Chương 6

GIAO TÁC SQL

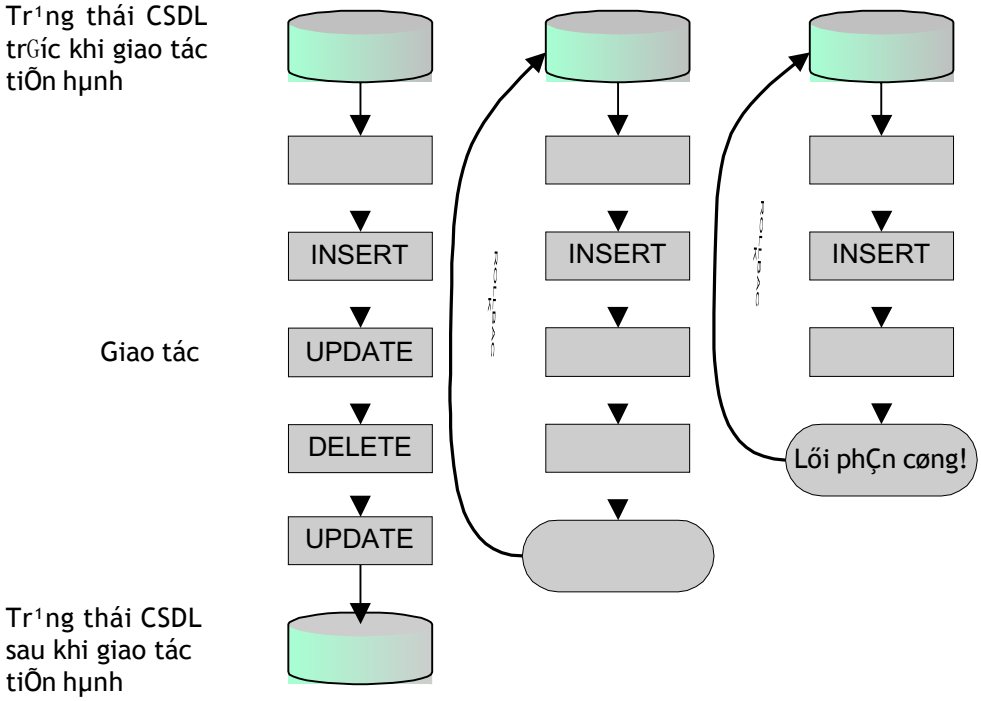
6.1 Giao tác và các tính chất của giao tác

Một giao tác (transaction) là một chuỗi một hoặc nhiều câu lệnh SQL được kết hợp lại với nhau thành một khối công việc. Các câu lệnh SQL xuất hiện trong giao tác thường có mối quan hệ tương đối mật thiết với nhau và thực hiện các thao tác độc lập. Việc kết hợp các câu lệnh lại với nhau trong một giao tác nhằm đảm bảo tính toàn vẹn dữ liệu và khả năng phục hồi dữ liệu. Trong một giao tác, các câu lệnh có thể độc lập với nhau nhưng tất cả các câu lệnh trong một giao tác đòi hỏi hoặc phải thực thi trọn vẹn hoặc không một câu lệnh nào được thực thi.

Các cơ sở dữ liệu sử dụng *nhật ký giao tác* (*transaction log*) để ghi lại các thay đổi mà giao tác tạo ra trên cơ sở dữ liệu và thông qua đó có thể phục hồi dữ liệu trong trường hợp gặp lỗi hay hệ thống có sự cố.

Một giao tác đòi hỏi phải có được bốn tính chất sau đây:

- **Tính nguyên tử (Atomicity):** Mọi thay đổi về mặt dữ liệu hoặc phải được thực hiện trọn vẹn khi giao tác thực hiện thành công hoặc không có bất kỳ sự thay đổi nào về dữ liệu xảy ra nếu giao tác không thực hiện được trọn vẹn. Nói cách khác, tác dụng của các câu lệnh trong một giao tác phải như là một câu lệnh đơn.
- **Tính nhất quán (Consistency):** Tính nhất quan đòi hỏi sau khi giao tác kết thúc, cho dù là thành công hay bị lỗi, tất cả dữ liệu phải ở trạng thái nhất quán (tức là sự toàn vẹn dữ liệu phải luôn được bảo toàn).
- **Tính độc lập (Isolation):** Tính độc lập của giao tác có nghĩa là tác dụng của mỗi một giao tác phải giống như khi chỉ mình nó được thực hiện trên chính hệ thống đó. Nói cách khác, một giao tác khi được thực thi đồng thời với những giao tác khác trên cùng hệ thống không chịu bất kỳ sự ảnh hưởng nào của các giao tác đó.
- **Tính bền vững (Durability):** Sau khi một giao tác đã thực hiện thành công, mọi tác dụng mà nó đã tạo ra phải tồn tại bền vững trong cơ sở dữ liệu, cho dù là hệ thống có bị lỗi đi chăng nữa.



Hình 6.1: *Giao tác SQL*

6.2 Mô hình giao tác trong SQL

Giao tác SQL được định nghĩa dựa trên các câu lệnh xử lý giao tác sau đây:

- BEGIN TRANSACTION: Bắt đầu một giao tác
- SAVE TRANSACTION: Đánh dấu một vị trí trong giao tác (gọi là điểm đánh dấu).
- ROLLBACK TRANSACTION: Quay lui trở lại đầu giao tác hoặc một điểm đánh dấu trước đó trong giao tác.
- COMMIT TRANSACTION: Đánh dấu điểm kết thúc một giao tác. Khi câu lệnh này thực thi cũng có nghĩa là giao tác đã thực hiện thành công.
- ROLLBACK [WORK]: Quay lui trở lại đầu giao tác.
- COMMIT [WORK]: Đánh dấu kết thúc giao tác.

Một giao tác trong SQL được bắt đầu bởi câu lệnh BEGIN TRANSACTION. Câu lệnh này đánh dấu điểm bắt đầu của một giao tác và có cú pháp như sau:

```
BEGIN TRANSACTION  [tên_giao_tác]
```

Một giao tác sẽ kết thúc trong các trường hợp sau:

- Câu lệnh COMMIT TRANSACTION (hoặc COMMIT WORK) được thực thi. Câu lệnh này báo hiệu sự kết thúc thành công của một giao tác. Sau câu lệnh này, một giao tác mới sẽ được bắt đầu.
- Khi câu lệnh ROLLBACK TRANSACTION (hoặc ROLLBACK WORK) được thực thi để huỷ bỏ một giao tác và đưa cơ sở dữ liệu về trạng thái như trước khi giao tác bắt đầu. Một giao tác mới sẽ bắt đầu sau khi câu lệnh ROLLBACK được thực thi.
- Một giao tác cũng sẽ kết thúc nếu trong quá trình thực hiện gặp lỗi (chẳng hạn hệ thống gặp lỗi, kết nối mạng bị "đứt",...). Trong trường hợp này, hệ thống sẽ tự động phục hồi lại trạng thái cơ sở dữ liệu như trước khi giao tác bắt đầu (tương tự như khi câu lệnh ROLLBACK được thực thi để huỷ bỏ mộtgiao tác). Tuy nhiên, trong trường hợp này sẽ không có giao tác mới được bắtđầu.

Ví dụ 6.1: Giao tác dưới đây kết thúc do lệnh ROLLBACK TRANSACTION và mọi thay đổi về mặt dữ liệu mà giao tác đã thực hiện (UPDATE) đều không có tác dụng.

```
BEGIN TRANSACTION giaotac1
UPDATE monhoc SET sodvht=4 WHERE sodvht=3
UPDATE diemthi SET diemlan2=0 WHERE diemlan2 IS NULL
ROLLBACK TRANSACTION giaotac1
```

còn giao tác dưới đây kết thúc bởi lệnh COMMIT và thực hiện thành công việc cập nhật dữ liệu trên các bảng MONHOC và DIEMTHI.

```
BEGIN TRANSACTION giaotac2
UPDATE monhoc SET sodvht=4 WHERE sodvht=3
UPDATE diemthi SET diemlan2=0 WHERE diemlan2 IS NULL
COMMIT TRANSACTION giaotac2
```

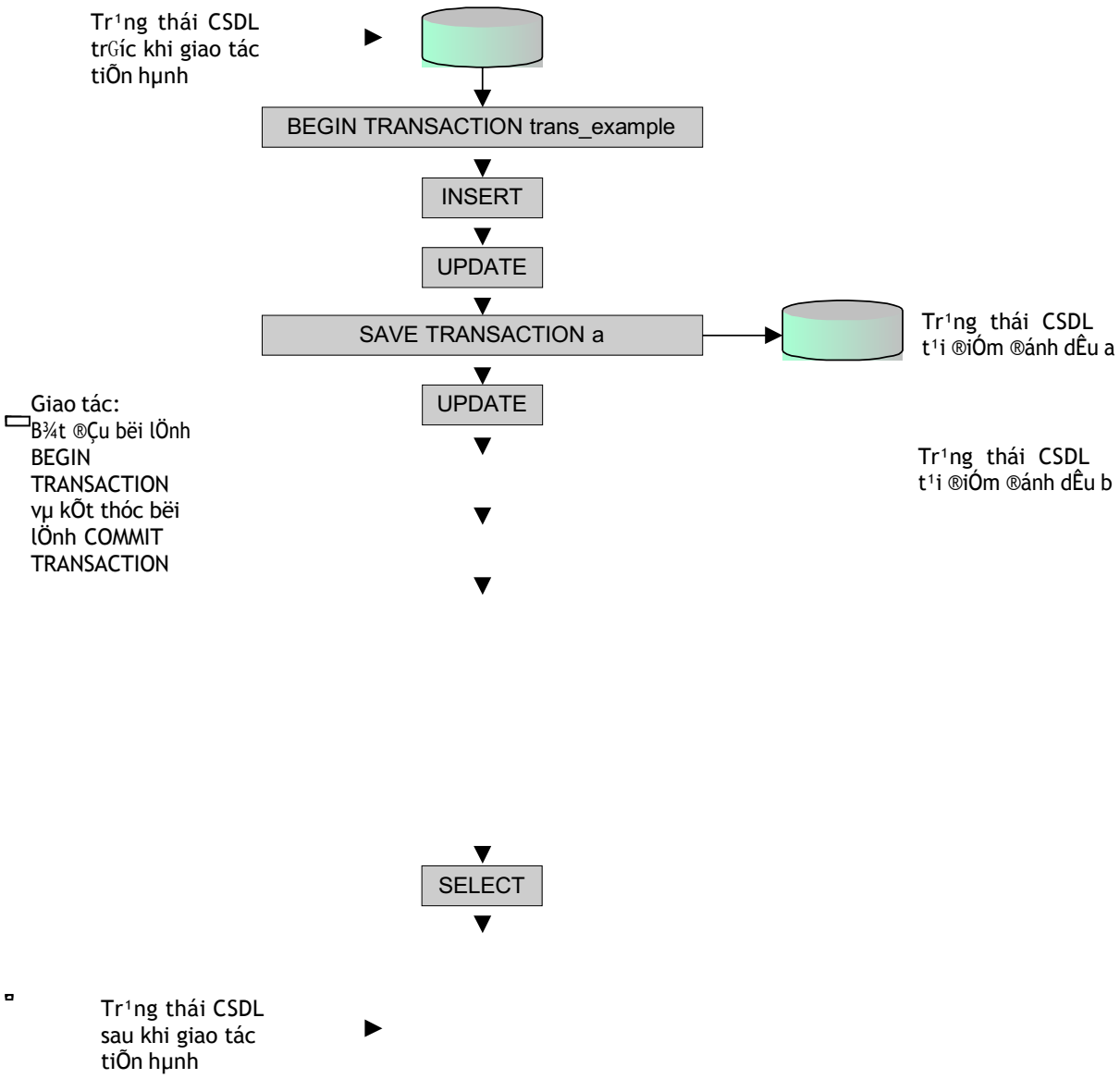
Câu lệnh:

```
SAVE TRANSACTION tên_điểm_đánh_dấu
```

được sử dụng để đánh dấu một vị trí trong giao tác. Khi câu lệnh này được thực thi, trạng thái của cơ sở dữ liệu tại thời điểm đó sẽ được ghi lại trong nhật ký giao tác. Trong quá trình thực thi giao tác có thể quay trở lại một điểm đánh dấu bằng cách sử dụng câu lệnh:

```
ROLLBACK TRANSACTION tên_điểm_đánh_dấu
```

Trong trường hợp này, những thay đổi về mặt dữ liệu mà giao tác đã thực hiện từ điểm đánh dấu đến trước khi câu lệnh ROLLBACK được triệu gọi sẽ bị huỷ bỏ. Giao tác sẽ được tiếp tục với trạng thái cơ sở dữ liệu có được tại điểm đánh dấu . Hình 6.2 mô tả cho ta thấy hoạt động của một giao tác có sử dụng các điểm đánh dấu:



Hình 6.2: *Hoạt động của một giao tác*

Sau khi câu lệnh ROLLBACK TRANSACTION được sử dụng để quay lui lại một điểm đánh dấu trong giao tác, giao tác vẫn được tiếp tục với các câu lệnh sau đó. Nhưng nếu câu lệnh này được sử dụng để quay lui lại đầu giao tác (tức là huỷ bỏ giao tác), giao tác sẽ kết thúc và do đó câu lệnh COMMIT TRANSACTION trong trường hợp này sẽ gặp lỗi.

Ví dụ 6.2: Câu lệnh COMMIT TRANSACTION trong giao tác dưới đây kết thúc thành công một giao tác

```
BEGIN TRANSACTION giaotac3
UPDATE diemthi SET diemlan2=0 WHERE diemlan2 IS NULL
```


5

```
SAVE TRANSACTION a
UPDATE monhoc SET sodvht=4 WHERE sodvht=3
ROLLBACK TRANSACTION a
UPDATE monhoc SET sodvht=2 WHERE sodvht=3
COMMIT TRANSACTION giaotac3
```

và trong ví dụ dưới đây, câu lệnh COMMIT TRANSACTION gặp lỗi:

```
BEGIN TRANSACTION giaotac4
UPDATE diemthi SET diemlan2=0 WHERE diemlan2 IS NULL
SAVE TRANSACTION a
UPDATE monhoc SET sodvht=4 WHERE sodvht=3
ROLLBACK TRANSACTION giaotac4
UPDATE monhoc SET sodvht=2 WHERE sodvht=3
COMMIT TRANSACTION giaotac4
```

6.3 Giao tác lồng nhau

Các giao tác trong SQL có thể được lồng vào nhau theo từng cấp. Điều này thường gặp đối với các giao tác trong các thủ tục lưu trữ được gọi hoặc từ một tiếntrình trong một giao tác khác.

Ví dụ dưới đây minh hoạ cho ta trường hợp các giao tác lồng nhau.

Ví dụ 6.3: Ta định nghĩa bảng T như sau:

```
CREATE TABLE T
(
     A    INT  PRIMARY KEY,
     B    INT
)
```

và thủ tục **sp_TransEx**:

```
CREATE PROC sp_TranEx(@a INT,@b INT)
AS
BEGIN
     BEGIN TRANSACTION T1
     IF NOT EXISTS (SELECT * FROM T WHERE A=@A )
          INSERT INTO T VALUES(@A,@B)
     IF NOT EXISTS (SELECT * FROM T WHERE A=@A+1)
          INSERT INTO T VALUES(@A+1,@B+1)
     COMMIT TRANSACTION T1
END
```

Lời gọi đến thủ tuch sp_TransEx được thực hiện trong một giao tác khác như sau:

```
BEGIN TRANSACTION T3
```

```
    EXECUTE sp_tranex
10,20ROLLBACK
TRANSACTION T3
```

Trong giao tác trên, câu lệnh `ROLLBACK TRANSACTION T3` huỷ bỏ giao tác và do đó tác dụng của lời gọi thủ tục trong giao tác không còn tác dụng, tức là không có dòng dữ liệu nào mới được bổ sung vào bảng T (cho dù giao tác T1 trong thủ tục *sp_tranex* đã thực hiện thành công với lệnh `COMMIT TRANSACTION T1`).

Ta xét tiếp một trường hợp của một giao tác khác trong đó có lời gọi đến thủ tục

sp_tranex như sau:

```
BEGIN TRANSACTION
EXECUTE sp_tranex
20,40SAVE TRANSACTION
a
EXECUTE sp_tranex 30,60
ROLLBACK TRANSACTION a
EXECUTE sp_tranex
40,80COMMIT
TRANSACTION
```

sau khi giao tác trên thực hiện xong, dữ liệu trong bảng T sẽ là:

A	B
20	40
21	41
40	80
41	81

Như vậy, tác dụng của lời gọi thủ tục `sp_tranex 30,60` trong giao tác đã bị huỷ bỏbởi câu lệnh `ROLLBACK TRANSACTION` trong giao tác.

Như đã thấy trong ví dụ trên, khi các giao tác SQL được lồng vào nhau, giao tácngoài cùng nhất là giao tác có vai trò quyết định. Nếu giao tác ngoài cùng nhất được uỷthác (commit) thì các giao tác được lồng bên trong cũng đồng thời uỷ thác; Và nếugiao tác ngoài cùng nhất thực hiện lệnh ROLLBACK thì những giao tác lồng bên trongcũng chịu tác động của câu lệnh này (cho dù những giao tác lồng bên trong đã thựchiện lệnh COMMIT TRANSACTION).