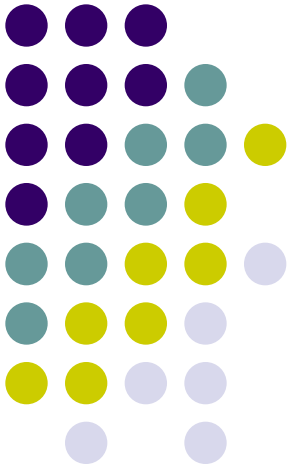
Applied Faculty
Hama University

Database

معالجة المناقلات (٢)

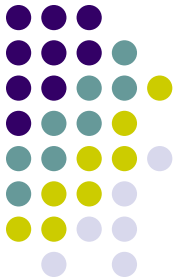
Transaction Processing



أسباب فشل المناقلات

- انهيار النظام (عطل عتادي أو برمجي أو شبكي)
- أخطاء حسابية (قسمة على صفر، . . .)
- استثناءات غير ملتقطة ضمن المناقلة (بيانات غير موجودة، خرق قيد كمال)
- التحكم بالتزامن
  - قد يتم إلغاء مناقلة على أن يعاد تشغيلها لاحقاً (لأنها تخرق قابلية التسلسل)
  - أو لوقوع عدة مناقلات في القفل الميت

جدولة المناقلات (Schedule)



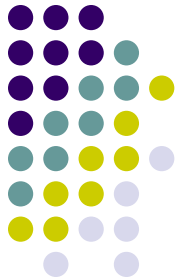
- عند تنفيذ المناقلات بشكل متزامن من خلال تداخل العمليات، نسمي ترتيب تنفيذ العمليات من المناقلات المختلفة بالجدولة (schedule)
- الجدولة S لـ n مناقلة $T_1, T_2, \ldots, T_n$ هو ترتيب لعمليات المناقلات بحيث :
  - من أجل كل مناقلة T_i مساهمة في S فإن عمليات T_i في S يجب أن تظهر في نفس الترتيب الذي تظهر فيه في T_i
  - العمليات من مناقلات أخرى T_j يمكن أن تتداخل مع عمليات T_i في S

جدولة المناقلات (Schedule)

- من وجهة نظر الاستعادة و التحكم بالتزامن ، نهتم بشكل رئيسي بعمليات :
  - القراءة read_item سنرمز لها بـ r
  - الكتابة write_item سنرمز لها بـ w
  - التثبيت commit سنرمز لها بـ c
  - الإلغاء abort سنرمز لها بـ a
- تكتب الجدولة على شكل تتالي من العمليات
  - لكل عملية نضيف دليل هو رقم المناقلة
  - في عمليات القراءة و الكتابة نضع عنصر البيانات X بين قوسين بعد رمز العملية r أو w

$$S_1 : r_1(x); r_2(z); w_1(x); w_2(z);$$

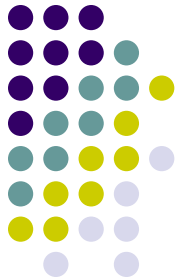
مثال



T_1	T_2
read_item(X);	
X := X - N;	
	read_item(X);
	X := X + M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y := Y + N;	
write_item(Y);	

$$S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$$

مثال



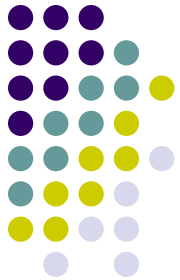
T_1	T_2
read_item(X);	
X := X - N;	
write_item(X);	
	read_item(X);
	X := X + M;
	write_item(X);
read_item(Y);	

$$S_b : r_1(X); w_1(X); r_2(X); w_2(X); r_1(Y); a_1;$$

العمليات المتنازعة

- نقول عن عمليتين في جدولة أنهما متنازعتين (conflict) إذا كانتا تحققان كل الشروط التالية:
  - تنتميان إلى مناقلتين مختلفتين
  - يصلان إلى نفس عنصر البيانات X
  - إحدى العمليتين على الأقل هي عملية كتابة write_item

العمليات المتنازعة



$$S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$$

$$S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$$

$$S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$$

$$S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$$

$$S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$$

$$S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$$

العمليات المتنازعة

متنازعتين $S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$

متنازعتين $S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$

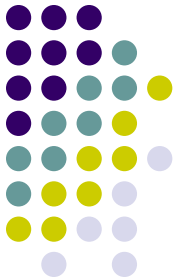
متنازعتين $S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$

غير متنازعتين $S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$ عمليتا قراءة

غير متنازعتين $S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$ عنصري بيانات مختلفين

غير متنازعتين $S_a : r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y);$ نفس المناقلة

تمييز الجدولات بحسب قابلية التسلسل

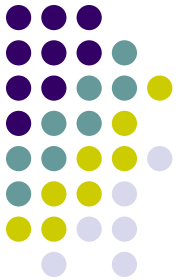


Time ↓

T_1	T_2
read_item(X);	
X := X - N;	
write_item(X);	
read_item(Y);	
Y := Y + N;	
write_item(Y);	
	read_item(X);
	X := X + M;
	write_item(X);

Schedule A

تمييز الجدولات بحسب قابلية التسلسل

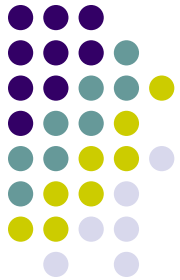


Time ↓

T_1	T_2
	read_item(X);
	X := X + M;
	write_item(X);
read_item(X);	
X := X - N;	
write_item(X);	
read_item(Y);	
Y := Y + N;	
write_item(Y);	

Schedule B

تمييز الجدولات بحسب قابلية التسلسل

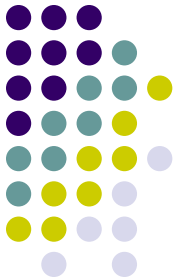


Time ↓

T_1	T_2
read_item(X);	
X := X - N;	
	read_item(X);
	X := X + M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y := Y + N;	
write_item(Y);	

Schedule C

تمييز الجدولات بحسب قابلية التسلسل



Time ↓

T_1	T_2
read_item(X);	
X := X - N;	
write_item(X);	
	read_item(X);
	X := X + M;
	write_item(X);
read_item(Y);	
Y := Y + N;	
write_item(Y);	

Schedule D

الجدولة التسلسلية (serial schedule)

- نقول عن جدولة أنها تسلسلية (serial) إذا كانت عملياتها تنفذ بشكل متتابع، بدون أي تداخل من عمليات المناقلات الأخرى

- في الجدولة التسلسلية تنجز كامل المناقلات بترتيب تسلسلي:
  - T1 ثم T2 في الجدولة A
  - T2 ثم T1 في الجدولة B

- الجدولات C و D تسمى جدولات غير تسلسلية (non-serial) لأنه في كل سلسلة العمليات تتداخل من كلا المناقلتين

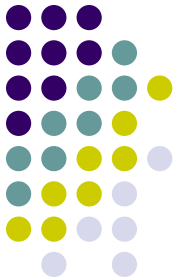
الجدولة التسلسلية (serial schedule)

- في الجدولة التسلسلية هناك مناقلة واحدة فقط تكون نشطة في وقت معين
- تثبيت أو إلغاء المناقلة النشطة يؤدي إلى بدء المناقلة التالية
- ليس هناك تداخل في العمليات في الجدولات التسلسلية
- إذا اعتبرنا أن المناقلات مستقلة عن بعضها، إذاً كل جدولة تسلسلية تعتبر صحيحة
- يمكننا اعتبار هذا لأن كل مناقلة تعتبر صحيحة إذا نفذت لوحدها (وفقاً لخاصية الحفاظ على التجانس)
- إذا ليس من المهم أي المناقلات تنفذ أولاً
- طالما أن كل مناقلة تنفذ من البداية إلى النهاية بدون أي تداخل من عمليات المناقلات الأخرى، فإننا سنحصل على نتيجة نهائية صحيحة في قاعدة البيانات

الجدولة التسلسلية (serial schedule)

- مشكلة الجدولات التسلسلية هي أنها تحد من التزامن و تداخل العمليات
- في الجدولة التسلسلية، إذا كانت مناقلة تنتظر انتهاء عملية I/O لا يمكننا تحويل عمل المعالج إلى مناقلة أخرى
  - و بالتالي هدر زمن المعالجة الثمينٍ للمعالج
- إذا كانت مناقلة T طويلة نسبيا فهذا يعني أنه على المناقلات الأخرى الانتظار طويلاً حتى تبدأ
- لذا تعتبر الجدولات التسلسلية غير مقبولة من الناحية العملية

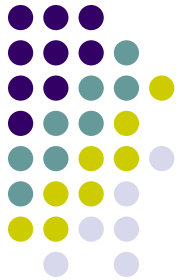
قابلية التسلسل (Serializability)



- نقول عن جدولة S من n مناقلة أنها قابلة للتسلسل إذا كانت مكافئة لجدولة تسلسلية ما من نفس الـ n مناقلة

- قولنا أن جدولة غير تسلسلية S قابلة للتسلسل مكافئ لقولنا أنها صحيحة لأنها مكافئة لجدولة تسلسلية و التي تعتبر صحيحة

- السؤال المتبقي هو: متى نعتبر جدولتين متكافئتين؟
  - التكافؤ بحسب النتيجة
  - التكافؤ بحسب النزاع



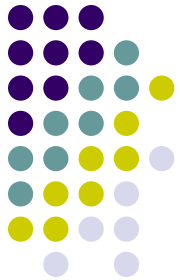
التكافؤ بحسب النتيجة

- نقول عن جدولتين أنهما متكافئتان بحسب النتيجة إذا كانتا تنتجان نفس الحالة النهائية لقاعدة البيانات
- أحياناً، جدولتان مختلفتان قد تنتجان نفس الحالة النهائية بالصدفة

X = 100

S_1	S_2
read_item(X);	read_item(X);
X := X + 10;	X := X * 1.1;
write_item(X);	write_item(X);

- لذا لا يمكن استخدام تكافؤ النتيجة لوحده لتعريف تكافؤ الجدولات
- الطريقة الآمن و الأعم لتعريف تكافؤ الجدولات هي أن لا نقوم بأي افتراض عن أنواع العمليات المتضمنة في المناقلات



التكافؤ بحسب النزاع

- نقول عن جدولتين أنهما متكافئتان بحسب النزاع إذا كان ترتيب أي عمليتين متنازعتين هو نفسه في كلا الجدولتين
- نقول عن عمليتين أنهما متنازعتين إذا كانتا:
  - تنتميان إلى مناقلتين مختلفتين
  - تصلان إلى نفس عنصر البيانات X
  - إحدى العمليتين على الأقل هي عملية كتابة write_item

التكافؤ بحسب النزاع

- إذا تم تطبيق عمليتين متنازعتين بترتيب مختلف في جدولتين فإن التأثير سيكون مختلف على قاعدة البيانات أو على مناقلات أخرى في الجدولة
- و بالتالي الجدولتين غير متكافئتين

$$S_1: \quad r_1(X); \quad \cdots \quad ; w_2(X);$$
$$S_2: \quad w_2(X); \quad \cdots \quad ; r_1(X);$$

القيمة المقروءة بالعملية $r_1(X)$ يمكن أن تكون مختلفة في الجدولتين

$$S_1: \quad w_1(X); \quad \cdots \quad ; w_2(X);$$
$$S_2: \quad w_2(X); \quad \cdots \quad ; w_1(X);$$

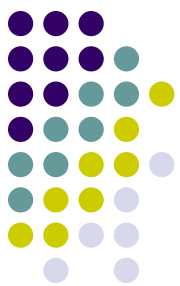
عملية $r(X)$ التالية في كلا الجدولتين ستقرأ قيمتين مختلفتين
أو إذا كانت هاتين هما العمليتان الأخيرتان اللتان تكتبان X في الجدولتين
فإن القيمة النهائية للعنصر X في قاعدة البيانات ستكون مختلفة

قابلية التسلسل بحسب النزاع
Conflict Serializability

- نقول عن جدولة S أنها قابلة للتسلسل بحسب النزاع إذا كانت مكافئة بحسب النزاع لجدولة تسلسلية ما S′

- في هذه الحالة يمكننا إعادة ترتيب العمليات غير المتنازعة في S حتى نحصل على الجدولة التسلسلية المكافئة S′

قابلية التسلسل بحسب النزاع
Conflict Serializability



Time ↓

T_1	T_2
read_item(X);	
X := X - N;	
write_item(X);	
read_item(Y);	
Y := Y + N;	
write_item(Y);	
	read_item(X);
	X := X + M;
	write_item(X);

Schedule A

(serial)

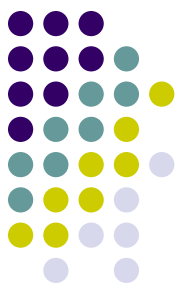
≡

T_1	T_2
read_item(X);	
X := X - N;	
write_item(X);	
	read_item(X);
	X := X + M;
	write_item(X);
read_item(Y);	
Y := Y + N;	
write_item(Y);	

Schedule D

(serializable)

قابلية التسلسل بحسب النزاع
Conflict Serializability



Time

T_1	T_2
read_item(X);	
X := X - N;	
write_item(X);	
read_item(Y);	
Y := Y + N;	
write_item(Y);	
	read_item(X);
	X := X + M;
	write_item(X);

Schedule A

(serial)

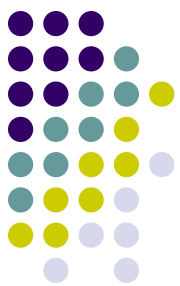
≠

T_1	T_2
read_item(X);	
X := X - N;	
	read_item(X);
	X := X + M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y := Y + N;	
write_item(Y);	

Schedule C

(not serializable)

قابلية التسلسل بحسب النزاع
Conflict Serializability



Time

T_1	T_2
	read_item(X);
	X := X + M;
	write_item(X);
read_item(X);	
X := X - N;	
write_item(X);	
read_item(Y);	
Y := Y + N;	
write_item(Y);	

Schedule B

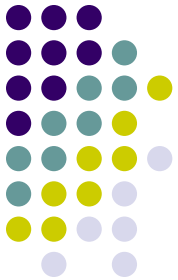
(serial)

≠

T_1	T_2
read_item(X);	
X := X - N;	
	read_item(X);
	X := X + M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y := Y + N;	
write_item(Y);	

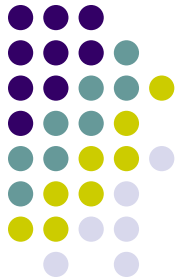
Schedule C

(not serializable)



اختبار قابلية التسلسل بحسب النزاع لجدولة

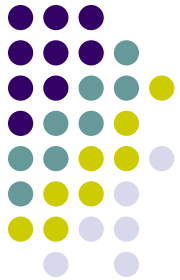
- هناك خوارزمية بسيطة لتحديد قابلية التسلسل بحسب النزاع لجدولة
- هذه الخوارزمية تنظر فقط إلى عمليات read_item و write_item في الجدولة من أجل بناء بيان الأسبقية (precedence graph) أو بيان التسلسل
- و هو بيان موجه $G=(N,E)$ يتكون من:
  - مجموعة من العقد $N = \{T_1, T_2, \dots , T_n\}$
  - مجموعة من الأضلاع الموجهة $E =\{e_1,e_2, \dots , e_m\}$
- هناك عقدة واحدة في البيان من أجل كل مناقلة T_i في الجدولة
- كل ضلع e_i في البيان له الشكل $T_j \rightarrow T_k$
  - ننشئ مثل هذا الضلع إذا كانت إحدى العمليات في T_j تظهر في الجدولة قبل عملية ما من T_k متنازعة معها



خوارزمية اختبار قابلية التسلسل

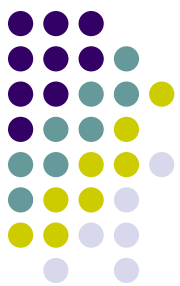
- ننظر فقط إلى عمليات read_item(X) و write_item(X)
- ننشئ بيان الأسبقية (بيان التسلسل)
  - من أجل كل مناقلة T_i في الجدولة ننشئ عقدة في بيان الأسبقية
  - ننشئ ضلع من T_i إلى T_j إذا كانت إحدى العمليات في T_i تظهر قبل عملية متنازعة معها في T_j
- تكون الجدولة قابلة للتسلسل إذا و فقط إذا كان بيان الأسبقية لا يحوي حلقات

خوارزمية اختبار قابلية التسلسل



1. For each transaction T_i participating in schedule S,
   create a node labeled T_i in the precedence graph.
2. For each case in S where T_j executes a read_item(X) after T_i executes a write_item(X),
   create an edge ($T_i \rightarrow T_j$) in the precedence graph.
3. For each case in S where T_j executes a write_item(X) after T_i executes a read_item(X),
   create an edge ($T_i \rightarrow T_j$) in the precedence graph.
4. For each case in S where T_j executes a write_item(X) after T_i executes a write_item(X),
   create an edge ($T_i \rightarrow T_j$) in the precedence graph.
5. The schedule S is serializable if and only if the precedence graph has no cycles.

مثال

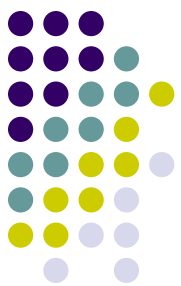


Time

T_1	T_2
read_item(X);	
X := X - N;	
write_item(X);	
read_item(Y);	
Y := Y + N;	
write_item(Y);	
	read_item(X);
	X := X + M;
	write_item(X);

Schedule A

مثال



Time

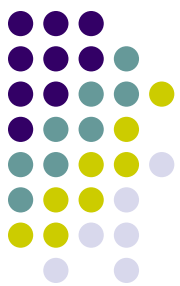
T_1	T_2

	read_item(X);
	X := X + M;
	write_item(X);

read_item(X);
X := X - N;
write_item(X);
read_item(Y);
Y := Y + N;
write_item(Y);	

Schedule B

مثال

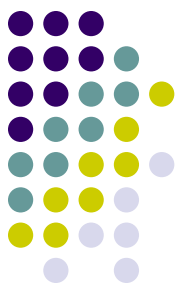


Time

T_1	T_2
read_item(X);	
X := X - N;	
	read_item(X);
	X := X + M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y := Y + N;	
write_item(Y);	

Schedule C

مثال



Time

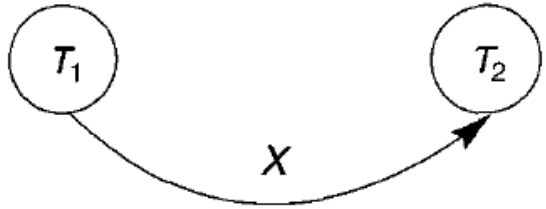
T_1	T_2
read_item(X);	
X := X - N;	
write_item(X);	
	read_item(X);
	X := X + M;
	write_item(X);
read_item(Y);	
Y := Y + N;	
write_item(Y);	

Schedule D

مثال

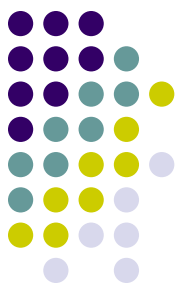
T_1	T_2
read_item(X);	
X := X - N;	
write_item(X);	
read_item(Y);	
Y := Y + N;	
write_item(Y);	
	read_item(X);
	X := X + M;
	write_item(X);

Time



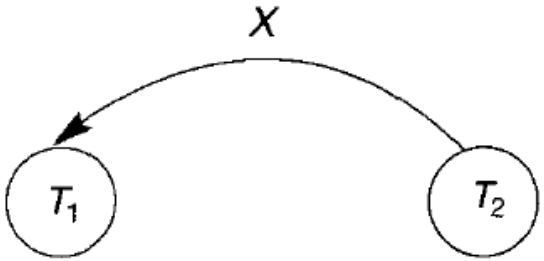
Schedule A

مثال



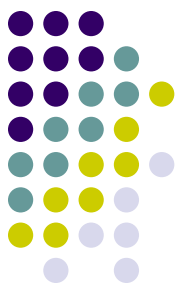
T_1	T_2
	read_item(X);
	X := X + M;
	write_item(X);
read_item(X);	
X := X - N;	
write_item(X);	
read_item(Y);	
Y := Y + N;	
write_item(Y);	

Time



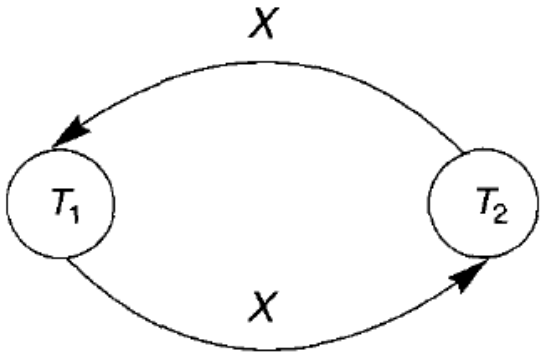
Schedule B

مثال



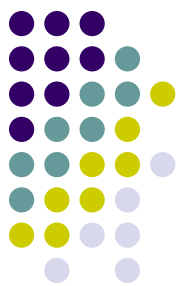
Time

T_1	T_2
read_item(X);	
X := X - N;	
	read_item(X);
	X := X + M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y := Y + N;	
write_item(Y);	



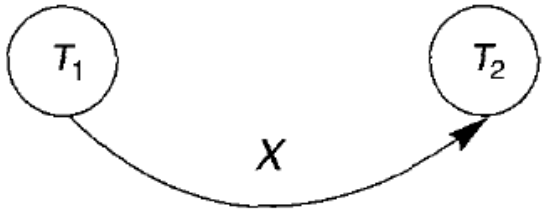
Schedule C

مثال



T_1	T_2
read_item(X);	
X := X - N;	
write_item(X);	
	read_item(X);
	X := X + M;
	write_item(X);
read_item(Y);	
Y := Y + N;	
write_item(Y);	

Time



Schedule D