CÁC HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

CHƯƠNG 3

ĐIỀU KHIỂN GIAO DỊCH ĐỒNG THỜI

Giảng viên: ThS. Nguyễn Thị Uyên Nhi
Email: uyennhisgu@gmail.com

KHOA CÔNG NGHỆ THÔNG TIN

Nội dung

- Kĩ thuật khóa.
  - Khóa nhị phân
  - Khóa đọc/ghi
- Khóa 2 pha.
- Deadlock và Starvation.
  - Deadlock Prevention.
  - Deadlock Detection.
- Kĩ thuật nhãn thời gian.
- Kĩ thuật sử dụng nhiều phiên bản.

GIỚI THIỆU

- Tìm hiểu một số kĩ thuật điều khiển song hành (Concurrency control) được sử dụng trong việc đảm bảo tính cô lập của các giao dịch được thực hiện.
- Các kĩ thuật này đảm bảo tính khả tuần tự của lịch trình dựa trên các **giao thức điều khiển song hành – Concurrency control protocols** (protocols – sets of rules)

GIAO THỨC DỰA TRÊN KHÓA

- Một phương pháp để đảm bảo tính tuần tự dựa trên khái niệm khóa (LOCKING) các hạng mục dữ liệu
- Kĩ thuật khóa ngăn chặn nhiều giao dịch truy xuất 1 hạng mục dữ liệu trong cùng 1 thời điểm.
- Cơ chế khóa được sử dụng trong hầu hết các hệ quản trị CSDL thương mại.
- Yêu cầu việc truy xuất đến một hạng mục dữ liệu được tiến hành theo kiểu loại trừ lẫn nhau (mutual exclusion).
- Một giao dịch đang truy xuất 1 hạng mục dữ liệu thì **không** cho phép giao dịch khác chỉnh sửa dữ liệu này.

GIAO THỨC DỰA TRÊN KHÓA

- Một khóa (lock) là một biến tương ứng với một hạng mục dữ liệu, quy định những hành động cụ thể nào được phép thực hiện trên hạng mục dữ liệu đó.
- Thông thường: 1 khóa cho mỗi hạng mục dữ liệu.
- Có nhiều loại khóa được sử dụng trong điều khiển song hành.

KHÓA NHỊ PHÂN

- Đơn giản nhưng rất hạn chế nên không dùng trong thực tế
- 1 khóa nhị phân (binary lock) gồm 2 trạng thái:
  - Locked (1)
  - Unlocked (0)
- Các khóa khác nhau trên mỗi hạng mục dữ liệu khác nhau.
- Nếu trạng thái khóa của X là 1, hạng mục dữ liệu X không thể được truy xuất bởi các thao tác dữ liệu khác.
  - Lock(X) = 1

KHÓA NHỊ PHÂN

- 2 thao tác trong khóa nhị phân:
  - Lock_item(X)
  - Unlock_item(X)

- Khi một giao dịch muốn truy xuất X, trước tiên nó thực hiện một thao tác *Lock_item(X)*.
  - Nếu Lock(X) = 1, giao dịch phải đợi.
  - Nếu Lock(X) =0, giá trị Lock(X) gán thành 1, giao dịch được thao tác trên X.

KHÓA NHỊ PHÂN

- Sau khi hoàn tất những thao tác trên X, giao dịch thực hiện thao tác Unlock_item(X):
  - Gán Lock(X)=0
- Khi đó, X có thể được truy xuất bởi các giao dịch khác.
- Giai đoạn giữa Lock_item(X) và Unlock_item(X), giao dịch được gọi là đang giữ khóa trên X.
- Chỉ một giao dịch được giữ khóa trên 1 hạng mục dữ liệu.

KHÓA NHỊ PHÂN

- Thao tác khóa/mở khóa:

```
lock_item(X):
B:   if LOCK(X) = 0                (* item is unlocked *)
         then LOCK(X) ←1           (* lock the item *)
     else
         begin
         wait (until LOCK(X) = 0
             and the lock manager wakes up the transaction);
         go to B
         end;
unlock_item(X):
     LOCK(X) ← 0;                  (* unlock the item *)
     if any transactions are waiting
         then wakeup one of the waiting transactions;
```

KHÓA NHỊ PHÂN

- Khóa nhị phân được thực hiện đơn giản bằng cách thêm 1 biến nhị phân ứng với mỗi hạng mục dữ liệu.
- Mỗi khóa được ghi nhận với 3 thuộc tính cơ bản: <Data_item_name, LOCK, Locking_transaction>
  - Tên hạng mục dữ liệu.
  - Giá trị khóa LOCK
  - Giao dịch giữ khóa
- Hệ thống quản lý các khóa trong một bảng, các hạng mục dữ liệu không có trong bảng là không khóa (unlocked).

KHÓA NHỊ PHÂN

Khi đó mỗi giao dịch phải tuân theo các luật:

1. Một giao dịch phải thực hiện thao tác Lock_item(X) trước khi thực hiện bất kì thao tác đọc/ghi nào trên X.
2. Một giao dịch phải Unlock_item(X) sau khi hoàn tất tất cả thao tác đọc/ghi trên X.
3. Một giao dịch không thực hiện thao tác Lock_item(X) nếu nó đang giữ khóa trên X.
4. Một giao dịch không thực hiện thao tác Unlock_item(X) trừ khi nó đang giữ khóa trên X.

KHÓA SHARED/EXCLUSIVE

- Khóa nhị phân hạn chế khi dùng cho dữ liệu vì chỉ một giao dịch được giữ khóa trên một hạng mục dữ liệu X.
- Các thao tác đọc cùng hạng mục dữ liệu trên các giao dịch khác nhau là không xung đột ➔ Ta có thể cho phép nhiều giao dịch được truy xuất X nếu các giao dịch đó chỉ đọc X.
- Nếu một giao dịch thực hiện việc ghi lên X, nó được cấp quyền truy xuất riêng (exclusive).
- Shared locks hay còn gọi là read locks: các giao dịch khác được phép đọc.
- Exclusive locks hay còn gọi là write locks: chỉ 1 giao dịch giữ khóa được thao tác.

KHÓA SHARED/EXCLUSIVE

- Có 3 thao tác trong giao thức khóa này:
  - Read_lock(X)
  - Write_lock(X)
  - Unlock(X)
- Tương tự, giá trị khóa trên X – Lock(X) có 3 trạng thái:
  - Read-locked (Shared-lock)
  - Write-locked (Exclusive-lock)
  - Unlocked

KHÓA SHARED/EXCLUSIVE

- Một phương pháp thực thi giao thức khóa này là lưu các giao dịch giữ khóa trong một bảng.
- Mỗi dòng trong bảng có 4 thông tin:
  - Tên hạng mục dữ liệu
  - Giá trị của khóa: read-locked/write-locked
  - Số lượng giao dịch đọc
  - Các giao dịch giữ khóa

KHÓA SHARED/EXCLUSIVE

Hệ thống phải tuân theo các luật:

1. Một giao dịch T phải thực hiện thao tác Read_lock(X) hoặc Write_lock(X) trước khi đọc X.

2. T phải thực hiện thao tác Write_lock(X) trước khi ghi X.

3. T phải thực hiện thao tác Unlock(X) sau khi hoàn tất việc đọc, ghi trong T.

KHÓA SHARED/EXCLUSIVE

4. T không thực hiện Read_lock(X) nếu nó đang giữ khóa read hoặc write trên X.

5. T không thực hiện Write_lock(X) nếu nó đang giữ khóa read hoặc write trên X.

6. T không thực hiện Unlock(X) trừ khi nó đang giữ khóa read hoặc write trên X.

THAO TÁC KHÓA ĐỌC

```
read_lock(X):
B:   if LOCK(X) = "unlocked"
        then begin LOCK(X) ← "read-locked";
                   no_of_reads(X) ← 1
                   end
     else if LOCK(X) = "read-locked"
        then no_of_reads(X) ← no_of_reads(X) + 1
     else begin
             wait (until LOCK(X) = "unlocked"
                   and the lock manager wakes up the transaction);
             go to B
             end;
```

THAO TÁC KHÓA GHI

```
write_lock(X):
B:   if LOCK(X) = "unlocked"
          then LOCK(X) ← "write-locked"
     else begin
               wait (until LOCK(X) = "unlocked"
                    and the lock manager wakes up the transaction);
               go to B
               end;
```

THAO TÁC MỞ KHÓA

```
unlock (X):
    if LOCK(X) = "write-locked"
        then begin LOCK(X) ← "unlocked";
                wakeup one of the waiting transactions, if any
                end
    else it LOCK(X) = "read-locked"
        then begin
                no_of_reads(X) ← no_of_reads(X) –1;
                if no_of_reads(X) = 0
                    then begin LOCK(X) = "unlocked";
                            wakeup one of the waiting transactions, if any
                            end
                end;
```

KHÓA SHARED/EXCLUSIVE

Chuyển đổi khóa (Lock conversion):

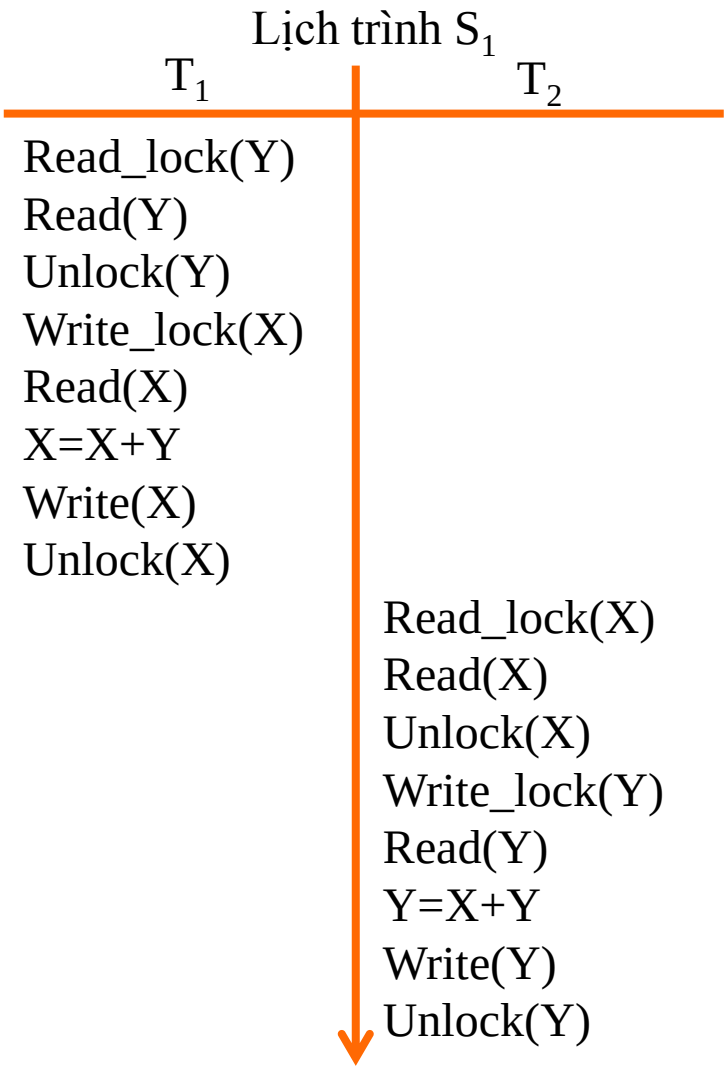
- Nâng cấp (Upgrade) khóa: Nếu giao dịch T là giao dịch **duy nhất** đang giữ khóa read trên X tại thời điểm thực hiện thao tác Write_lock(X), khóa read này sẽ được nâng cấp thành khóa Write, nếu không giao dịch phải đợi.
- Hạ cấp (Downgrade) khóa: từ khóa Write chuyển thành khóa read.
- Khi đó bảng lưu thông tin về khóa phải thay đổi giá trị tương ứng.

KĨ THUẬT KHÓA 2 PHA

- Về mặt lý thuyết ta thấy rằng kĩ thuật khóa sẽ đảm bảo được tính khả tuần tự.

- Tuy nhiên trên thực tế, khi thực hiện đúng luật của cơ chế khóa vẫn không đảm bảo hoàn toàn tính khả tuần tự khi điều khiển song hành.

Kĩ thuật khóa 2 pha



- X=20, Y=30
- T1→T2: X=50, Y=80
- T2→T1: X=70, Y=50

KĨ THUẬT KHÓA 2 PHA

Lịch trình S_2

T_1	T_2
Read_lock(Y)	
Read(Y)	
Unlock(Y)	
	Read_lock(X)
	Read(X)
	Unlock(X)
	Write_lock(Y)
	Read(Y)
	Y=X+Y
	Write(Y)
	Unlock(Y)
Write_lock(X)	
Read(X)	
X=X+Y	
Write(X)	
Unlock(X)	

- X=???
- Y=???
- Có đảm bảo tính nhất quán của CSDL?
- Khả tuần tự khi sử dụng cơ chế lock?

KĨ THUẬT KHÓA 2 PHA

- Nguyên nhân: Y trong T_1 và X trong T_2 được unlock quá sớm.
- ➔**Thời điểm mở khóa** là một yếu tố quan trọng.
- Do đó để đảm bảo tính khả tuần tự phải áp dụng thêm một số giao thức khác liên quan đến thời điểm khóa và mở khóa cho mỗi giao dịch.
- Giao thức phổ biến nhất là khóa 2 pha (Two-phase locking)

KĨ THUẬT KHÓA 2 PHA

T_1	T_2

read_lock(Y);
read_item(Y);
unlock(Y);
write_lock(X);
read_item(X);
$X := X + Y$;
write_item(X);
unlock(X);
	read_lock(X);
	read_item(X);
	unlock(X);
	write_lock(Y);
	read_item(Y);
	$Y := X + Y$;
	write_item(Y);
	unlock(Y);

T_1'	T_2'

read_lock(Y);
read_item(Y);
write_lock(X);
unlock(Y)
read_item(X);
$X := X + Y$;
write_item(X);
unlock(X);
	read_lock(X);
	read_item(X);
	write_lock(Y);
	unlock(X)
	read_item(Y);
	$Y := X + Y$;
	write_item(Y);
	unlock(Y);

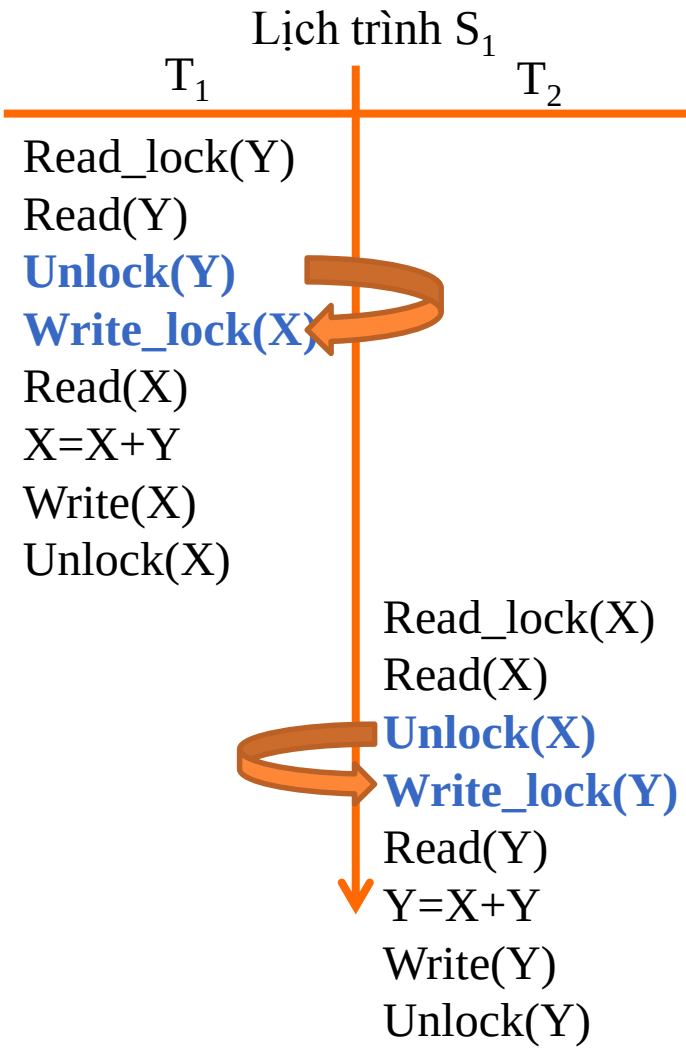
Kĩ thuật khóa 2 pha

- Tất cả thao tác khóa phải diễn ra trước thao tác mở khóa đầu tiên trong giao dịch.
- Khi đó một giao dịch được chia làm 2 pha (giai đoạn):
  - Giai đoạn đầu tiên - tạo khóa (expanding/growing phase): các khóa được phép tạo nhưng không khóa nào được phép mở.
  - Giai đoạn mở khóa (shrinking phase): các khóa đang tồn tại được phép mở nhưng không khóa mới nào được tạo.

KĨ THUẬT KHÓA 2 PHA

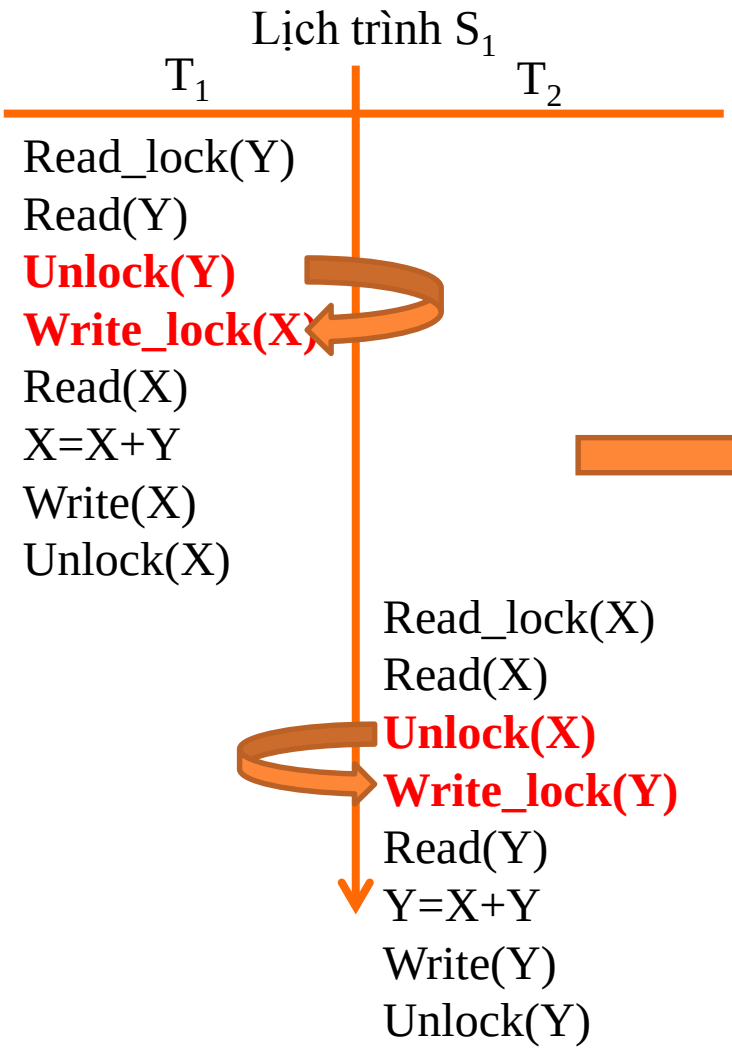
- Trong trường hợp cho phép chuyển đổi khóa:
  - Nâng cấp khóa (Read-locked →Write-locked): thực hiện ở giai đoạn tạo khóa.
  - Hạ cấp khóa: giai đoạn mở khóa.

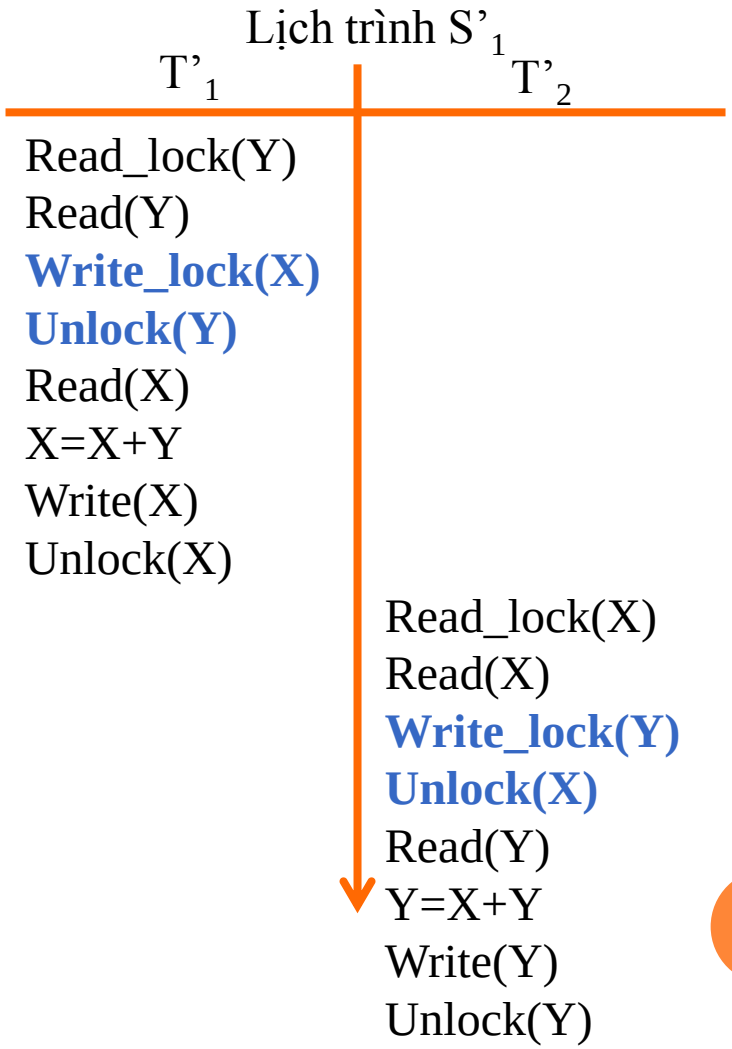
KĨ THUẬT KHÓA 2 PHA



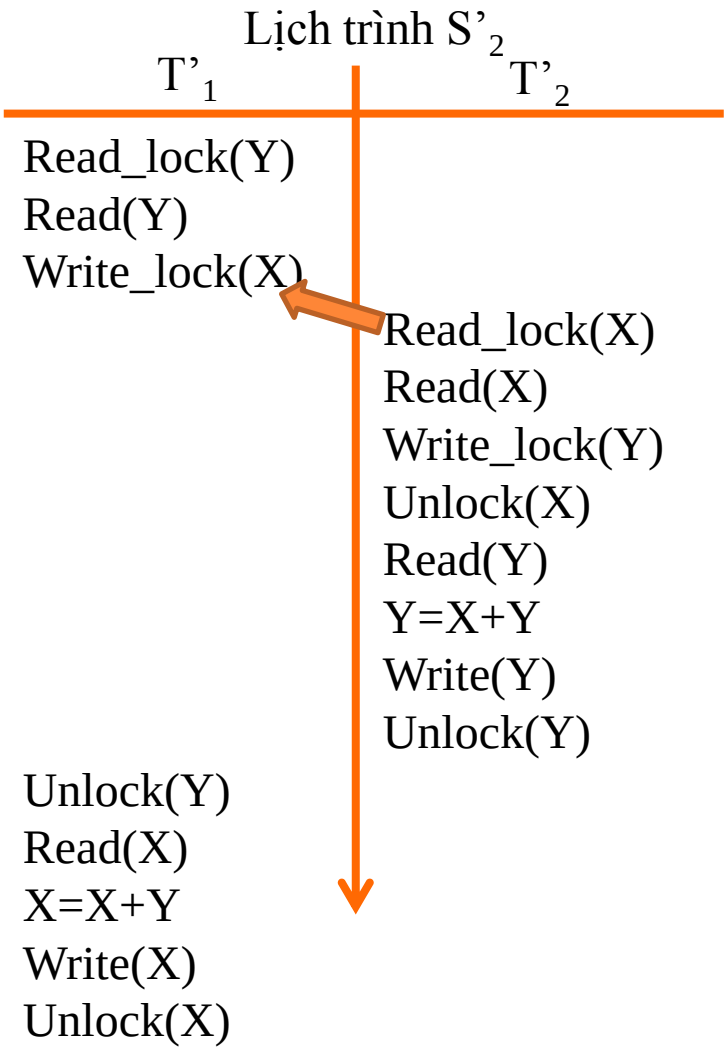
- T1 và T2 không theo giao thức khóa 2 pha.

KĨ THUẬT KHÓA 2 PHA





KĨ THUẬT KHÓA 2 PHA



- Read_lock(X) trong T'2 buộc phải đợi cho đến khi T'1 Unlock(X).
- Đảm bảo được tính khả tuần tự.

MỘT SỐ HẠN CHẾ

- Khóa 2 pha có thể giới hạn số lượng giao dịch đồng thời.
- Giao dịch T không thể mở khóa X sau khi X được sử dụng nếu sau đó T phải khóa Y
- ➔Các giao dịch khác sử dụng đến X buộc phải đợi mặc dù có thể T đã hoàn tất thao tác trên X.

Read_lock(X)
Read(X)
Write_lock(Y)
Unlock(X)
Read(Y)
Y=X+Y
Write(Y)
Unlock(Y)

MỘT SỐ HẠN CHẾ

- Hoặc là T phải khóa thêm Y mặc dù T chưa sử dụng đến Y để mở được khóa X.
- ➔ Các giao dịch khác buộc phải đợi khi thao tác trên Y mặc dù Y chưa sử dụng đến.

```
Read_lock(X)
Write_lock(Y)
Read(X)
Unlock(X)
Read(Y)
Y=X+Y
Write(Y)
Unlock(Y)
```

- **Đó là cái giá của việc đảm bảo tính khả tuần tự mà không cần kiểm tra mỗi lịch trình.**

MỘT SỐ HẠN CHẾ

- Khóa 2 pha mặc dù đảm bảo được tính khả tuần tự nhưng nó không cho phép tất cả các lịch trình khả tuần tự do một số lịch trình bị hạn chế bởi giao thức.
- Ngoài ra còn xuất hiện 2 vấn đề:
  - Deadlock
  - Starvation

MỘT SỐ BIẾN THỂ CỦA KHÓA 2 PHA

- Kĩ thuật mô tả trên: Basic 2PL (2 phase locking).
- Conservative (thận trọng, bảo thủ) 2PL: giao dịch khóa toàn bộ dữ liệu cần thiết trước khi giao dịch bắt đầu thực thi (read_set và write_set).
  Nếu có bất kì dữ liệu nào không thể khóa, giao dịch chờ cho đến khi tất cả dữ liệu được mở khóa → không thực tế.

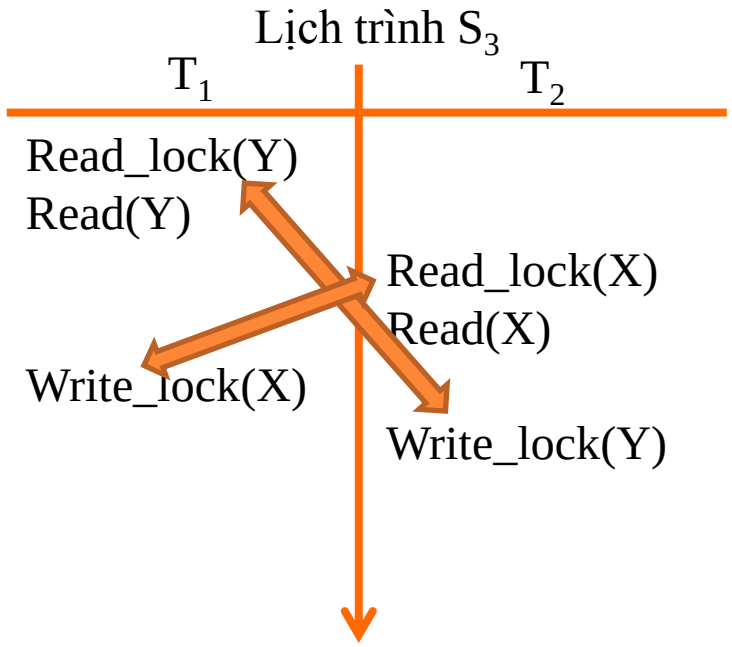
Một số biến thể của khóa 2 pha

- Strict (chặt chẽ) 2PL: giao dịch không mở khóa bất kì khóa exclusive nào cho đến khi giao dịch kết thúc.

- Rigorous (khắt khe) 2PL: giao dịch không mở bất kì khóa nào cho đến khi giao dịch kết thúc.

DEADLOCK

- Deadlock xảy ra khi thực hiện 2 hay nhiều giao dịch đồng thời, khi một giao dịch T chờ đợi một dữ liệu bị khóa bởi giao dịch T’ và ngược lại.



Deadlock Prevention Protocols

- Conservation 2PL: khóa tất cả hạng mục cần thiết trước khi thực hiện giao dịch.
- Một số kĩ thuật khác được đưa ra dựa trên việc quyết định cách thức xử lý giao dịch liên quan đến deadlock: đợi/hủy bỏ/ưu tiên, hủy giao dịch khác
- Một số trong các kĩ thuật này sử dụng khái niệm nhãn thời gian của giao dịch (Transaction timestamp)

Deadlock Prevention Protocols

- Nhãn thời gian: định danh duy nhất gán cho mỗi giao dịch dựa trên thứ tự bắt đầu của giao dịch.
  - Giao dịch bắt đầu trước (cũ hơn) sẽ có nhãn thời gian nhỏ hơn.
  - T_1 bắt đầu trước T_2: $TS(T_1)<TS(T_2)$
- Giả sử **giao dịch T_i muốn khóa một dữ liệu X đã bị khóa bởi một giao dịch T_j khác**. Khi đó có 2 luật để ngăn chặn deadlock:
  - Wait-die
  - Wound-wait

Deadlock Prevention Protocols

Wait-die

- Nếu TS(Ti)<TS(Tj):
  - Ti được phép đợi
- Ngược lại:
  - hủy Ti
  - khởi động lại Ti với TS(Ti) như cũ.

Wound-wait

- Nếu TS(Ti)<TS(Tj):
  - hủy Tj
  - khởi động lại Tj với TS(Tj) như cũ.
- Ngược lại:
  - Ti được phép đợi

Deadlock Prevention Protocols

- Cả 2 cùng kết thúc với việc hủy bỏ giao dịch mới hơn mà **có thể** dẫn đến deadlock.
- Cả 2 kĩ thuật này đảm bảo không xuất hiện deadlock (deadlock-free) do:
  - Wait-die: chỉ đợi giao dịch bắt đầu sau (younger)
  - Wound-die: chỉ đợi giao dịch bắt đầu trước (older)
- Tuy nhiên cả 2 kĩ thuật có thể dẫn đến tình trạng hủy bỏ và khởi động lại giao dịch một cách không cần thiết trong trường hợp không xuất hiện deadlock thật sự.

Deadlock Prevention Protocols

Một số luật khác để chống deadlock không sử dụng nhãn thời gian:

- No waiting algorithm: nếu 1 giao dịch không thể khóa, nó được hủy bỏ và khởi động lại sau 1 khoảng thời gian định trước mà không cần kiểm tra deadlock.
- Cautious waiting algorithm: nếu Tj không đợi một dữ liệu bị khóa nào khác thì Ti được phép đợi; ngược lại hủy bỏ Ti.

Deadlock Detection

Wait-for Graph:

- Node: là các giao dịch đang thực hiện.
- Khi Ti đợi để khóa X đang được khóa bởi Tj: tạo cung Ti→Tj.
- Khi Tj mở khóa X đó, xóa cung Ti→Tj.
- Deadlock: đồ thị có chu trình.
- Lựa chọn giao dịch để hủy (victim selection): tránh các giao dịch được bắt đầu lâu, thực hiện nhiều cập nhật mà chọn giao dịch mới bắt đầu và chưa thực hiện nhiều thay đổi.

TIMEOUTS

- Một cách thức đơn giản để xử lý deadlock khác là kĩ thuật sử dụng thời gian chờ (Timeout)
- Khả thi bởi sự đơn giản.
- Timeouts: nếu một giao dịch thực hiện việc đợi trong thời gian lớn hơn một khoảng thời gian timeouts định sẵn, hệ thống xem như giao dịch gặp deadlock và hủy bỏ nó mà không cần biết deadlock có thực sự xảy ra hay không.

STARVATION

- Khi 1 giao dịch không thể tiếp tục trong một thời gian quá lâu trong khi giao dịch khác vẫn thực hiện bình thường.
- Xảy ra do cơ chế đợi hoặc do cơ chế chọn giao dịch hủy (victim selection) không công bằng.
- Một số giải pháp:
  - First come first served
  - Chấp nhận độ ưu tiên cho các giao dịch nhưng tăng dần độ ưu tiên cho các giao dịch đợi lâu

KĨ THUẬT NHÃN THỜI GIAN

- Kĩ thuật khóa: lịch trình tương đương với 1 vài lịch trình tuần tự (hợp với giao thức khóa)
- Kĩ thuật nhãn thời gian: lịch trình tương đương với 1 lịch trình tuần tự cụ thể được xác định dựa trên nhãn thời gian.
- Điều khiển song hành dựa trên kĩ thuật sắp xếp nhãn thời gian không sử dụng khóa do đó không phát sinh deadlock.
- Nhãn thời gian có thể phát sinh bằng nhiều cách:
  - Thứ tự tăng dần: 1, 2, 3… Tuy nhiên phải reset bộ đếm khi không có giao dịch thực hiện trong một khoảng thời gian.
  - Gán nhãn theo ngày giờ hệ thống.
- Về cơ bản, kĩ thuật này đảm bảo thứ tự truy xuất mỗi hạng mục dữ liệu bởi các chỉ thị xung đột trong lịch trình không vi phạm nhãn thời gian.

KĨ THUẬT NHÃN THỜI GIAN

- Để thực hiện điều này, giải thuật gắn với mỗi hạng mục dữ liệu X 2 giá trị nhãn thời gian:
  - Read_TS(X): nhãn thời gian lớn nhất trong số các nhãn thời gian của các giao dịch đọc thành công X. Read_TS(X)=TS(T) với T là giao dịch mới nhất (youngest) đọc thành công X.
  - Write_TS(X): nhãn thời gian lớn nhất trong số các nhãn thời gian của các giao dịch ghi thành công X. Write_TS(X)=TS(T) với T là giao dịch mới nhất (youngest) ghi thành công X.
- Có nhiều kĩ thuật nhãn thời gian khác nhau.

THỨ TỰ NHÃN THỜI GIAN CƠ BẢN

- Khi giao dịch T thực hiện thao tác Read(X) hoặc Write(X), giải thuật so sánh nhãn thời gian của T với Read_TS(X) hoặc Write_TS(X) để đảm bảo thứ tự thực hiện không bị vi phạm.
- Nếu thứ tự bị vi phạm, giao dịch T phải hủy và gọi lại với một nhãn thời gian mới.
- Trong tình huống T hủy bỏ và có rollback dữ liệu → giao dịch T1 nếu có sử dụng giá trị được ghi bởi T phải rollback theo.
- Tương tự nếu T2 sử dụng giá trị ghi bởi T1 cũng rollback theo.
- → Cascading rollback là một vấn đề và cần có một số giao thức đi kèm

THỨ TỰ NHÃN THỜI GIAN CƠ BẢN

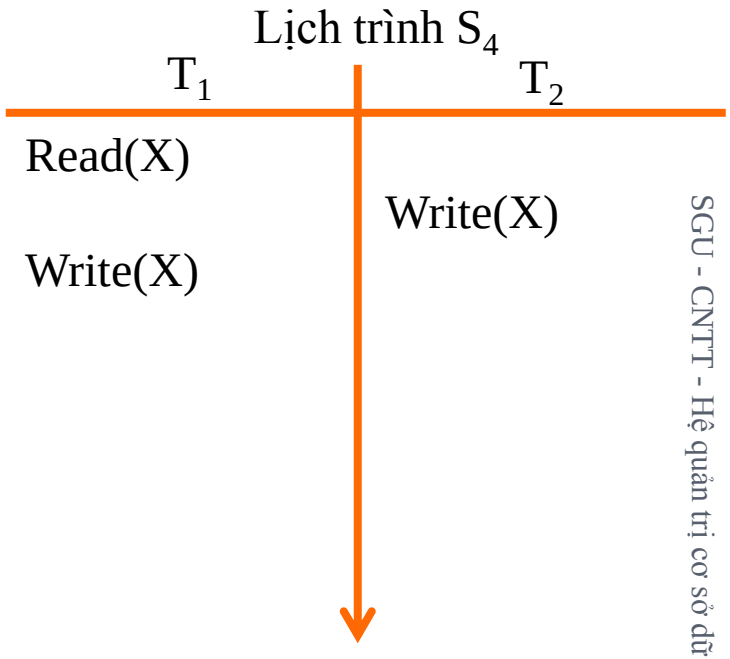
- Giao dịch T thực hiện thao tác Write(X):
  - Nếu Read_TS(X)>TS(T) hoặc Write_TS(X)>TS(T): hủy bỏ và rollback T vì đã có giao dịch sau T đọc hoặc ghi lên giá trị X trước khi T thực hiện Write(X).
  - Ngược lại thì thực hiện Write(X) và gán giá trị Write_TS(X)=TS(T)
- Giao dịch T thực hiện thao tác Read(X):
  - Nếu Write_TS(X)>TS(T): hủy bỏ và rollback vì T cần đọc giá trị X cũ nhưng đã bị ghi lên bởi một giao dịch nào đó sau T.
  - Nếu Write_TS(X)<=TS(T): thực hiện Read(X) và gán Read_TS(X) bằng giá trị lớn hơn trong 2 giá trị TS(T) và Read_TS(X).

THỨ TỰ NHÃN THỜI GIAN CƠ BẢN

- Khi phát hiện 2 chỉ thị xung đột (xét về mặt thứ tự thực hiện của chỉ thị), giải thuật này hủy bỏ chỉ thị mới hơn bằng cách hủy bỏ giao dịch tương ứng.
- Do đó đảm bảo tính khả tuần tự.
- Deadlock không xuất hiện tuy nhiên xuất hiện việc hủy bỏ - khởi tạo lại giao dịch → có khả năng xuất hiện starvation.

Ví dụ

- T1 → T2
- T1:Read(X) và T2:Write(X) được thực hiện thành công.
- Xét T1:Write(X)
  - Write_TS(X) ? TS(T1)
  - Write(X)?? Rollback??



Strict Timestamp Ordering

- Một biến thể của nhãn thời gian cơ bản đảm bảo tính khả phục hồi và tính khả tuần tự xung đột.

- Giao dịch T thực hiện thao tác Read(X) hoặc Write(X) mà TS(T)>Write_TS(X) sẽ tạm dừng thao tác đọc/ghi đó lại cho đến khi giao dịch T’ nào đó đã ghi giá trị X kết thúc (nghĩa là TS(T’)=Write_TS(X))

THOMAS'S WRITE RULE

- Một biến thể khác của kĩ thuật nhãn thời gian cơ bản không đảm bảo khả tuần tự xung đột của lịch trình nhưng hạn chế sự rollback trong thao tác Write(X):
  - Nếu Read_TS(X)>TS(T): hủy bỏ T, rollback.
  - Nếu Write_TS(X)>TS(T): không thực hiện thao tác Write(X) nhưng vẫn tiếp tục xử lý, **không rollback**.
  - Nếu không xảy ra 2 trường hợp trên: thực hiện thao tác Write(X) và thay đổi Write_TS(X)=TS(T)

KĨ THUẬT NHIỀU PHIÊN BẢN

- Giữ lại giá trị cũ của dữ liệu khi dữ liệu được cập nhật.
- Khi một giao dịch muốn truy xuất một dữ liệu, một phiên bản (version) thích hợp (giá trị) của dữ liệu được chọn để đảm bảo tính khả tuần tự.
- Ý tưởng: một số hành động Read không được chấp nhận ở các kĩ thuật khác vẫn có thể thực hiện thông qua phiên bản cũ của dữ liệu.

KĨ THUẬT NHIỀU PHIÊN BẢN

- Bất lợi: Cần phải lưu trữ nhiều dữ liệu hơn tuy nhiên thông thường các dữ liệu cũ vẫn cần được lưu trữ cho các mục đích khác như
  - Phục hồi (recovery)
  - Lịch sử thay đổi của dữ liệu (data history)
- 2 kĩ thuật nhiều phiên bản:
  - Dựa trên thứ tự nhãn thời gian (Timestamp Ordering).
  - Dựa trên khóa 2 pha (2PL).

MULTIVERSION – TIMESTAMP

- Nhiều phiên bản X_1, X_2, …, X_k của X được lưu trữ, với mỗi phiên bản X_i:
  - Read_TS(X_i): giá trị Timestamp lớn nhất của giao dịch đọc thành công X_i.
  - Write_TS(X_i): giá trị Timestamp của giao dịch ghi X_i.
- Mỗi khi giao dịch T thực hiện Write(X), một giá trị X_{k+1} của X được tạo ra với cả Write_TS(X_{k+1}) và Read_TS(X_{k+1}) bằng TS(T).
- Tương tự, mỗi khi giao dịch T thực hiện Read(X_i), giá trị của Read_TS(X_i) được gán bằng giá trị lớn hơn (mới hơn) giữa 2 giá trị Read_TS(X_i) hiện tại và TS(T).

MULTIVERSION – TIMESTAMP

Để đảm bảo tính khả tuần tự phải thỏa 2 luật sau:

- Nếu T thực hiện Write(X):
  - Nếu **Read_TS(Xi)**>TS(T) thì hủy bỏ và rollback T. Trong đó phiên bản i của X là phiên bản có giá trị Write_TS(Xi) lớn nhất trong tất cả phiên bản của X (nhưng nhỏ hơn hoặc bằng TS(T)).
  - Ngược lại, tạo phiên bản Xj của X với Read_TS(Xj)=Write_TS(Xj)=TS(T)

MULTIVERSION – TIMESTAMP

- Nếu T thực hiện Read(X), tìm phiên bản i của X có **Write_TS(Xi)** cao nhất và nhỏ hơn hoặc bằng TS(T), trả về giá trị Xi và thay đổi Read_TS(Xi) bằng giá trị lớn nhất trong 2 giá trị TS(T) và Read_TS(Xi)

- Ta có thể thấy, việc Read(X) luôn thành công.

MULTIVERSION – TIMESTAMP

- Trong trường hợp Write(X), giao dịch T có thể bị hủy bỏ khi T ghi một phiên bản X mà cần được đọc bởi giao dịch T’ (sau T, có timestamp là Read_TS(Xi)) nhưng T’ đã đọc phiên bản Xi được ghi bởi giao dịch có timestamp Write_TS(Xi) (không phải T).

MULTIVERSION – TIMESTAMP

- T: Write(X)
- Các phiên bản của X đang có tại thời điểm đó: X1, X2, … Xi, … Xn
- i là phiên bản thỏa điều kiện:
  1. Write_TS(Xi)<=TS(T)
  2. Write_TS(Xi) lớn nhất trong số các phiên bản phù hợp điều kiện 1 (sau cùng)
- Read_TS(Xi)>TS(T): rollback

KHÓA 2 PHA ĐA PHIÊN BẢN MULTIVERSION – 2PL

- Trong mô hình khóa tiêu chuẩn: khóa đọc (read lock/shared lock) và khóa ghi (write lock/exclusive lock)
- Bảng tương thích khóa:
  - Ngang: giao dịch giữ khóa
  - Dọc: giao dịch yêu cầu khóa
- 3 trạng thái khóa của một dữ liệu
  - Read-locked
  - Write-locked
  - Certify-locked

	Read	Write
Read	yes	no
Write	no	no

KHÓA 2 PHA ĐA PHIÊN BẢN MULTIVERSION – 2PL

- Ý tưởng: Cho phép giao dịch T’ đọc giá trị X trong khi X đang ở trạng thái khóa ghi bởi giao dịch T.
- Thực hiện: cho phép 2 phiên bản của X
  - Một phiên bản X chỉ được ghi khi giao dịch hoàn tất.
  - Một phiên bản X’ được tạo ra khi có giao dịch T giữ khóa ghi trên X.
- Các giao dịch có thể tiếp tục đọc giá trị X.
- Giao dịch T đang giữ khóa trên X có thể tiếp tục ghi các giá trị X’ khi cần.

KHÓA 2 PHA ĐA PHIÊN BẢN MULTIVERSION – 2PL

- Tuy nhiên, khi T muốn hoàn tất (commit), T phải đặt một khóa certify trên tất cả các giá trị mà T đang giữ khóa ghi.
- Khi đó T phải đợi cho đến khi tất cả các giá trị đó được mở khóa hoàn toàn bởi các giao dịch đang giữ khóa đọc mới có thể hoàn tất việc đặt khóa Certify.
- Cập nhật X bằng X’, xóa X’ và mở khóa certify.

KHÓA 2 PHA ĐA PHIÊN BẢN MULTIVERSION – 2PL

- Bảng tương thích khóa (Lock compatibility table)

	Read	Write	Certify
Read	yes	yes	no
Write	yes	no	no
Certify	no	no	no

MỘT SỐ KĨ THUẬT KHÁC

- Giao thức dựa trên tính hợp lệ: Validation (Optimistic) Concurrency Control Techniques.

- Đa hạt: Multiple Granularity Locking

END!

Tham khảo: Chương 23

Fundamentals of Database Systems, 6th Edition