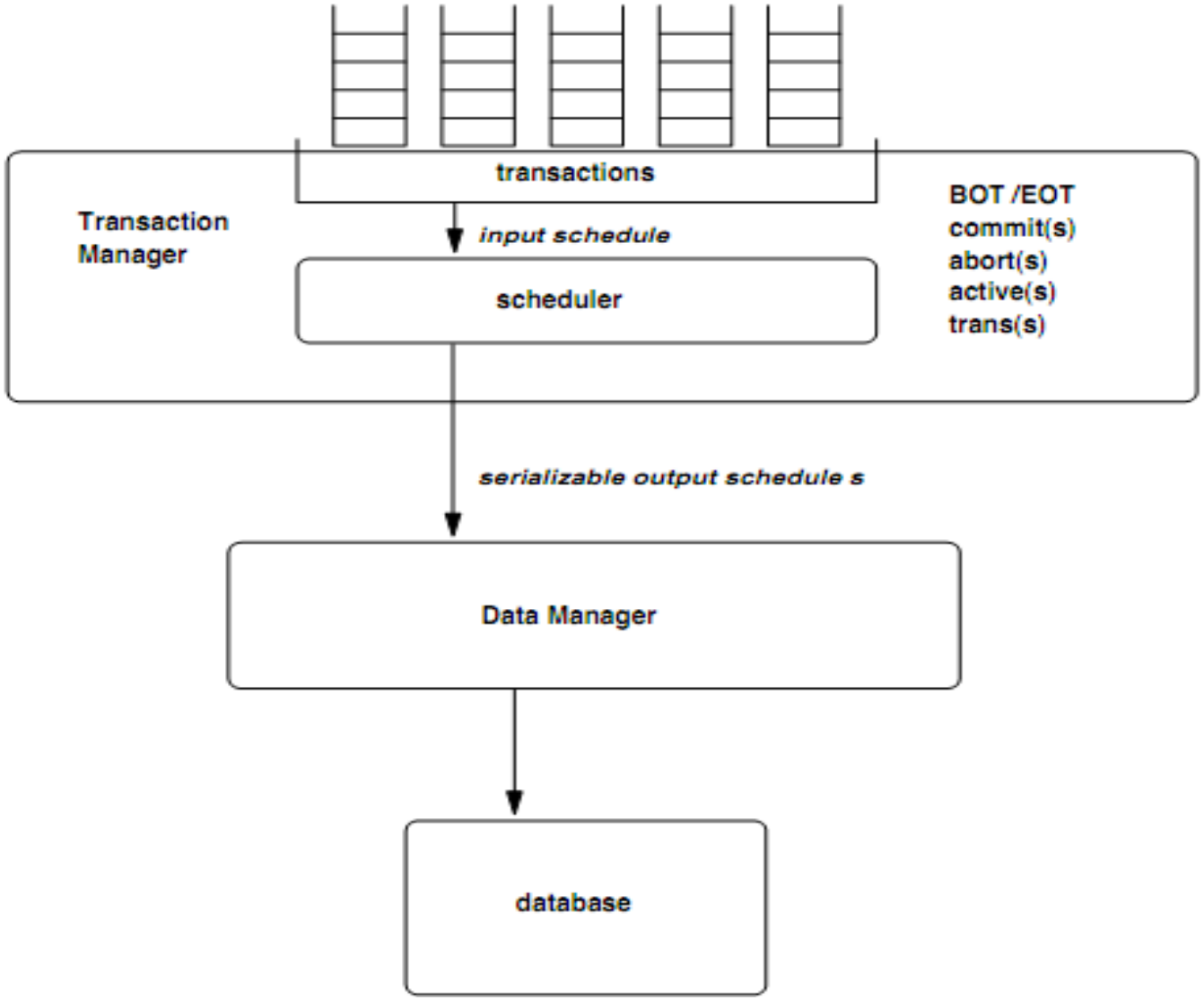
التحكم بالوصول المتزامن

Dr. Eng. Ramez Alkhatib

ramezalkhatib@hotmail.com

التحكم بالوصول المتزامن
Concurrency Control

- برتوكولات التحكم بالوصول المتزامن هي عبارة عن خوارزميات أو مجموعة من القواعد المطبقة بواسطة جدولة نظم إدارة قواعد البيانات (DBMS scheduler) من اجل انتاج فقط جدولات قابلة للتسلسل



التحكم بالوصول المتزامن
Concurrency Control

- البروتوكولات المرتكزة على الأقفال Lock
- البروتوكولات المرتكزة على الإقفال على مرحلتين Tow Phase Locking
- البروتوكولات المرتكزة على البيان graph based Protocol (البرتوكول الشجري)
- البروتوكولات المرتكزة على الأختام Timestamp
- التعامل مع العرقلة المتبادلة Deadlock
- عمليات الإضافة والحذف
- التزامن في بنية الفهارس

البروتوكولات المرتكزة على الأقفال

- الأقفال : هي تقنية تُستخدم للتحكم بالوصول المتزامن للمعطيات.
- يوجد نوعان لإقفل المعطيات :
  - ★ *exclusive (X)* صريح وتام (إقفال كلي) تتم العملية باستخدام التعليمة **lock-X**
  - ★ *shared (S)* جزئي ومشترك (إقفال جزئي مشترك) وتتم العملية باستخدام التعليمة **lock-S**

– Read locks can be **shared** by several transactions

– Write locks are **exclusive** locks

- **يجري طلب الإقفال من قبل إدارة التحكم بالوصول المتزامن. وتُعالج المناقلات فقط بعد تنفيذ طلب الإقفال (منح الأقفال).**

		Lock requested	
		read lock	write lock
Lock held	read lock	yes	no
	write lock	no	no

البروتوكولات المرتكزة على الأقفال

	S	X
S	true	false
X	false	false

- مصفوفة تجانس الأقفال Lock-compatibility

- يمكن لمناقلة أن تحصل على قفل على عنصر item إذا كان طلب الإقفال متوافق مع الأقفال الموضوعة سابقاً على العنصر من قبل مناقلات أخرى.
- يمكن لعدة مناقلات أن تضع قفلاً مشتركاً shared lock على عنصر، وبالمقابل إذا وضعت مناقلة قفلاً من نوع كلي exclusive على عنصر لا يمكن لأية مناقلة أخرى وضع أي قفل على هذا العنصر.
- إذا لم تتمكن مناقلة من الحصول على القفل فإنها تنتظر تحرير جميع الأقفال التي وضعتها المناقلات الأخرى على هذا العنصر قبل متابعة التنفيذ.

البروتوكولات المرتكزة على الأقفال

- مثال (طلب إقفال)

T_2: **lock-S***(A)*;
read *(A)*;
unlock*(A)*;
lock-S*(B)*;
read *(B)*;
unlock*(B)*;
display*(A+B)*

- إن طلب الإقفال السابق غير كاف لتحقيق التسلسلية- في حال جرى تعديل A, B بين عمليتي القراءة لـ A,B نتيجة المجموع المعروضة تكون خاطئة.
- بروتوكول الإقفال **locking protocol**: هو مجموعة من القواعد المُتبعة من قبل جميع المناقلات عند طلب وتحرير الأقفال. يحدد بروتوكول الإقفال مجموعة مخططات التنفيذ الممكنة.

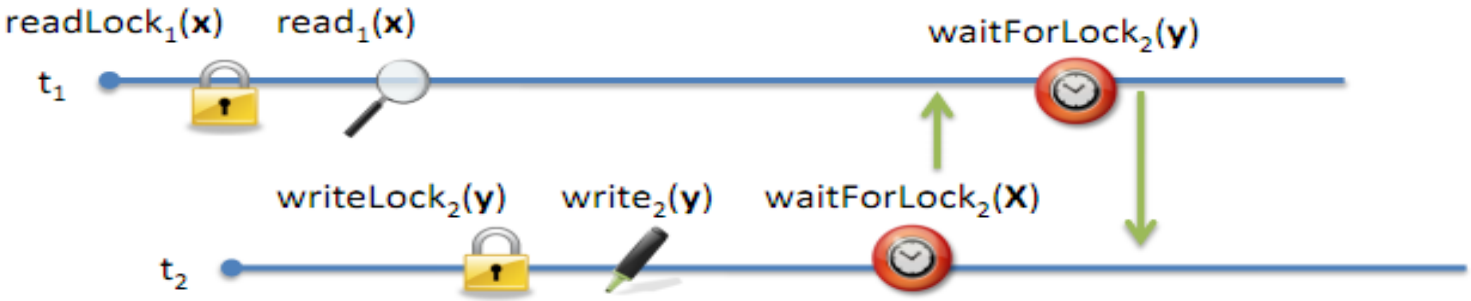
مخططات = جدولات

الصعوبات والمشاكل في البروتوكولات المرتكزة على الأقفال

T_3	T_4
lock-X(B)	
read(B)	
$B := B - 50$	
write(B)	
	lock-S(A)
	read(A)
	lock-S(B)
lock-X(A)	

- لا يمكن لكل من T_3 أو T_4 أن تتقدما- إن طلب تنفيذ **lock-S**(B) يجعل T_4 تنتظر T_3 أن تحرر قفلها على B، بينما تنفيذ تعليمة **lock-X**(A) تجعل T_3 تنتظر T_4 أن تحرر قفلها على A.
- تدعى مثل تلك الحالة بـ العرقلة المتبادلة **deadlock**

★ في هذه الحالة يجب ان يتم تنفيذ تعليمة العودة عن التنفيذ rollback لإحدى المناقلتين وتحرير الأقفال المحجوزة لها.

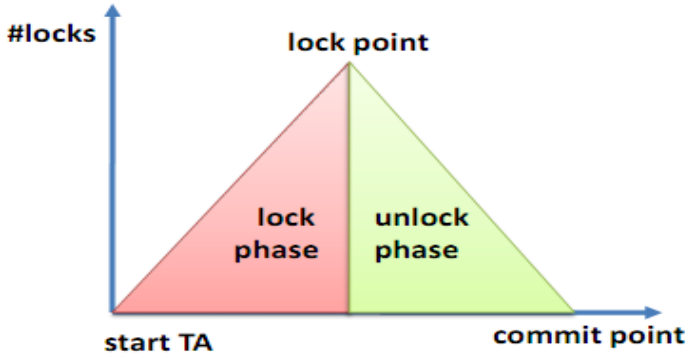


الصعوبات والمشاكل في البروتوكولات المرتكزة على الأقفال

- **الحرمان Starvation** (عدم القدرة على الوصول إلى حالة اكتفاء) ممكن حدوثه في حال كان تصميم إدارة التحكم المتزامن ليس جيداً، مثال :
  - ★ وجود مناقلة تنتظر لحجز X-lock على عنصر، في الوقت الذي حصلت مناقلة أخرى على قفل مشترك S-lock لنفس العنصر، ووجود مناقلات أخرى قد طلبت الحصول على قفل مشترك لنفس العنصر بشكل متسلسل، وبالتالي ستحصل عليه ويجب على المناقلة الأولى أن تنتظر تحرير القفل على العنصر من جميع المناقلات للحصول على القفل من نوع الكامل. (يمكن أن لا تحصل عليه أبداً).
- يمكن تصميم إدارة التحكم بالوصول المتزامن بحيث تتجنب الوصول إلى حالة الحرمان starvation بالشكل التالي :
  - ★ عند طلب مناقلة T_i إقفال عنصر Q بنوع معين (مشترك أو كامل)، يجب التأكد قبل منح القفل لها مما يلي :
    - عدم وجود مناقلة أخرى حصلت على قفل العنصر Q من نوع متناقض مع نوع القفل المطلوب؛
    - عدم وجود مناقلة أخرى تنتظر الحصول على قفل العنصر Q، وجرى طلب الإقفال فيها قبل T_i.

بروتوكول الأقفال على مرحلتين
Two phase locking Protocol

- هو بروتوكول يؤمن الحصول على مخططات متسلسلة تصادمياً conflict-serializable schedules
- المرحلة الأولى : مرحلة الإنشاء والنمو Growing Phase
  - ★ تحصل المناقلة على أقفال
  - ★ لا تحرر المناقلة في هذه المرحلة الأقفال
- مرحلة التقلص Shrinking Phase
  - ★ تحرر المناقلة الأقفال
  - ★ لا تحصل المناقلة على أقفال



- تقوم المناقلة بقفل جميع العناصر التي تحتاج إليها في المرحلة الأولى، وعندما تدخل مناقلة في مرحلة التقلص لن تعود إلى طلب أي قفل.
- يمكن التحقق من أن برتوكول الإقفال على مرحلتين يؤمن تنفيذ متسلسل تصادمياً، حيث يمكن تسلسل المناقلات بترتيب نقاط الإقفال **lock points** (مكان حصول المناقلة على آخر قفل تحتاجه).

بروتوكول الأقفال على مرحلتين

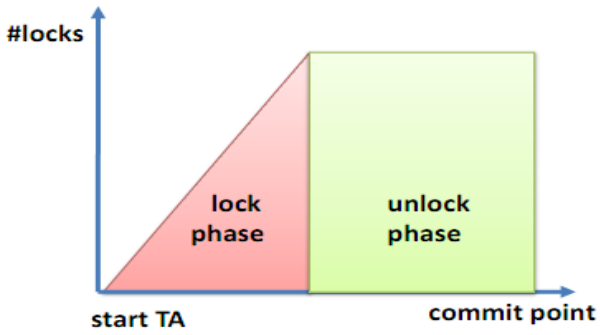
- لا يحرر برتوكول الإقفال على مرحلتين من مشكلة العرقلة المتبادلة **deadlock**.
- مثال :

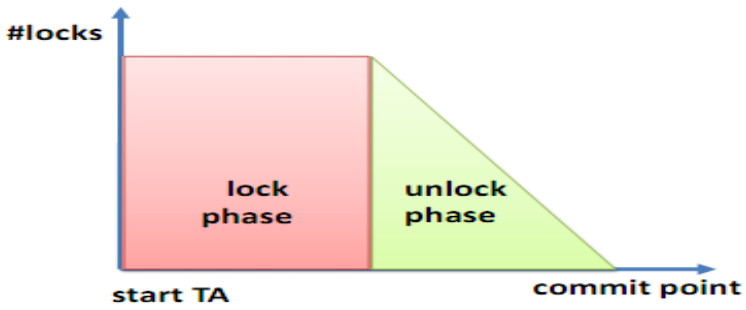
T3	T4
Lock-X(B) Read(B) B:=B-50 Write(B)	
	lock-S(A) Read(A) Lock-S(B)
Lock-X(A)	

المناقلتان T3, T4 تحققان بروتوكول الإقفال على مرحلتين، ولدينا في مخطط التنفيذ مشكلة العرقلة المتبادلة.

بروتوكول الأقفال على مرحلتين

- يمكن أن يحوي مخطط التنفيذ المحقق للإقفال على مرحلتين مشكلة العودة بشكل شلالCascading roll-back، يجري تجنب هذه المشكلة باستخدام بروتوكول معدّل يُسمى بالبروتوكول التام للإقفال على مرحلتين **strict two-phase locking** يتضمن عملية الإقفال على مرحلتين مع المحافظة على الأقفال الكلية Exclusive lock في المناقلة حتى القيام بعملية commit أو rollback.
- **البروتوكول الصارم للإقفال على مرحلتين Rigorous two-phase locking** : هو بروتوكول أكثر دقةً حيث أنه يجري المحافظة على جميع الأقفال حتى تنفيذ تعليمة commit/ abort . باستخدام هذا البرتوكول يتم تسلسل المناقلات وفقاً لترتيب تسجيلها (commit).





strict

بروتوكول الأقفال على مرحلتين

- مخطط تنفيذ يعتمد برتوكول الإقفال على مرحلتين.
- عودة بشكل شلال cascading rollback، في حال حدوث عطل في T5 بعد read (A) في T_7.
- يمكن تجنب عملية العودة بشكل شلال باستخدام بروتوكول التام للإقفال على مرحلتين strict two phase locking protocol (عدم تحرير الأقفال Exclusive إلا قبل عملية commit أو rollback)

T_5	T_6	T_7
lock-X(A)		
read(A)		
lock-S(B)		
read(B)		
write(A)		
unlock(A)		
	lock-X(A)	
	read(A)	
	write(A)	
	unlock(A)	
		lock-S(A)
		read(A)

التحصيل الآلي للاقفال

- تقوم المناقلة بتنفيذ عمليات القراءة والكتابة دون استدعاء صريح للأقفال.
- تُنفذ عملية القراءة read(D) بالشكل التالي :

if T_i has a lock on D **then**

read(D)

else

begin

if necessary

wait until no other transaction has a **lock-X** on D

grant T_i a **lock-S** on D;

read(D)

end

التحصيل الآلي للاقفال

- تُنفذ عملية الكتابة **write**(*D*) بالشكل التالي

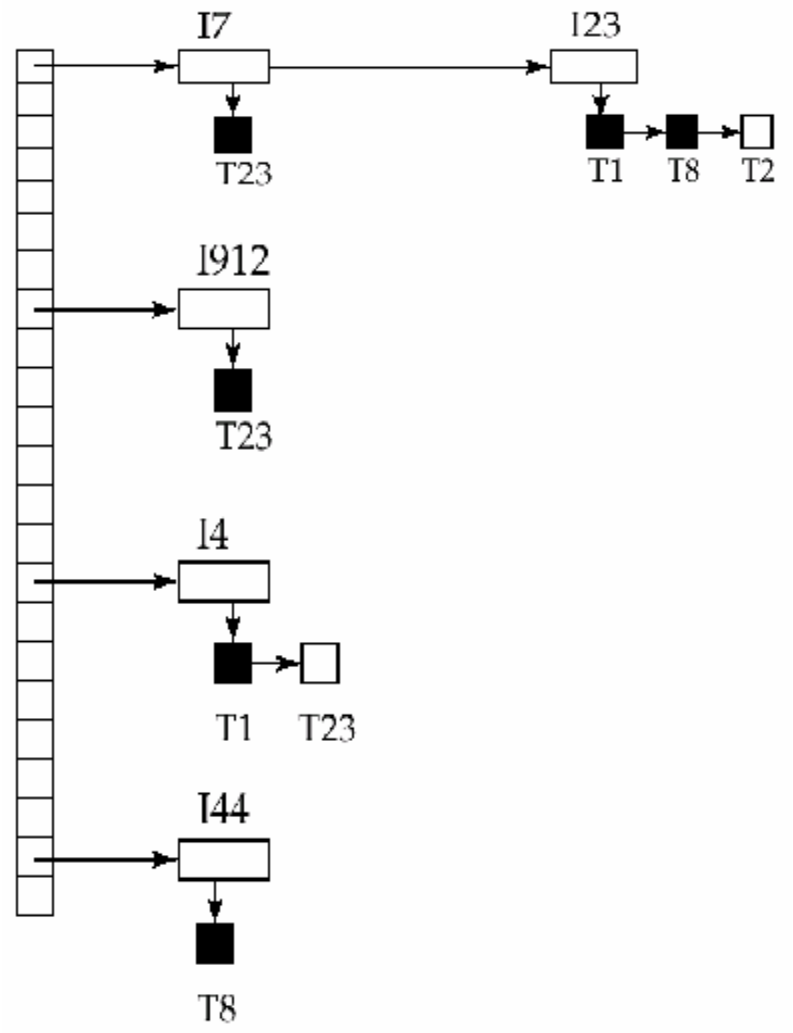
```
if T_i has a lock-X on D then
    write(D)
else
    begin
      if necessary
        wait until no other trans. has any lock on D,
        if T_i has a lock-S on D then
          upgrade lock on D to lock-X
        else
          grant T_i a lock-X on D
        write(D)
    end;
```

- تُحرر جميع الأقفال بعد تعليمة commit or abort

تنفيذ الاقفال

- يجري تنفيذ إدارة الأقفال من قبل مدير الأقفال **Lock manager** **وتتضمن معالجة منفصلة لطلبات الإقفال والتحرير المرسلة من قبل المناقلات**
- **يجيب** مدير الأقفال على طلب إقفال بإرسال رسالة منح قفل (أو رسالة تطلب من المناقلة أن تعود بالتنفيذ rollback في حالة العرقلة المتبادلة)
- تنتظر المناقلة الطالبة حتى تحصل على جواب لطلبها
- يستخدم مدير الأقفال بنية تُسمى جدول الأقفال lock table لتسجيل الأقفال الممنوحة و الطلبات المعلقة أو المنتظرة.
- عادةً يُنظم جدول الأقفال lock table بطريقة مفهرسة كجدول تقطيع **in-memory hash table** على حقل المعطيات المطلوب إقفاله.

جدول الاقفال
Lock Table



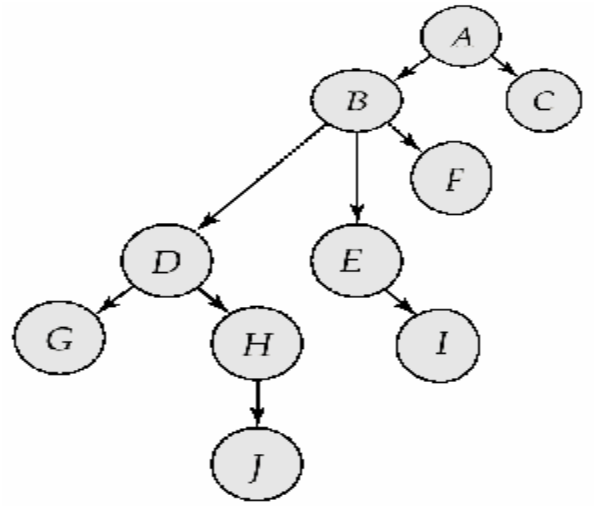
- تشير المستطيلات السوداء إلى الأقفال الممنوحة، والمستطيلات البيضاء إلى الطلبات المنتظرة
- يسجل جدول الأقفال نوع القفل الممنوح أو المطلوب
- يُضاف الطلب الجديد إلى نهاية سلسلة الطلبات المعنية بالعنصر، ويُمنح الطلب إذا كان متجانس مع جميع الأقفال السابقة
- طلبات تحرير الأقفال تؤدي إلى حذف طلب الإقفال واختبار إمكانية منح الاقفال المطلوبة من الطلبات اللاحقة
- في حال انتهى عمل مناقلة abort ، تُحذف جميع الطلبات الممنوحة أو المنتظرة المتعلقة بهذه المناقلة

★ يمكن لمدير الاقفال الاحتفاظ بقائمة الأقفال الممنوحة لكل مناقلة .

البروتوكولات المرتكزة على البيان
Graph Based Protocol

- تُستخدم البروتوكولات المرتكزة على البيان كبديل للبروتوكولات المرتكزة على الإقفال على مرحلتين.
- بفرض وجود ترتيب جزئي → على مجموعة من عناصر المعطيات
  D = {d1, d2 ,..., dh} (هذا الترتيب نتيجة التنظيم الفيزيائي أو المنطقي للمعطيات مثلاً)
  - ★ في حال كان لدينا di → dj فإن أية مناقلة تستخدم المعطيات di و dj معاً يجب أن تصل إلى di قبل أن تصل إلى dj .
  - ★ يمكن تمثيل مجموعة العناصر D كبيان موجه بدون حلقات directed acyclic graph يُسمى بيان قاعدة المعطيات *database graph*.
- البرتوكول الشجري *tree-protocol* هو حالة بسيطة من بروتوكول البيان graph protocol.

Tree Protocol



- يسمح فقط بالأقفال الكلية exclusive locks
- يمكن أن تطلب المناقلة T_i أول قفل على أي عنصر من المعطيات، يمكن لاحقاً قفل عنصر Q من قبل المناقلة T_i فقط في حال كان والد Q مقفول من قبل T_i.
- يمكن تحرير الأقفال في أي وقت، والمعطيات التي يجري تحريرها من قبل مناقلة لا يمكن إعادة إقفالها
- جميع المخططات التي تحقق البروتوكول الشجري هي مخططات متسلسلة تصادمياً

البروتوكولات المرتكزة على الأختام
Timestamp Based Protocol

- تُعطى كل مناقلة ختم عند دخولها إلى النظام. في حال جرى إعطاء مناقلة قديمة T_i ختم $TS(T_i)$، تُعطى مناقلة جديدة T_j ختم آخر $TS(T_j)$ بحيث يكون $TS(T_i) < TS(T_j)$
- يقوم البروتوكول بإدارة تنفيذ الوصول المتزامن حيث أنه يجري استخدام الأختام في تحديد ترتيب إمكانية التسلسل.
- لتأمين ما سبق يقوم البروتوكول من أجل كل عنصر مثل Q بربط قيمتين للختم :
  - ★ **W-timestamp**(Q) وهو أكبر قيمة للختم من أية قيمة معطاة لمناقلة نفذت write(Q) بنجاح.
  - ★ **R-timestamp**(Q) وهو أكبر قيمة للختم من أية قيمة معطاة لمناقلة نفذت read(Q) بنجاح.

البروتوكولات المرتكزة على الأختام

- يؤكد بروتوكول ترتيب الأختام على أن تنفيذ أية عمليات متصادمة read, write يجري وفق ترتيب الختم
- **بمعنى أن لا تقوم مناقلة ذات ختم أقل بعملية قراءة لعنصر جرى كتابته من مناقلة ذات ختم أكبر، وأن لا تقوم مناقلة ذات ختم أصغر بعملية كتابة لمعطيات جرى قراءتها أو كتابتها من مناقلة ذات ختم أكبر. مما يسمح بجعل المخطط يكافئ مخطط متعاقب مرتب وفق الأختام.**
- يعمل البرتوكول وفق ما يلي :
- بفرض أن المناقلة T_i تنفذ عملية قراءة لعنصر **read**(Q) :

1. If TS(T_i) < **W**-timestamp(Q) then

★ هذا يعني أن المناقلة Ti تحتاج لقراءة Q التي جرى تسجيلها (الكتابة حديثاً) بعد بدء المناقلة، لذا يوقف تنفيذ عملية القراءة وتنفذ عملية العودة rollback للمناقلة T_i.

2. If TS(T_i)≥ **W**-timestamp(Q) then

★ تنفذ عملية القراءة، وتُعطى قيمة

R-timestamp(Q) = max (R-timestamp(Q),TS(T_i))

البروتوكولات المرتكزة على الأختام

- بفرض أن المناقلة T_i تنفذ عملية كتابة لعنصر **write**(Q) :

1. If TS(T_i) < **R**-timestamp(Q) then

★ جرى قراءة قيمة سابقة لـ Q بعد بدء تنفيذ المناقلة T_i، وهي تختلف عن القيمة الناتجة من المناقلة T_i والمطلوب تسجيلها، ولذلك يوقف تنفيذ عملية الكتابة وتنفذ عملية العودة rollback للمناقلة T_i.

2. If TS(T_i)< **W**-timestamp(Q) then

★ في هذه الحالة تحاول المناقلة T_i كتابة قيمة مهملة لـ Q ولذلك يوقف تنفيذ عملية الكتابة وتنفذ عملية العودة rollback للمناقلة T_i.

3. Else {Otherwise}

★ تُنفذ عملية الكتابة وتُسند قيمة الختم الخاص بالمناقلة TS(T_i) إلى ختم كتابة العنصر Q أي إلى W-timestamp(Q)

مثال حول استخدام البروتوكول

ليكن لدينا مخطط جزئي من مخطط تنفيذ مجموعة من المناقلات مع أختام بقيم 1،2،3،4،5

R-timestamp(X) = TS(T5)=5
R-timestamp(Y) = TS(T2)=2

W-Timestamp(Y) = TS(T3)= 3
W-Timestamp(Z) = TS(T3)= 3

R-timestamp(Z) = TS(T5)=5

R-timestamp(X) = max (R-timestamp(X),TS(*T1*))= 5

T_1	T_2	T_3	T_4	T_5
	read(*Y*)			read(*X*)
read(*Y*)		write(*Y*)		
		write(*Z*)		
				read(*Z*)
	write(*X*)			
	abort			
read(*X*)				
		write(*Z*)		
		abort		
				write(*Y*)
				write(*Z*)

يجب العودة بشكل شلال للمحافظة على صحة القاعدة

مثال حول استخدام البروتوكول

ليكن لدينا مخطط تنفيذ مع أختام للمناقلتين التاليتين : في هذا المخطط لدينا TS(T1) < TS(T2)

T1	T2
Read(B)	
	Read (B) B := B-50 Write(B)
Read(A)	
	Read(A)
Display(A+B)	
	A := A+50 Write(A) Display(A+B)

R-timestamp(B) = max (R-timestamp(B),TS(*T1*))= TS(T1)

R-timestamp(B) = max (R-timestamp(B),TS(*T2*))= TS(T2)

W-Timestamp(B) = TS(T2)

R-timestamp(A) = max (R-timestamp(A),TS(*T1*))= TS(T1)

R-timestamp(A) = max (R-timestamp(A),TS(*T2*))= TS(T2)

W-Timestamp(A) = TS(T2)

صحة بروتوكول ترتيب الأختام
Correctness of Timestamp-Ordering Protocol

- يؤمن بروتوكول ترتيب الأختام التسلسلية حيث أن جميع الأقواس الموجودة في بيان السوابق هي من الشكل :

أي أنه لن يكون هناك حلقات في بيان السوابق.

- يؤمن بروتوكول المرتكز على الأختام التحرر من مشكلة العرقلة المتبادلة حيث أن المناقلات لا تنتظر إطلاقاً.
- يمكن أن يحوي المخطط الناتج على مشكلة العودة بشكل شلال ، وحتى يمكن أن يكون غير قابل للاستعادة.

الاستعادة والتحرر من مشكلة الشلال

- المشاكل الموجودة في البروتوكول المرتكز على الأختام :
  - ★ بفرض أن المناقلة T_i أنهي عملها، وبنفس الوقت جرت عملية قراءة من قبل المناقلة T_j لمعطيات كُتبت من قبل المناقلة T_i ، في هذه الحالة يجب على المناقلة T_j أن تنهي عملها، وفي حال قامت المناقلة T_j بعملية تسجيل مبكرة (commit) يكون مخطط التنفيذ غير قابل للاستعادة.
  - ★ جميع المناقلات التي قرأت عناصر مكتوبة في المناقلة T_j يجب أن تُنهي عملها (abort rollback) (العودة بشكل شلال)
- الحل :
  - ★ تنظيم المناقلات بحيث تنفذ جميع عمليات التسجيل لديها في آخر المناقلة.
  - ★ تُشكل جميع عمليات التسجيل في المناقلة وحدة كتلية، أي لا يمكن لأية مناقلة أخرى تنفيذ أي عملية خلالها.
  - ★ تبدأ المناقلة المنهية (abort) بختم جديد.

التعامل مع العرقلة المتبادلة
Deadlock Handling

ليكن لدينا المناقلتين التاليتين :

T_1: write (X)
write(Y)

T_2: write(Y)
write(X)

- مخطط تنفيذ يحوي مشكلة العرقلة المتبادلة

T_1	T_2
lock-X on X write (X)	
	lock-X on Y write (Y)
wait for **lock-X** on Y	wait for **lock-X** on X

التعامل مع العرقلة المتبادلة
Deadlock Handling

- نقول أن النظام في حالة عرقلة متبادلة إذا وجدت مجموعة من المناقلات بحيث تنتظر كل مناقلة من المجموعة مناقلة أخرى من المجموعة نفسها.
- تؤمن بروتوكولات منع العرقلة المتبادلة ***Deadlock prevention*** protocols عدم دخول النظام في حالة عرقلة متبادلة.
- نذكر بعض استراتيجيات المنع :
  - ★ تعريف مسبق
    - ➢ أن تقوم كل مناقلة بالحصول على إقفال جميع العناصر التي تتعامل معها قبل البدء بالتنفيذ
  - ★ تفترض وجود ترتيب جزئي على جميع العناصر وطلب أن يكون باستطاعة المناقلة إقفال العناصر بالترتيب المحدد بالترتيب الجزئي (البروتوكول المرتكز على البيان).

استراتيجيات أخرى لمنع العرقلة المتبادلة

★ **استخدام أختام المناقلات لمنع حدوث العرقلة المتبادلة.**

★ **wait-die**

- تنتظر المناقلة الأقدم تحرير مناقلة شابة للعناصر. بينما لا تنتظر المناقلات الشابة أبداً المناقلات الأقدم، وإنما تقوم بعملية rollback بدلاً من الانتظار.
- يمكن أن يؤدي ذلك إلى موت مناقلة die عدة مرات قبل أن تحصل على المعطيات التي تحتاجها.

★ **wound-wait**

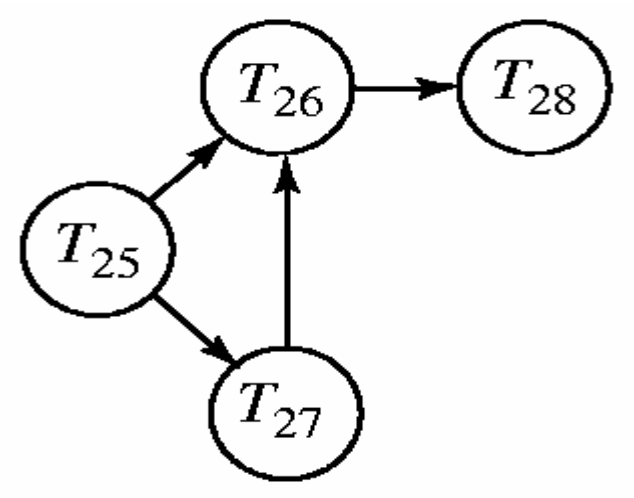
- تُجبر المناقلة الأقدم المناقلة الشابة على التراجع roolback عند حاجتها إلى معطيات مقفولة من قبلها بدلاً من انتظارها. يُسمح للمناقلة الشابة أو الأحدث انتظار المناقلة الأقدم تحرير معطياتها.
- يمكن أن تسبب هذه الاستراتيجية عدد من مرات التراجع rollback أقل من الاستراتيجية السابقة.

★ في الاستراتيجيتين السابقتين يعاد تنفيذ المناقلة بعد عملية التراجع rollback باستخدام الختم البدائي لها، وهذا ما يساعد على تجنب ظاهرة الحرمان.

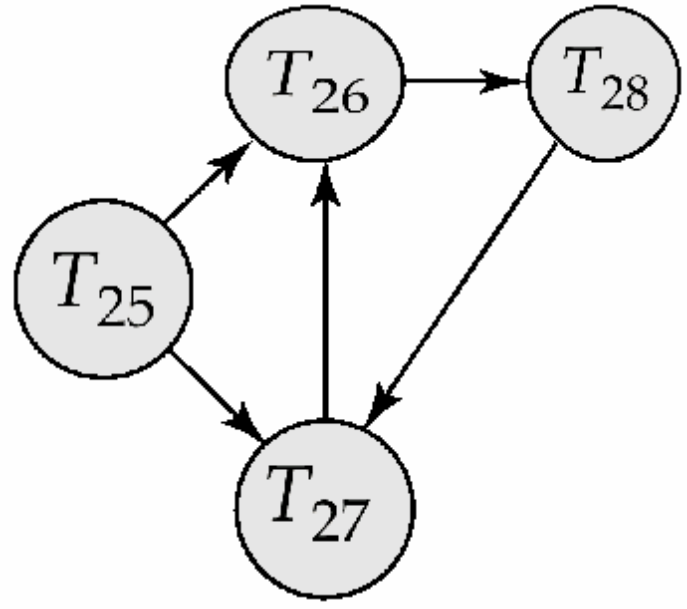
كشف العرقلة المتبادلة

- يمكن وصف العرقلة المتبادلة كعملية "انتظار من أجل" ضمن بيان يتألف من :
  - ★ V مجموعة من العقد الممثلة لجميع المناقلات المتواجدة في النظام
  - ★ E مجموعة من الأقواس التي تربط بين المناقلات ، عناصر هذه المجموعة هي أزواج مرتبة من الشكل $T_i \rightarrow T_j$
  - ★ إذا كان $T_i \rightarrow T_j$ من E فإنه يوجد قوس موجه من T_i إلى T_j والدال على أن المناقلة T_i تنتظر أن تقوم T_j بتحرير العنصر الذي تريد إقفاله.
- عندما تطلب مناقلة T_i التعامل مع عنصر حالياً مُنح لمناقلة T_j يُضاف الزوج المرتب $T_i \rightarrow T_j$ إلى بيان "wait-for" . يحذف هذا القوس فقط عند حصول T_i على العنصر المطلوب.
- يكون النظام في حالة عرقلة متبادلة إذا وفقط إذا احتوى بيان "الانتظار من أجل" حلقة. يجب تنفيذ خوارزمية كشف العرقلة المتبادلة بشكل دوري لكشف الحلقات.

كشف العرقلة المتبادلة



بيان "wait-for" بدون حلقة



بيان "wait-for" مع حلقة

إزالة العرقلة المتبادلة

- عند اكتشاف عرقلة متبادلة :
  - ★ سوف نختار بعض المناقلات لإرجاعها rollback لإزالة العرقلة المتبادلة وبحيث تكون الكلفة أقل ما يمكن.
  - ★ يحدث الحرمان starvation إذا اختيرت نفس المناقلات للإرجاع دوماً لإزالة العرقلة المتبادلة. يمكن إدخال عدد مرات الإرجاع rollback كعامل من عوامل حساب الكلفة لتجنب الحرمان