Trường Đại Học Bách Khoa Tp.HCM
Hệ Đào Tạo Từ Xa
Khoa Khoa Học và Kỹ Thuật Máy Tính

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Chương 5. Xử lý giao tác (Transaction processing)

Bùi Hoài Thắng

Nội dung

* Dẫn nhập về xử lý giao tác
* Các khái niệm giao tác và hệ thống
* Các đặc tính cần có của các giao tác
* Đặc trưng hóa các thời biểu (schedule) dựa vào tính khả khôi phục
* Đặc trưng hóa các thời biểu dựa vào tính khả tuần tự hóa
* Hỗ trợ giao tác trong SQL chuẩn và trong SQL Server

BÀI 1 – TỔNG QUAN

Giới thiệu

* Hệ thống đơn người dùng (single-user system):
  - Có tối đa một người dùng tại một thời điểm có thể dùng hệ thống
* Hệ thống đa người dùng (multi-user system):
  - Nhiều người dùng có thể truy cập hệ thống đồng thời
* Các loại tương tranh (concurrency)
  - Xử lý đan xen (interleaved processing):
    - Các tiến trình (process) tương tranh sẽ được thi hành xen kẻ trên một CPU
  - Xử lý song song (parallel processing):
    - Các tiến trình được đồng thời thi hành trên nhiều CPU

Giới thiệu (tt.)

- Một giao tác (transaction):
  - Một đơn vị luận lý để xử lý dữ liệu
  - Bao gồm một hoặc nhiều tác vụ truy đạt dữ liệu
    - Đọc (read): truy xuất (retrieval)
    - Ghi (write): chèn (insert), cập nhật (update), xoá (delete)
- Một giao tác (là một tập các tác vụ) có thể được gửi đến hệ thống một cách độc lập (stand-alone) và được mô tả bằng một ngôn ngữ cấp cao như SQL hoặc được nhúng vào trong một chương trình
- Có thể xác định biên của giao tác (transaction boundary) bằng các lệnh **Begin transaction** và **End transaction**
- Một chương trình ứng dụng có thể gồm nhiều giao tác khác nhau (phân biệt bằng các biên của giao tác)

Mô hình một hệ CSDL đơn giản

* Là mô hình đơn giản dùng để giải thích vấn đề xử lý giao tác
* Một CSDL là một tập các mục liệu (data item) được đặt tên
* Độ mịn dữ liệu (granularity of data) là kích thước của mục dữ liệu:
  - Có thể là một mục tin (field), bản ghi (record) hoặc một khối (block)
  - Chúng ta chỉ xem xét các mục liệu độc lập với độ mịn của chúng
* Có 2 tác vụ cơ bản: đọc và ghi
  - **read_item(X)**: Đọc một mục liệu tên X trong CSDL và truyền vào một biến của chương trình. Để đơn giản, biến của chương trình cũng tên là X.
  - **write_item(X)**: Ghi giá trị của biến chương trình tên X vào mục liệu tên X trong CSDL.

Mô hình một hệ CSDL đơn giản (tt.)

* Đơn vị đọc/ghi dữ liệu:
  - Đơn vị dữ liệu cơ bản truyền giữa đĩa và bộ nhớ chính là 1 khối
  - Các mục liệu được đọc hay ghi thường là các mục tin của các bản ghi trong CSDL hoặc một bản ghi hoặc cả khối

* read_item(X):
  1. Tìm địa chỉ khối trên đĩa chứa mục liệu X
  2. Sao chép khối chứa mục liệu X lên vùng đệm (buffer) trong bộ nhớ chính (nếu vùng đệm chưa có khối này)
  3. Sao chép mục liệu X từ vùng đệm vào biến X trong chương trình

Mô hình một hệ CSDL đơn giản (tt.)

* write_item(X):
  1. Tìm địa chỉ của khối đĩa chứa mục liệu X
  2. Sao chép khối này vào vùng đệm trong bộ nhớ chính (nếu vùng đệm chưa có khối này)
  3. Sao chép biến X từ chương trình vào đúng mục liệu X trong vùng đệm
  4. Lưu và cập nhật khối (chứa mục liệu X) từ vùng đệm xuống đĩa (ngay lập tức hoặc vào một thời điểm sau này)

Ví dụ về 2 giao tác đơn giản

(a) T_1

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

(b) T_2

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

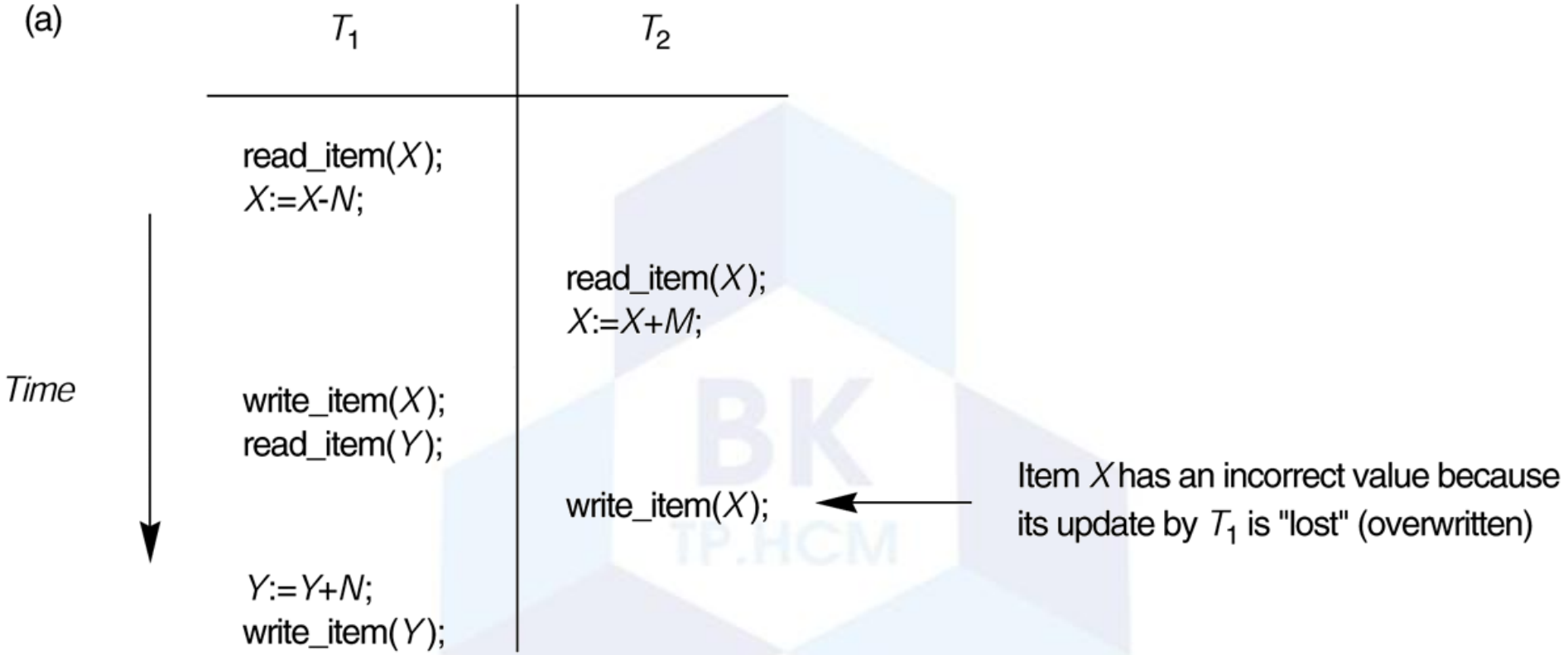
Hình 5.1

Các vấn đề khi có tương tranh

* Vấn đề mất cập nhật (Lost Update Problem)
  - Xảy ra khi hai giao tác truy đạt vào cùng các mục liệu theo cách đan xen và làm cho giá trị của một số mục liệu không đúng.
  - Ví dụ:
    - một giao tác sau ghi đè lên một mục liệu mà không biết là mục liệu này đã được cập nhật từ giao tác trước

Các vấn đề khi có tương tranh (tt.)

Lost Update Problem



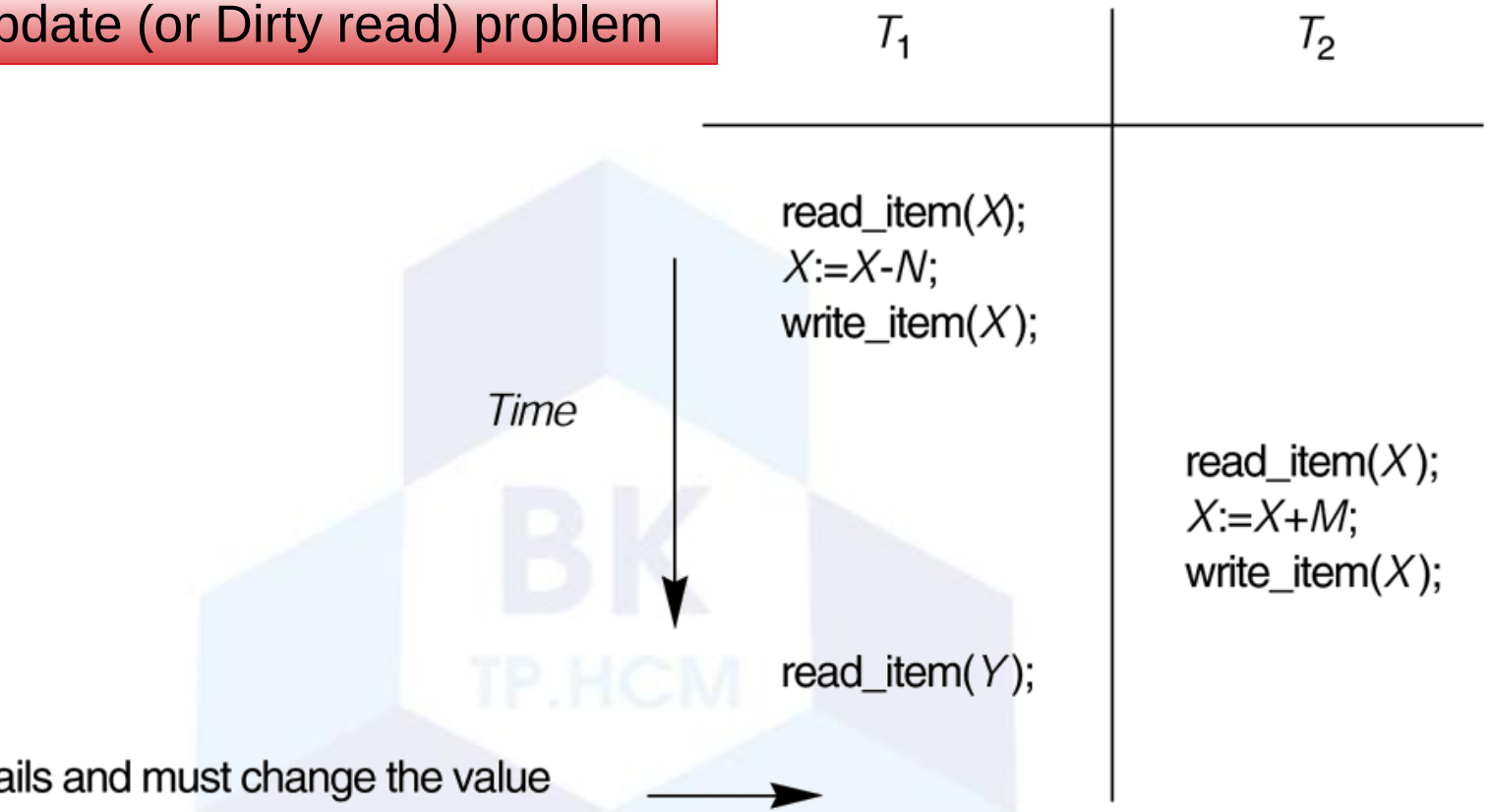
Hình 5.2

Các vấn đề khi có tương tranh (tt.)

* Vấn đề cập nhật tạm (hoặc đọc phải dữ liệu tạm) (The Temporary Update or Dirty Read Problem)
  - Xảy ra khi một giao tác cập nhật một mục liệu và sau đó bị hỏng, trong khi đó, một giao tác khác truy đạt vào mục liệu này trước khi nó được khôi phục giá trị gốc

Các vấn đề khi có tương tranh (tt.)

Temporal update (or Dirty read) problem



Transaction T_1 fails and must change the value of X back to its old value; meanwhile T_2 has read the "temporary" incorrect value of X.

Hình 5.3

Các vấn đề khi có tương tranh (tt.)

* Vấn đề tóm tắt sai (The Incorrect Summary Problem)
  - Khi một giao tác đang tính toán một tác vụ gộp (như SUM, AVG, …) trên một số bản ghi trong khi các giao tác khác đang cập nhật các bản ghi này, tác vụ gộp có thể sử dụng một số giá trị trước khi chúng được cập nhật và sử dụng một số giá trị sau khi chúng được cập nhật.
  - Ví dụ khác:
    - Một giao tác đọc một mục liệu 2 lần và giữa hai lần đó có một giao tác khác cập nhật => dữ liệu không nhất quán

Các vấn đề khi có tương tranh (tt.)

Incorrect summary problem

(c)

T_1	T_3
	sum:=0;
	read_item(*A*);
	sum:=*sum*+*A*;
	. . .
read_item(*X*);	
X:=*X*-N;	
write_item(*X*);	
	read_item(*X*);
	sum:=*sum*+*X*;
	read_item(*Y*);
	sum:=*sum*+*Y*;
read_item(*Y*);	
Y:=*Y*+*N*;	
write_item(*Y*);	

T_3 reads *X* after *N* is subtracted and reads *Y* before *N* is added; a wrong summary is the result (off by *N*).

Hình 5.4

Các vấn đề khi có tương tranh (tt.)

* Tóm tắt:
  - Có một số vấn đề có thể xảy ra khi hệ thống có nhiều các giao tác tương tranh
  - Cần thiết phải điều khiển tương tranh (concurency control)
    - Chương 6

Các sự cố gây hỏng giao tác

* 1. Hỏng máy tính (hoặc sụp hệ thống - system crash):
  - Lỗi phần cứng hoặc phần mềm xảy ra trong hệ thống máy tính khi thi hành giao tác
* 2. Lỗi giao tác:
  - Vài tác vụ của giao tác có thể gây ra hỏng giao tác (ví dụ lỗi chia cho số không)

Các sự cố gây hỏng giao tác (tt.)

* 3. Lỗi cục bộ hoặc các điều kiện ngoại lệ:
  - Một số điều kiện nào đó làm cho giao tác bị huỷ bỏ.
    - Ví dụ: dữ liệu dùng cho giao tác không tồn tại
  - Một lệnh huỷ được lập trình trong giao tác
* 4. Do tuân thủ điều khiển tương tranh:
  - Phương pháp điều khiển tương tranh có thể quyết định huỷ và khởi động lại (sau một thời gian) một giao tác để giải quyết vấn đề khoá chết (deadlock), khả tuần tự hoá (serializability)

Các sự cố gây hỏng giao tác (tt.)

* 5. Hỏng hóc đĩa:
  - Vài khối đĩa có thể mất dữ liệu do trục trặc khi đọc/ghi hoặc do đầu đọc ghi bị hỏng
* 6. Các vấn đề vật lý hoặc tai hoạ:
  - Như nguồn điện, cháy nổ, trộm, phá hoại, …

Các sự cố gây hỏng giao tác (tt.)

* Xử lý khi xảy ra sự cố hỏng hóc giao tác:
  - Khôi phục (recovery)
    - Chương 7

Các trạng thái và tác vụ của giao tác

* Giao tác:
  - Một đơn vị tác vụ đơn nguyên (atomic unit)
  - Hoặc hoàn tất toàn bộ các tác vụ trong giao tác
  - Hoặc không thực hiện tác vụ nào trong giao tác cả
* Các trạng thái của giao tác:
  - Trạng thái hoạt động (Active state)
  - Trạng thái commit bán phần (Partially committed state)
  - Trạng thái đã commit (Committed state)
  - Trạng thái hỏng (Failed state)
  - Trạng thái đã kết thúc (Terminated state)

Các trạng thái và tác vụ của giao tác (tt.)

* Các tác vụ của giao tác:
  - **begin_transaction**: bắt đầu thực hiện giao tác
  - **read** hoặc **write**: đọc hoặc ghi các mục liệu trong giao tác
  - **end_transaction**: kết thúc toàn bộ giao tác
    - Lúc này, cần phải kiểm tra là các thay đổi tạo ra trong giao tác có thể tác dụng vĩnh viễn hoặc giao tác cần phải bị huỷ bỏ do vi phạm các ràng buộc hệ thống (ví dụ như tương tranh)

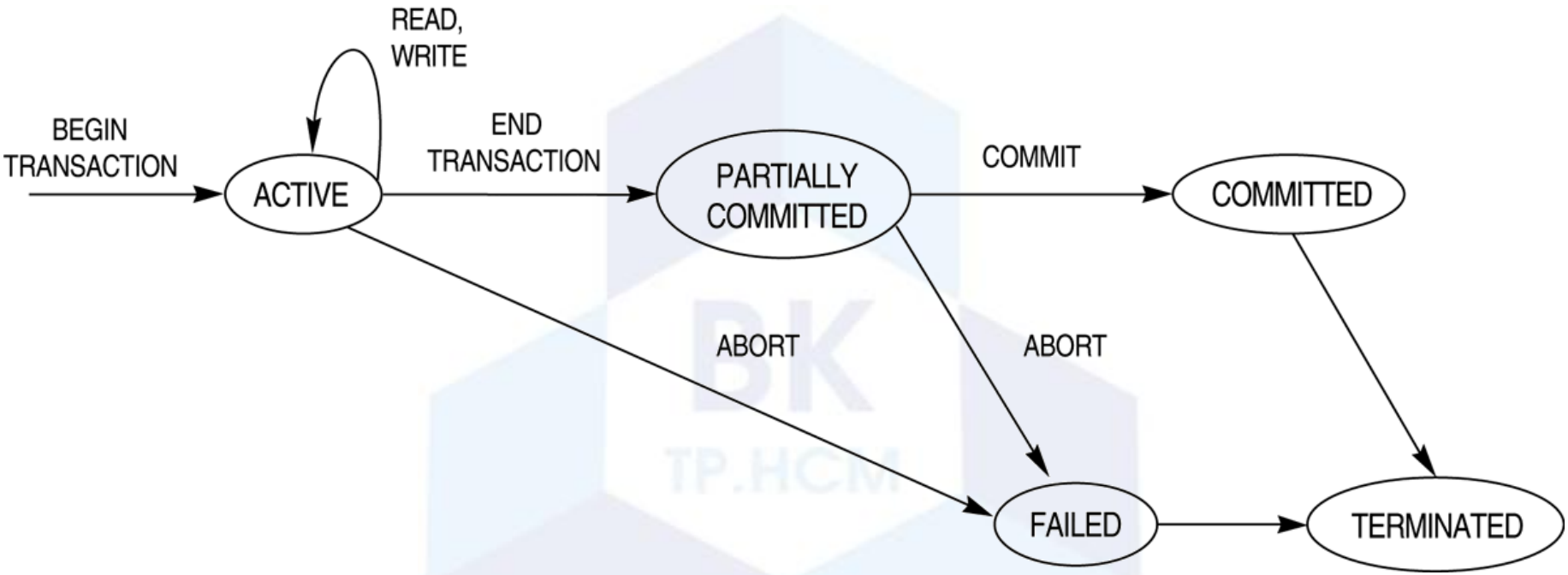
Các trạng thái và tác vụ của giao tác (tt.)

* Các tác vụ của giao tác (tt.):
  - **commit_transaction**:
    - Kết thúc giao tác và tất cả các thay đổi trong CSDL tạo ra bởi giao tác được commit xong và sẽ không thể tháo gỡ (undo) được
  - **rollback** (hoặc **abort**):
    - Kết thúc giao tác không thành công, các thay đổi tạo ra bởi giao tác cần phải được tháo gỡ (undo).

Các trạng thái và tác vụ của giao tác (tt.)

* Các kỹ thuật khôi phục dùng thêm các tác vụ sau:
  - **undo – tháo gỡ**: Giống như tác vụ quay ngược (rollback) nhưng áp dụng cho một tác vụ chứ không phải một giao tác
  - **redo – tái thực thi**: một số tác vụ trong giao tác phải được tái thực thi để đảm bảo là tất cả các tác vụ của một giao tác đã commit được áp dụng thành công vào CSDL

Các trạng thái và tác vụ của giao tác (tt.)



Sơ đồ chuyển trạng thái giao tác

Hình 5.5

Sổ ghi hệ thống (System Log)

* Sổ ghi (log) hoặc nhật ký (journal):
  - Sổ ghi dùng để theo dõi tất cả các tác vụ của giao tác
  - Thông tin trong sổ ghi dùng để khôi phục khi giao tác thất bại
* Sổ ghi được lưu trên đĩa
  - Tránh được các hỏng hóc bất thường trừ khi bị lỗi đĩa hoặc tai hoạ (như mất cắp, phá hoại, …)
  - Ngoài ra, sổ ghi thường được sao chép dự phòng thường xuyên đến các bộ lưu trữ ổn định (như băng từ - tape) khác để tránh các tai hoạ

Sổ ghi hệ thống (tt.)

* Các loại bản ghi trong sổ ghi (log record):
  - **[start_transaction, T]**: ghi nhận là giao tác T đã bắt đầu thực hiện
  - **[write_item, T, X, old_value, new_value]**: ghi nhận là giao tác T đã thay đổi giá trị của mục tin X từ giá trị old_value thành new_value
  - **[read_item, T, X]**: ghi nhận là giao tác T đã đọc giá trị từ mục tin X
  - **[commit, T]**: ghi nhận là giao tác T đã hoàn tất thành công, tất cả các thay đổi từ giao tác này có thể được lưu trữ vĩnh viễn trong CSDL
  - **[abort, T]**: ghi nhận là giao tác T đã bị huỷ

Sổ ghi hệ thống (tt.)

* Ghi chú:
  - Một số giao thức khôi phục (recovery protocol) theo hướng tránh các quay ngược dắt dây (cascading rollback) không cần các tác vụ READ lưu trong sổ ghi
  - Một số giao thức khôi phục như giao thức nghiêm cách (strict protocol) chỉ cần các mục sổ ghi (log entry) WRITE đơn giản: không cần các giá trị mới (new_value)

Sổ ghi hệ thống (tt.)

* Khôi phục dùng sổ ghi:
  - Khi hệ thống sụp, chúng ta có thể khôi phục hệ thống về một trạng thái CSDL nhất quán bằng cách khám xét sổ ghi và dùng các kỹ thuật khôi phục
    - Chương 7
  - Do sổ ghi chứa các thông tin về tất cả các tác vụ thay đổi giá trị của các mục tin CSDL, nên có thể tháo gỡ (undo) các tác vụ đó bằng cách truy ngược trên sổ ghi và phục hồi mục tin tương ứng về giá trị trước đó của nó (dùng old_value)
  - Cũng có thể duyệt xuôi chiều trên sổ ghi để tái thực thi (redo) một giao tác bằng cách thay đổi các mục tin trong các lệnh WRITE tương ứng về giá trị mới của nó (new_value)
    - Các tác vụ ghi rồi nhưng chưa kịp lưu trữ vĩnh viễn có thể được lặp lại thông qua việc tái thực thi này

Điểm Commit (Commit point)

* Định nghĩa:
  - Một giao tác T đạt đến điểm commit khi tất cả các tác vụ truy đạt CSDL của nó đã hoàn thành và các thay đổi CSDL của nó đã được ghi nhận trong sổ ghi.
* Sau điểm commit, một giao tác được gọi là đã commit:
  - Các ảnh hưởng của nó đối với CSDL được cho là đã lưu trữ vĩnh viễn trong CSDL
  - Ghi mục [commit, T] vào sổ ghi

Điểm Commit (tt.)

* Nếu có trục trặc xảy ra:
  - Các giao tác đã ghi mục [start_transaction,T] vào sổ ghi nhưng chưa có [commit,T] sẽ phải bị quay ngược (rollback):
  - Các giao tác đã có đủ 2 mục trên có thể cần phải tái thực thi (redo)
* Buộc ghi sổ ghi trước:
  - Trước khi một giao tác đạt đến điểm commit, các phần chưa ghi xuống đĩa của sổ ghi cần phải ghi xuống đĩa

Các thuộc tính ACID của giao tác

* ACID = **A**tomicity + **C**onsistency preservation + **I**solation + **D**urability

* **Atomicity – Tính đơn thể**:
  - Một giao tác là một đơn vị đơn thể (atomic unit) các tác vụ: được thực thi toàn bộ (các tác vụ) hoặc không thực thi tác vụ nào cả

Các thuộc tính ACID của giao tác (tt.)

* **Consistency preservation – Tính bảo toàn tính nhất quán**:
  - Một sự thực thi đúng của một giao tác phải chuyển CSDL từ trạng thái nhất quán này sang trạng thái nhất quán khác
  - Trạng thái nhất quán của CSDL:
    - Thoả tất cả các ràng buộc (constraint) của lược đồ CSDL và các ràng buộc bắt buộc khác

Các thuộc tính ACID của giao tác (tt.)

* **Isolation – Tính đơn lập**:
  - Một giao tác cần được thực thi một cách đơn lập so với các giao tác khác: Sự thực thi của nó không thể bị can thiệp bởi bất kỳ các giao tác đang được thực thi cùng lúc
* **Durability** hoặc **permanency – Tính bền vững**:
  - Một khi một giao tác thay đổi CSDL đã được commit, các thay đổi này cần phải đảm bảo để không thể mất mát do các hỏng hóc xảy ra sau đó.

Lịch biểu giao tác

Lịch biểu giao tác (Transaction Schedule)

* Lịch biểu giao tác (Transaction schedule hoặc history):
  - Khi các giao tác được thực thi tương tranh theo cách đan xen, thứ tự thực thi của các tác vụ từ các giao tác tạo thành lịch biểu giao tác (hoặc lịch sử giao tác)
* Ràng buộc lịch biểu:
  - Trong một lịch biểu (schedule) S của n giao tác T1, T2, ..., Tn
  - Thứ tự thực thi của các tác vụ trong một giao tác Ti trong S được giữ nguyên như được mô tả trong Ti, mặc dù các tác vụ này có thể được thực thi đan xen với các tác vụ của các giao tác Tj khác (trong S)

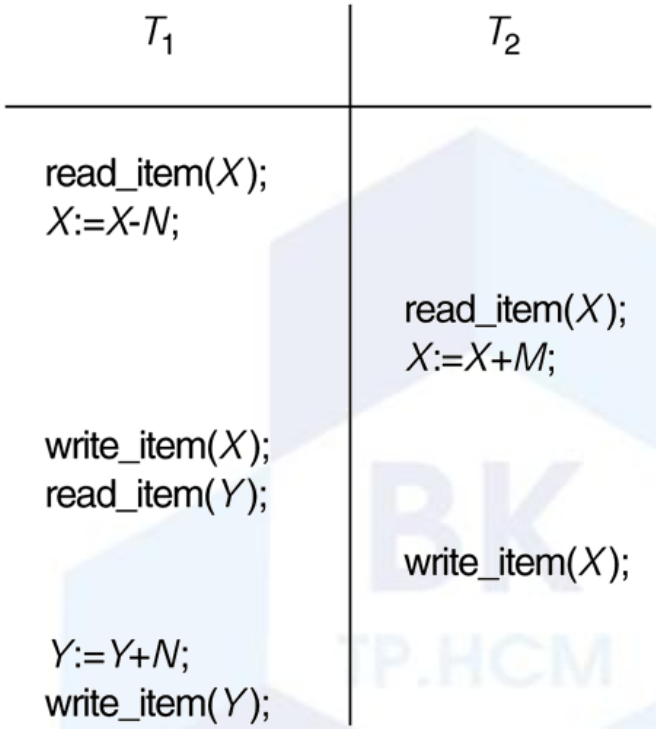
Lịch biểu giao tác (tt.)

* Để đơn giản, chỉ các tác vụ read_item, write_item, commit và abort được thể hiện trên lịch biểu
* Viết tắt các tác vụ của giao tác:

Viết tắt	Tác vụ
$r_i(X)$	read_item(X) của giao tác T_i
$w_i(X)$	write_item(X) của giao tác T_i
c_i	commit của giao tác T_i
a_i	abort của giao tác T_i

Lịch biểu giao tác (tt.)

* Ví dụ:

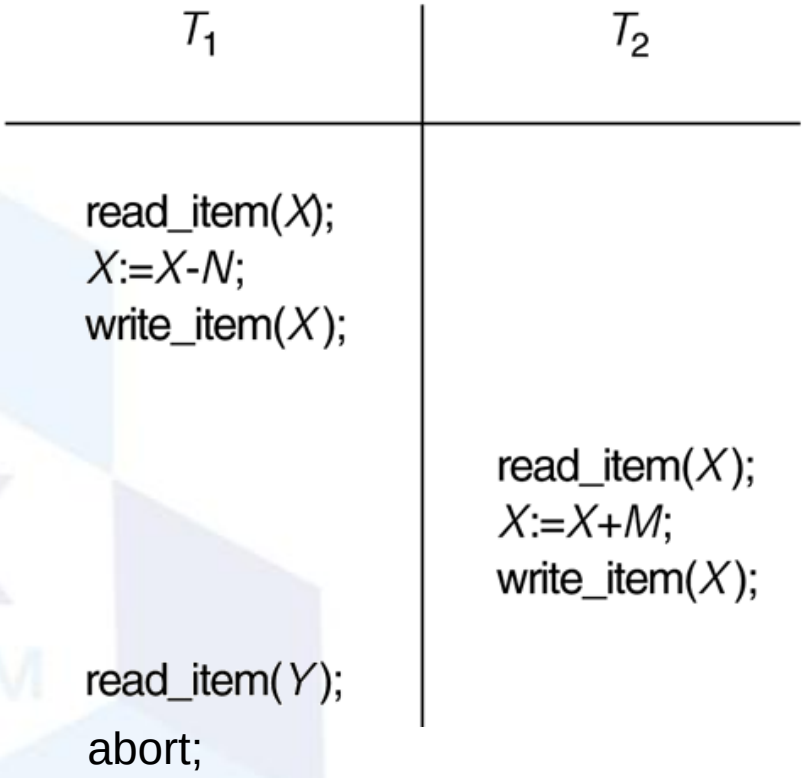


➢ S_a: $r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$;

Hình 5.6

Lịch biểu giao tác (tt.)

* Ví dụ khác:



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

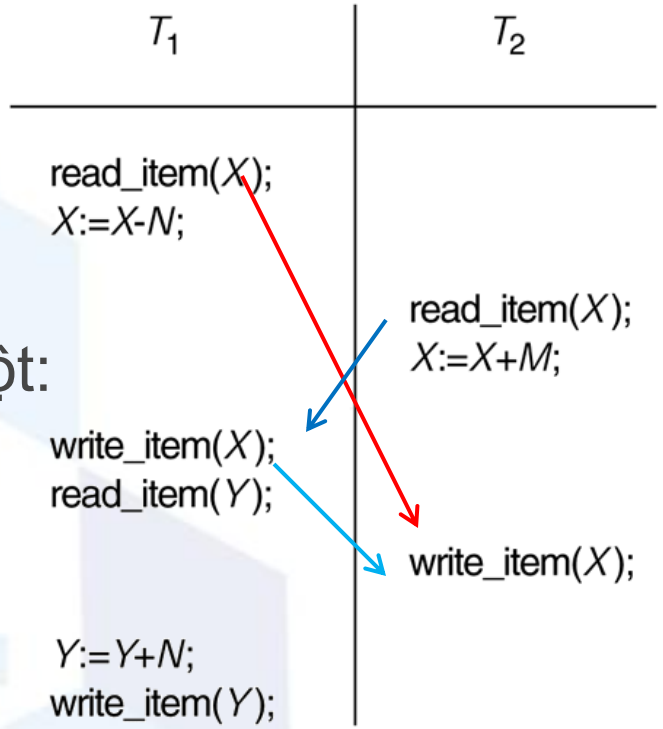
Hình 5.7

Lịch biểu giao tác (tt.)

* Hai tác vụ trong một lịch biểu là có xung đột (conflict) nếu:
  - (1) chúng thuộc về hai giao tác khác nhau
  - (2) chúng truy đạt vào cùng một mục liệu
  - (3) ít nhất một trong chúng là lệnh ghi (write_item)

Lịch biểu giao tác (tt.)

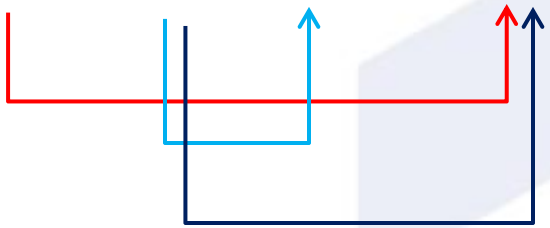
* Ví dụ các tác vụ có xung đột:
  - $r_1(X)$ và $w_2(X)$
  - $r_2(X)$ và $w_1(X)$
  - $w_1(X)$ và $w_2(X)$
* Ví dụ các tác vụ không có xung đột:
  - $r_1(X)$ và $r_2(X)$
  - $r_1(Y)$ và $w_2(X)$
  - $r_1(X)$ và $w_1(X)$
  - …



Hình 5.8

Lịch biểu giao tác (tt.)

* Ví dụ khác:
  - Sb: r1(X); w1(X); r2(X); w2(X); r1(Y); a1;



  - Các tác vụ có xung đột:
    - $r_1(X)$ và $w_2(X)$
    - $w_1(X)$ và $r_2(X)$
    - $w_1(X)$ và $w_2(X)$

Lịch biểu giao tác (tt.)

* Một lịch biểu S của n giao tác T1, T2, … Tn là một lịch biểu hoàn thành (complete schedule) nếu:
  - (1) Các tác vụ trong S chính là tất cả các tác vụ trong các giao tác T1, T2, …, Tn bao gồm cả một lệnh commit hoặc abort là lệnh cuối cùng của mỗi giao tác trong lịch biểu
  - (2) Với bất kỳ cặp tác vụ nào từ cùng một giao tác T_i, thứ tự của chúng trong S là thứ tự của chúng xuất hiện trong T_i
  - (3) Với bất kỳ hai tác vụ có xung đột nào, một trong hai tác vụ này phải xảy ra trước tác vụ còn lại trong lịch biểu

Lịch biểu giao tác (tt.)

* Vì tất cả các giao tác hoặc là commit hoặc là abort nên không còn tác vụ nào đang hoạt động khi kết thúc lịch biểu
* Trên thực tế các lịch biểu không bao giờ hoàn thành vì các giao tác liên tục xuất hiện
* Ta chỉ xem xét trạng thái committed projection của một lịch biểu:
  - Chỉ xem xét các tác vụ của các giao tác đã hoàn thành

Tóm tắt bài 1

- Dẫn nhập về xử lý giao tác
  - Giao tác
  - Mô hình CSDL đơn giản
  - Vấn đề tương tranh
  - Vấn đề khôi phục
- Các khái niệm giao tác và hệ thống
  - Trạng thái giao tác
  - Sổ ghi
  - Điểm commit
- Các đặc tính cần có của các giao tác
  - Atomicity
  - Consistency preservation
  - Isolation
  - Durability
- Lịch biểu giao tác
  - Lịch biểu
  - Các tác vụ xung đột
  - Lịch biểu hoàn thành