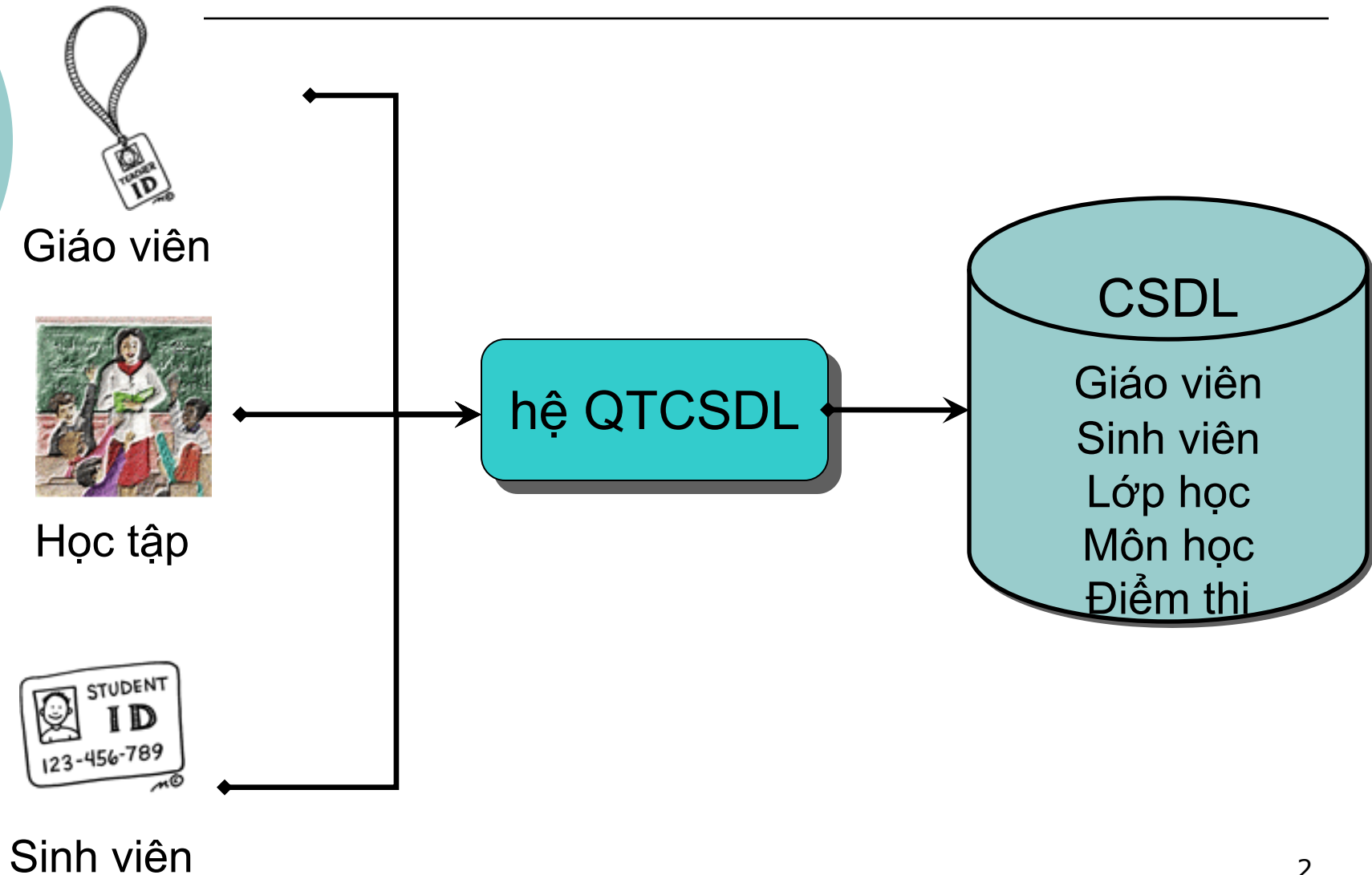





An toàn và toàn vẹn dữ liệu

Ví dụ





GIAO_VIEN (maGV, hoten, ngaysinh, gioitinh, diachi,
hocham, hocvi, bacluong)

LOP (malop, tenlop, khoa, maloptruong, maGVVCN, TSSV)

SINH_VIEN(maSV, hoten, ngaysinh, tuoi, gioitinh, diachi,
malop)

MON_HOC(mamon, tenmon, soHT)

DIEM_THI(maSV, mamon, lanthi, diem)

Đặt vấn đề

- Mục đích của CSDL
 - Lưu trữ lâu dài
 - Khai thác hiệu quả
- Yêu cầu đ/v thiết kế CSDL
 - Đảm bảo tính đúng đắn của DL
 - Tránh sai sót khi cập nhật DL \Rightarrow định nghĩa và kiểm tra các ràng buộc DL
 - Tránh sai sót trong quá trình thao tác với DL \Rightarrow kiểm tra tính toàn vẹn của các thao tác với DL
 - Đảm bảo tính an toàn của DL
 - Tránh truy nhập DL không hợp lệ từ phía người dùng \Rightarrow phân quyền và kiểm tra quyền hạn người sử dụng



Nội dung

- An toàn dữ liệu
- Ràng buộc dữ liệu
- Toàn vẹn dữ liệu

An toàn dữ liệu

- Bảo vệ CSDL chống lại sự truy nhập bất hợp pháp
- Cần các cơ chế cho phép:
 - Nhận biết người dùng
 - Xác định các thao tác hợp lệ với từng (nhóm) người dùng

Lệnh tạo (nhóm) người dùng

- Cú pháp

- Tạo người dùng

- ```
CREATE USER username
IDENTIFIED BY password;
```

- Xoá người dùng

- ```
DROP USER name [CASCADE];
```

- Ví dụ

- ```
CREATE USER tin123K47
IDENTIFIED BY nmcsdl
```

# Lệnh phân quyền cho người dùng

---

- Cú pháp

`Grant` <privilege> `On` <Object> `To` <user>  
[With Grant Option]

`REVOKE` <privilege> `ON` <Object> `FROM` <user>  
[RESTRICT | CASCADE]

<privilege> = {Insert | Update | Delete | Select |  
Create | Alter | Drop | Read | Write}

<object > = {Table | View}

- Ví dụ:

`GRANT` SELECT `ON` DIEM\_THI `TO` tin123K47

`GRANT` SELECT, UPDATE `ON` DIEM\_THI `TO` vutrinh  
`WITH GRANT OPTION`





# Nội dung

---

- An toàn dữ liệu
- Ràng buộc dữ liệu
- Toàn vẹn dữ liệu

# Ràng buộc dữ liệu

---

- Mục đích: định nghĩa tính đúng đắn của DL trong toàn bộ CSDL
- Phân loại
  - Ràng buộc về miền giá trị
    - Trên 1 thuộc tính
    - Trên nhiều thuộc tính (cùng 1 bản ghi)
    - Trên nhiều bản ghi
  - Ràng buộc về khoá
    - Trên 1 quan hệ: khoá chính
    - Trên nhiều quan hệ: khoá ngoài

# Lệnh đ/n ràng buộc miền giá trị

---

- Cú pháp

`CONSTRAINT <ten-rang-buoc> CHECK <dieu-kien>`

- Ví dụ:

- Trong bảng DIEM

`CONSTRAINT gtdiem CHECK ((diem>=0) and  
(diem<=10))`

- Trong bảng SINH\_VIEN

`CONSTRAINT gttuoi CHECK (tuoi = year(date()) –  
year(ngaysinh))`

# Lệnh đ/n ràng buộc khoá chính

---

- Cú pháp

`CONSTRAINT` <ten-rang-buoc>

`PRIMARY KEY` <cac-thuoc-tinh-khoa>

- Ví dụ

- Trong bảng SINH\_VIEN

`CONSTRAINT` SV-khoa `PRIMARY KEY` maSV

- Trong bảng DIEM

`CONSTRAINT` diemthi-khoa `PRIMARY KEY` (maSV,  
mamon)

# Lệnh đ/n ràng buộc khoá ngoài

---

- Cú pháp

`CONSTRAINT <ten-rang-buoc>`

`FOREIGN KEY <cac-thuoc-tinh-khoa>`

`REFERENCES <ten-bang>[khoa-tham-chieu]`

- Ví dụ: Trong bảng DIEM

`CONSTRAINT diem-SV FOREIGN KEY maSV`

`REFERENCES SINH_VIEN[maSV]`

`CONSTRAINT diem-mon FOREIGN KEY mamon`

`REFERENCES MON_HOC[mamon]`



# Nội dung

---

- An toàn dữ liệu
- Ràng buộc dữ liệu
- Toàn vẹn dữ liệu

# Toàn vẹn dữ liệu

---

- Mục đích: đảm bảo tính đúng đắn của DL trong quá trình thao tác (thêm, sửa, xoá DL)
- Yêu cầu
  - Kiểm tra các ràng buộc toàn vẹn DL khi thực hiện các thao tác thêm, sửa, xoá
    - sử dụng các triggers
  - Kiểm tra tính đúng đắn của các thao tác trên CSDL
    - Quản trị giao dịch
    - Điều khiển tương tranh

# Trigger

---

- Đ/n

- Là các xử lý được gắn với các bảng DL
- Được tự động kích hoạt khi thực hiện các thao tác thêm, sửa, xóa bản ghi

- Cú pháp

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
{BEFORE | AFTER | INSTEAD OF }
{UPDATE | INSERT | DELETE}
[OF <attribute_name>] ON <table name>
[FOR EACH ROW]
BEGIN
 << trigger body goes here >>
END <trigger_name>;
```



# Ví dụ

---

LOP (malop, tenlop, khoa, maloptruong, maGVCN, TSSV)  
SINH\_VIEN(maSV, hoten, ngaysinh, tuoi, gioitinh, diachi, malop)

```
CREATE TRIGGER tang_TSSV
AFTER INSERT ON SINH_VIEN
FOR EACH ROW
BEGIN
 update LOP set TSSV= TSSV+1
 where malop = :new.malop
END;
```

# Ví dụ

---

LOP (malop, tenlop, khoa, maloptruong, maGVCN, TSSV)  
SINH\_VIEN(maSV, hoten, ngaysinh, tuoi, gioitinh, diachi, malop)

```
CREATE TRIGGER giam_TSSV
AFTER DELETE ON SINH_VIEN
FOR EACH ROW
BEGIN
 update LOP set TSSV= TSSV-1
 where malop = :old.malop
END;
```

# Giao dịch – ví dụ

---



Đọc số dư của tài khoản A

Kiểm tra (số dư > số tiền cần rút)

Tăng số dư của tài khoản B

Giảm số dư của tài khoản A

← **Sự  
cố**

**Ngân hàng  
chịu lỗi ???**

# Giao dịch

---

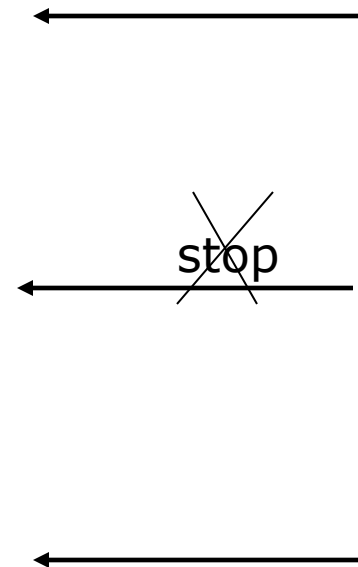
- Đ/n: một tập các thao tác được xử lý như **một đơn vị không chia cắt được**
  - Cho phép đảm bảo tính nhất quán và tính đúng đắn của dữ liệu
- Tính chất ACID
  - Nguyên tố (**A**tomicity)
  - Tính nhất quán (**C**onsistency)
  - Tính cô lập (**I**solation)
  - Tính bền vững (**D**urability)

|   |                        |
|---|------------------------|
| } | Điều khiển tương tranh |
| } | Phục hồi dữ liệu       |

# Tính nguyên tử (*Atomicity*)

- Đ/n: Hoặc là toàn bộ hành động của giao dịch được thực hiện hoặc không có hành động nào được thực hiện
- Ví dụ:

```
T: Read(A,t1);
 If t1 > 500 {
 Read(B,t2);
 t2:=t2+500;
 Write(B,t2);
 t1:=t1-500;
 Write(A,t1);
 }
```



# Tính nhất quán (*Consistency*)

---

- Đ/n: Tính nhất quán của dữ liệu trước khi bắt đầu và sau khi kết thúc giao dịch
- Ví dụ

```
T: Read(A,t1);
 If t1 > 500 {
 Read(B,t2);
 t2:=t2+500;
 Write(B,t2);
 t1:=t1-500;
 Write(A,t1);
 }
```

←  $A+B = C$

←  $A+B = C$

# Tính cô lập (*Isolation*)

- Đ/n: 1 giao dịch được tiến hành độc lập với các giao dịch khác tiến hành đồng thời
- Ví dụ:  $A = 5000$ ,  $B = 3000$

```
T: Read(A,t1);
 If t1 > 500 {
 Read(B,t2);
 t2:=t2+500;
 Write(B,t2);
 t1:=t1-500;
 Write(A,t1);
 }
```

←  $T': A+B$   
     $(= 5000+3500)$

←  $(A+B = 4500+3500)$

# Tính bền vững (*Durability*)

---

- Đ/n

- Mọi thay đổi mà giao dịch thực hiện trên CSDL phải được ghi nhận bền vững

- Ví dụ:  $A = 5000$ ,  $B = 3000$

```
T: Read(A,t1);
 If t1 > 500 {
 Read(B,t2);
 t2:=t2+500;
 Write(B,t2);
 t1:=t1-500;
 Write(A,t1);
 }
```

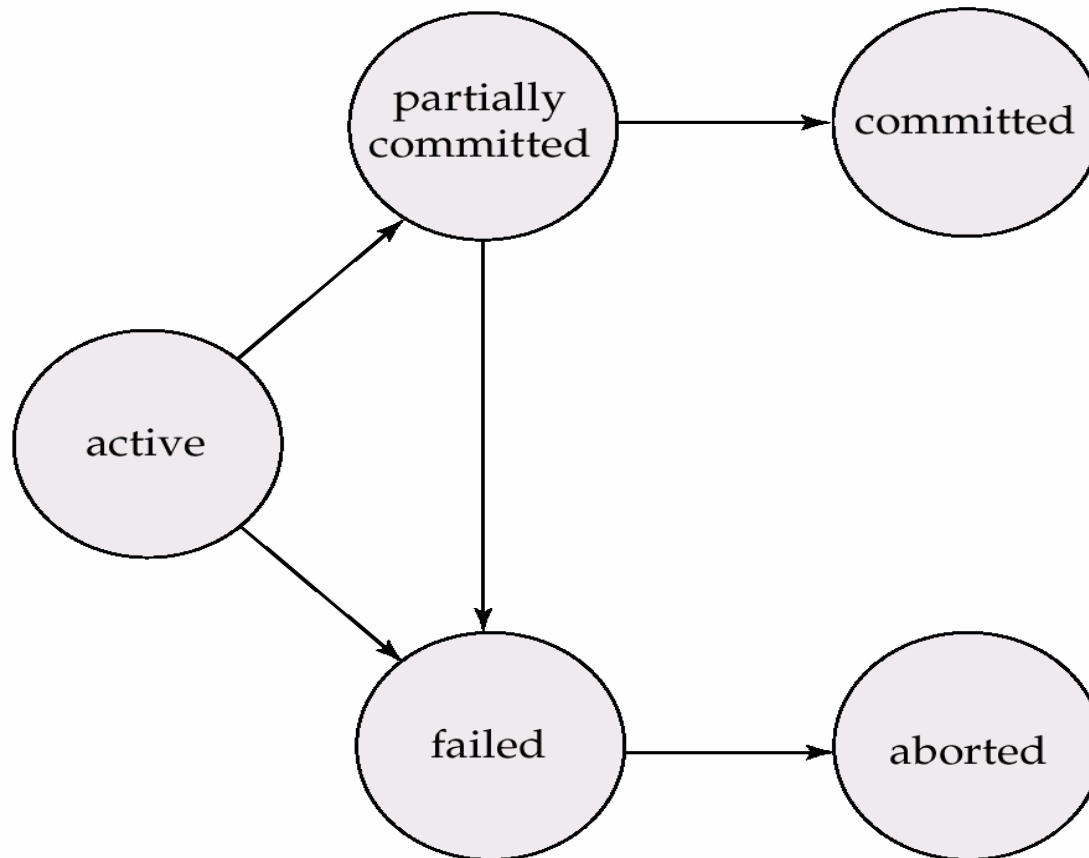
← sự cố

$A = 4500$ ,  $B = 3500$



# Trạng thái của giao dịch

---



# Điều khiển tương tranh

---

- Mục đích: tránh đụng độ giữa các giao dịch (một dãy các thao tác) trên cùng một đối tượng có thể làm mất tính nhất quán của DL

|                                                                                                                        |                                                                                                                                                     |
|------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>T0: read(A);<br/>    A := A - 50;<br/>    write(A);<br/>    read(B);<br/>    B := B + 50;<br/>    write(B);</pre> | <pre>T1: read(A);<br/>    temp := A * 0.1;<br/>    A := A - temp;<br/>    write(A);<br/>    read(B);<br/>    B := B + temp;<br/>    write(B);</pre> |
|------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|

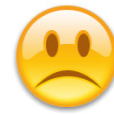
# Ví dụ về thực hiện giao dịch



| T <sub>0</sub>                                                           | T <sub>1</sub>                                                                                  |
|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| read(A)<br>A := A - 50<br>write(A)<br>read(B)<br>B := B + 50<br>write(B) | read(A)<br>temp := A * 0.1<br>A := A - temp<br>write(A)<br>read(B)<br>B := B + temp<br>write(B) |



| T <sub>0</sub>                                                           | T <sub>1</sub>                                                                                  |
|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| read(A)<br>A := A - 50<br>write(A)<br>read(B)<br>B := B + 50<br>write(B) | read(A)<br>temp := A * 0.1<br>A := A - temp<br>write(A)<br>read(B)<br>B := B + temp<br>write(B) |



| T <sub>0</sub>                                                               | T <sub>1</sub>                                                                                      |
|------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| read(A)<br>A := A - 50<br><br>write(A)<br>read(B)<br>B := B + 50<br>write(B) | read(A)<br>temp := A * 0.1<br>A := A - temp<br>write(A)<br>read(B)<br><br>B := B + temp<br>write(B) |

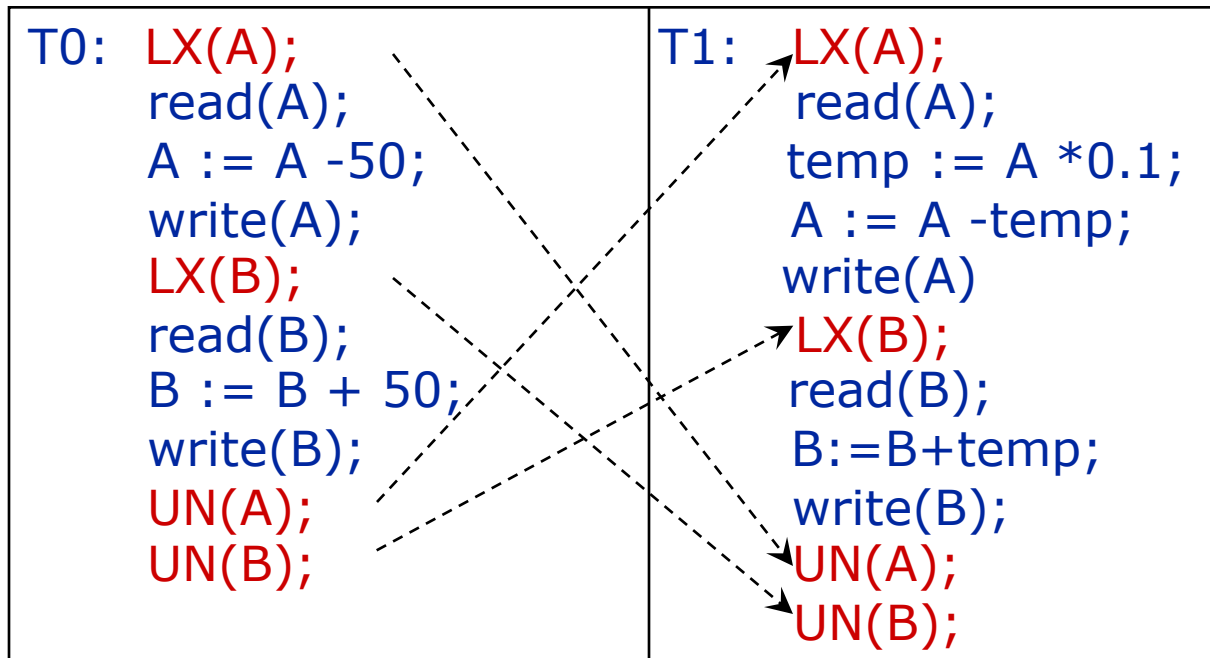
# Kỹ thuật khoá

---

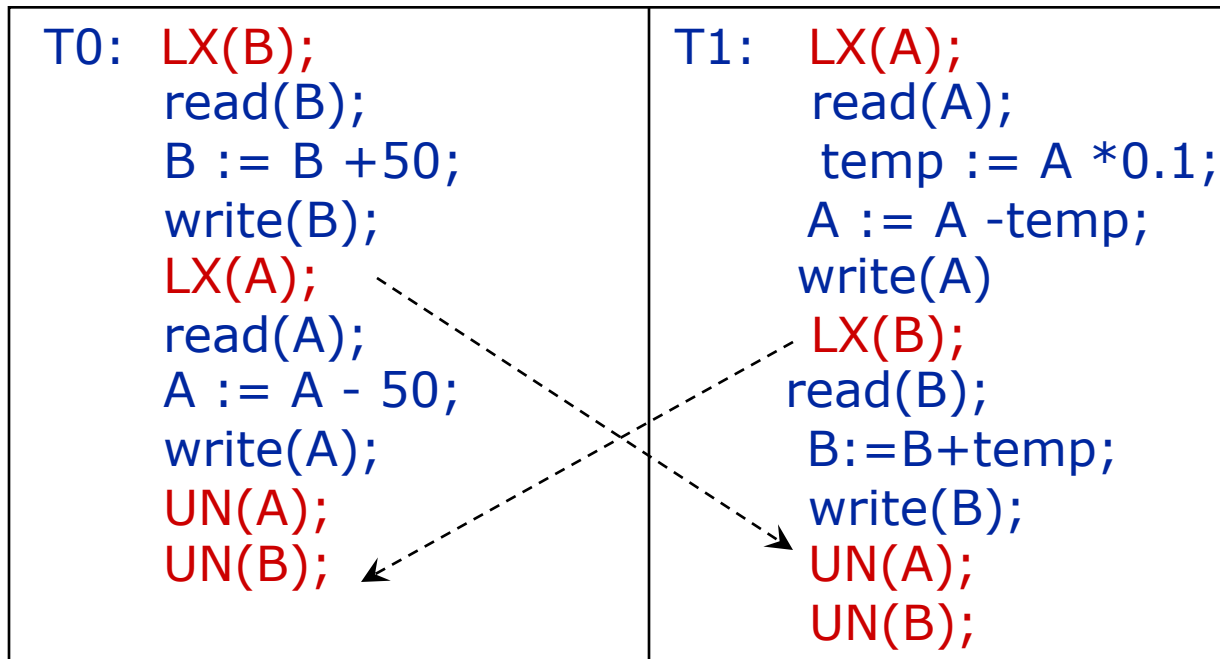
- Mục đích
  - Đảm bảo việc truy nhập đến các DL được thực hiện theo phương pháp loại trừ nhau
- Các kiểu khoá
  - Chia sẻ: có thể đọc nhưng không ghi DL
  - Độc quyền: đọc và ghi DL
- Ký hiệu
  - LS(D): khoá chia sẻ
  - LX(D): khoá độc quyền
  - UN(D): mở khoá
- Tính tương thích:

|    | LS    | LX    |
|----|-------|-------|
| LS | true  | false |
| LX | false | false |

# Ví dụ



# Khoá chết (*deadlock*)



# Các vấn đề về quản trị giao dịch

---

- Các kỹ thuật điều khiển tương tranh
  - các chế độ khoá, giải quyết khoá chết
  - kỹ thuật gán nhãn
- Lập lịch
- Các kỹ thuật phục hồi (*recovery*)
- ...

# Kết luận

---

Để đảm bảo tính an toàn và toàn vẹn dữ liệu

- Đ/v người thiết kế CSDL

- Phải định nghĩa các ràng buộc toàn vẹn về dữ liệu

- Đ/v người quản trị hệ thống

- Phải định nghĩa các khung nhìn
- Phải phân quyền cho (nhóm) người dùng

- Đ/v hệ CSDL

- Phải xác minh được người dùng
- Phải kiểm tra các ràng buộc DL một cách tự động
- Phải đảm bảo các tính chất ACID cho giao dịch người dùng



