**TRANSACTION trong SQL** là tiến trình thực hiện một nhóm các câu lệnh SQL. Các câu lệnh này được thực thi một cách tuần tự và độc lập. Một Transaction được thực hiện thành công khi tất cả câu lệnh đều thành công, khi đó tất cả các thay đổi dữ liệu được thực hiện trong Transaction được lưu vào cơ sở dữ liệu. Tuy nhiên, nếu chỉ một trong số đó thất bại thì toàn bộ tiến trình sẽ thất bại, đồng nghĩa với việc dữ liệu phải rollback về trạng thái ban đầu (dữ liệu được khôi phục về trạng thái trước khi thực hiện Transaction).

**Thuộc tính của Transaction**

**1. Atomicity - Tính bảo toàn:**nguyên tắc "all or nothing", đảm bảo rằng tất cả các câu lệnh trong nhóm lệnh được thực thi thành công. Nếu không, Transaction bị hủy bỏ tại thời điểm thất bại và tất cả các thao tác trước đó được khôi phục về trạng thái cũ đồng nghĩa với việc không có gì thay đổi về mặt dữ liệu.

**2. Consistency - Tính nhất quán:** đảm bảo rằng cơ sở dữ liệu thay đổi chính xác các trạng thái khi một transaction được thực thi thành công.

**3. Isolation - Tính độc lập:** cho phép các Transaction hoạt động độc lập và minh bạch với nhau.

**4. Durability - Tính bền vững:** đảm bảo rằng kết quả của một transaction được xác định, không có chuyện dữ liệu của Transaction sau khi thực thi có thể chuyển lại trạng thái dữ liệu lúc trước khi thực hiện.

## Xử lý Transaction

Trong SQL, có các lệnh sau được sử dụng để điều khiển Transaction:

* **COMMIT:** để lưu các thay đổi.
* **ROLLBACK:** để quay trở lại trạng thái trước khi có thay đổi.
* **SAVEPOINT:** tạo các điểm (point) bên trong các nhóm Transaction để ROLLBACK, tức là để quay trở lại điểm trạng thái đó.
* **SET TRANSACTION:** đặt một tên cho một Transaction.

Các lệnh điều khiển Transaction chỉ được sử dụng với các lệnh thao tác dữ liệu như INSERT, UPDATE và DELETE. Tuy nhiên chúng không thể được sử dụng trong lệnh CREATE TABLE hoặc DROP TABLE vì các hoạt động này được tự động xác định trong cơ sở dữ liệu.

## Lệnh COMMIT trong SQL

Transaction kết thúc với một trong hai câu lệnh COMMIT hoặc ROLLBACK.

Khi một Transaction hoàn chỉnh được hoàn thành thì **lệnh COMMIT** phải được gọi ra. Đây là lệnh điều khiển Transaction được sử dụng để lưu các thay đổi gọi bởi một Transaction tới cơ sở dữ liệu.

Cú pháp cơ bản của lệnh COMMIT như sau:

COMMIT;

**Ví dụ:** Giả sử bảng NHANVIEN có các bản ghi như sau:

+----+----------+-----+-----------+----------+

| ID | TEN |TUOI | DIACHI | LUONG |

+----+----------+-----+-----------+----------+

| 1 | Thanh | 32 | Haiphong | 2000.00 |

| 2 | Loan | 25 | Hanoi | 1500.00 |

| 3 | Nga | 23 | Hanam | 2000.00 |

| 4 | Manh | 25 | Hue | 6500.00 |

| 5 | Huy | 27 | Hatinh | 8500.00 |

| 6 | Cao | 22 | HCM | 4500.00 |

| 7 | Lam | 24 | Hanoi | 10000.00 |

+----+----------+-----+-----------+----------+

Ví dụ sau sẽ xóa các bản ghi từ bảng mà có tuoi = 25 và sau đó COMMIT các thay đổi vào trong Database.

SQL> DELETE FROM NHANVIEN  
 WHERE TUOI = 25;  
SQL> COMMIT;

Vì vậy, hai hàng từ bảng sẽ bị xóa và câu lệnh SELECT sẽ cho kết quả sau.

+----+----------+-----+-----------+----------+

| ID | TEN |TUOI | DIACHI | LUONG |

+----+----------+-----+-----------+----------+

| 1 | Thanh | 32 | Haiphong | 2000.00 |

| 3 | Nga | 23 | Hanam | 2000.00 |

| 5 | Huy | 27 | Hatinh | 8500.00 |

| 6 | Cao | 22 | HCM | 4500.00 |

| 7 | Lam | 24 | Hanoi | 10000.00 |

+----+----------+-----+-----------+----------+

## Lệnh ROLLBACK trong SQL

**Lệnh ROLLBACK** là lệnh điều khiển Transaction được sử dụng để trao trả Transaction về trạng thái trước khi có các thay đổi mà chưa được lưu tới Database. Lệnh ROLLBACK chỉ có thể được sử dụng để undo các Transaction trước khi xác nhận bằng lệnh Commit hay Rollback cuối cùng.

Cú pháp cơ bản của lệnh ROLLBACK như sau:

ROLLBACK;

**Ví dụ:** Giả sử bảng NHANVIEN có các bản ghi như sau:

+----+----------+-----+-----------+----------+

| ID | TEN |TUOI | DIACHI | LUONG |

+----+----------+-----+-----------+----------+

| 1 | Thanh | 32 | Haiphong | 2000.00 |

| 2 | Loan | 25 | Hanoi | 1500.00 |

| 3 | Nga | 23 | Hanam | 2000.00 |

| 4 | Manh | 25 | Hue | 6500.00 |

| 5 | Huy | 27 | Hatinh | 8500.00 |

| 6 | Cao | 22 | HCM | 4500.00 |

| 7 | Lam | 24 | Hanoi | 10000.00 |

+----+----------+-----+-----------+----------+

Bây giờ sử dụng lệnh ROLLBACK với lệnh xóa tuoi = 25, mà chưa được commit như sau:

SQL> DELETE FROM NHANVIEN  
 WHERE TUOI = 25;  
SQL> ROLLBACK;

Trong kết quả thu được, hoạt động DELETE này không ảnh hưởng tới bảng vì đã ROLLBACK các thay đổi trong cơ sở dữ liệu, lệnh SELECT sẽ cho kết quả:

+----+----------+-----+-----------+----------+

| ID | TEN |TUOI | DIACHI | LUONG |

+----+----------+-----+-----------+----------+

| 1 | Thanh | 32 | Haiphong | 2000.00 |

| 2 | Loan | 25 | Hanoi | 1500.00 |

| 3 | Nga | 23 | Hanam | 2000.00 |

| 4 | Manh | 25 | Hue | 6500.00 |

| 5 | Huy | 27 | Hatinh | 8500.00 |

| 6 | Cao | 22 | HCM | 4500.00 |

| 7 | Lam | 24 | Hanoi | 10000.00 |

+----+----------+-----+-----------+----------+

## Lệnh SAVEPOINT trong SQL

**SAVEPOINT** là một điểm trong một Transaction giúp bạn có thể lùi Transaction trở lại một điểm nhất định mà không cần lùi Transaction về trạng thái đầu trước khi có thay đổi đó.

Cú pháp cơ bản của lệnh SAVEPOINT như sau:

SAVEPOINT TEN\_SAVEPOINT;

Lệnh này chỉ tạo ra SAVEPOINT trong các câu lệnh Transaction. Sau đó ROLLBACK cần được sử dụng để hoàn tác trở lại một SAVEPOINT như sau:

ROLLBACK TO TEN\_SAVEPOINT;

**Ví dụ:** Bạn muốn xóa ba bản ghi khác nhau từ bảng NHANVIEN và muốn tạo SAVEPOINT trước mỗi lần xoá để có thể ROLLBACK trở lại SAVEPOINT bất kỳ lúc nào giúp trả lại dữ liệu thích hợp cho trạng thái ban đầu.

Giả sử bảng NHANVIEN có các bản ghi như sau:

+----+----------+-----+-----------+----------+

| ID | TEN |TUOI | DIACHI | LUONG |

+----+----------+-----+-----------+----------+

| 1 | Thanh | 32 | Haiphong | 2000.00 |

| 2 | Loan | 25 | Hanoi | 1500.00 |

| 3 | Nga | 23 | Hanam | 2000.00 |

| 4 | Manh | 25 | Hue | 6500.00 |

| 5 | Huy | 27 | Hatinh | 8500.00 |

| 6 | Cao | 22 | HCM | 4500.00 |

| 7 | Lam | 24 | Hanoi | 10000.00 |

+----+----------+-----+-----------+----------+

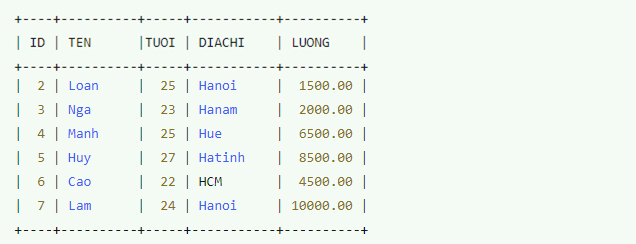
Dưới đây là một chuỗi các câu lệnh:

SQL> SAVEPOINT SP1;  
Savepoint created.  
SQL> DELETE FROM NHANVIEN WHERE ID=1;  
1 row deleted.  
SQL> SAVEPOINT SP2;  
Savepoint created.  
SQL> DELETE FROM NHANVIEN WHERE ID=2;  
1 row deleted.  
SQL> SAVEPOINT SP3;  
Savepoint created.  
SQL> DELETE FROM NHANVIEN WHERE ID=3;  
1 row deleted.

Ở trên, có 3 hoạt động xóa dữ liệu diễn ra. Giả sử bạn thay đổi suy nghĩ và quyết định ROLLBACK về SAVEPOINT mà bạn đã xác định là SP2. Bởi vì SP2 được tạo sau hoạt động xóa đầu tiên, do đó hai hoạt động xóa cuối cùng được khôi phục lại.

SQL> ROLLBACK TO SP2;  
Rollback complete.

Vì vậy chỉ có hoạt động xóa đầu tiên diễn ra sau khi bạn ROLLBACK về SP2.



### **Lệnh RELEASE SAVEPOINT trong SQL**

**Lệnh SAVEPOINT RELEASE** được sử dụng để loại bỏ một SAVEPOINT mà bạn đã tạo ra. Khi SAVEPOINT bị xóa, bạn không thể sử dụng lệnh ROLLBACK để hoàn tác các Transaction về SAVEPOINT đó nữa.

Cú pháp của SAVEPOINT RELEASE như sau:

RELEASE SAVEPOINT TEN\_SAVEPOINT;

## Lệnh SET TRANSACTION trong SQL

**SET TRANSACTION**có thể được sử dụng để khởi tạo một Database Transaction. Lệnh này được sử dụng để chỉ định các đặc tính cho Transaction đó. Ví dụ, bạn có thể chỉ định một Transaction chỉ được đọc (read only) hoặc đọc viết (read write).

Cú pháp cơ bản của lệnh SET TRANSACTION trong SQL như sau:

SET TRANSACTION [ READ WRITE | READ ONLY ];

Như chúng ta đã từng biết một trong các chức năng chính của hệ QTCSDL là phải đảm bảo tính tin cậy, tính nhất quán của CSDL, ngay cả khi có nhiều người dùng đồng thời truy cập vào CSDL hay thậm chí ngay cả khi xảy ra sự cố (phần cứng hay phần mềm). Trong bài viết này chúng ta sẽ cùng nhau đi tìm hiểu ba dịch vụ mà một hệ QTCSDL cung cấp: hỗ trợ giao tác, điều khiển tương tranh và khôi phục CSDL, những dịch vụ đó phụ thuộc vào nhau. Điều khiển tương tranh và khôi phục CSDL mỗi khi có sự cố , cả hai đều nhằm đảm bảo sự nhất quán về dữ liệu và tránh tổn thất thông tin.

# I. CÁC KHÁI NIỆM CƠ BẢN VỀ HỆ THỐNG CHO KHÔI PHỤC VÀ ĐIỀU KHIỂN TƯƠNG TRANH

## 1. Khái niệm giao tác:

**Một giao tác (transaction)** là một hành động hay một chuỗi các hành động truy cập vào CSDL hoặc làm thay đổi nội dung CSDL, giao tác được đưa ra bởi một người sử dụng hay một chương trình ứng dụng

Như vậy một giao tác là một đơn vị công việc trên CSDL xét trên tính logic. Một giao tác có thể là toàn bộ một chương trình (viết trong ngôn ngữ thao tác dữ liệu bậc cao hay trong ngôn ngữ lập trình như SQL, COBOL,...), nó cũng có thể là một phần của chương trình hay một câu lệnh đơn lẻ như câu lệnh INSERT, UPDATE của SQL.

Trong ngữ cảnh CSDL, có thể xem sự thực hiện một chương trình ứng dụng như một dãy các **giao tác** và xen kẽ giữa chúng là những thao tác không đòi hỏi xử lí CSDL (thao tác CPU). Để cho việc tiện theo dõi, mình sẽ minh họa hai lược đồ quan hệ **Products** và **Categories** đơn giản như sau:

**Products**(id, category\_id, name, price)

**Categories**(id, name)

**Giải thích:**

* Lược đồ **Products** gồm các thuộc tính về mã sản phẩm (**id**), mã thể loại (**category\_id**), tên sản phẩm (**name**) và giá của sản phẩm (**price**)
* Lược đồ **Categories** dùng để tham chiếu mã thể loại ứng với từng sản phẩm bao gồm các thuộc tính về mã thể loại (**id**), tên thể loại (**name**)

**Ví dụ 1:** Một giao tác đơn giản là nâng giá của một sản phẩm có mã số là x*x* thêm 10$%$ giá hiện tại của sản phẩm đó. Chúng ta có thể viết giao tác này như sau:

read(product.id = x, price)

new\_price = price \* 1.1

write(product.id = x, new\_price)

Ở đây chúng ta quy ước dùng **read(x)** để chỉ thao tác đọc một mục dữ liệu có tên là x*x* ở CSDL vào một biến của chương trình. Để cho đơn giản việc trình bày chúng ta coi như biến chương trình này cũng có tên là x*x*. Tương tự như vậy **write(x)** dùng để chỉ thao tác ghi giá trị biến x*x* của chương trình vào mục dữ liệu x*x* của CSDL

Việc thưc hiện một lệnh **read(x)** sẽ bao gồm các bước sau:

* Tìm địa chỉ của khối có chứa mục dữ liệu x*x*
* Sao chép khối đó vào vùng đệm của bộ nhớ chính (nếu nó chưa có sẵn trong một vùng đệm của bộ nhớ chính)
* Sao chép mục dữ liệu x*x* từ vùng đệm vào biến có tên là x*x* của chương trình

Việc thưc hiện một lệnh **write(x)** sẽ bao gồm các bước sau:

* Tìm địa chỉ của khối có chứa mục dữ liệu x*x*
* Sao chép khối đó vào vùng đệm của bộ nhớ chính (nếu nó chưa có sẵn trong một vùng đệm của bộ nhớ chính)
* Sao chép mục dữ liệu x*x* từ biến chương trình có tên là x*x* vào chỗ chính xác của nó trong vùng đệm
* Lưu trữ khối vừa cập nhập trong vùng đệm vào lại thiết bị nhớ để sau này có thể truy xuất (có thể lưu ngay tức thời hoặc tại thời điểm muộn hơn)

Trong ví dụ trên lệnh **read(**[**product.id**](http://product.id/)**= x, price)** dùng để chỉ thao tác đọc dữ liệu ở thuộc tính ***price*** của bản ghi có giá trị khóa chính id=x, lệnh **write(**[**product.id**](http://product.id/)**= x, new\_price)** dùng để chỉ giao tác ghi lại giá trị của **new\_price** vào cho bản ghi có khóa chính là x. Giữa hai thao tác đọc và ghi dữ liệu nới trên có một thao tác không truy cập CSDL, đó là thao tác new\_price = price \* 1.1

**Ví dụ 2:** Hãy xem xét một ví dụ phức tạp hơn.

Giả sử có một giao tác xóa bỏ bản ghi về **Categories** có mã thể loại id = x*id*=*x* khỏi quan hệ trong **ví dụ 1**. Do ràng buộc tham chiếu từ lược đồ **Products** đến lược đồ **Categories**, để có thể xóa được bản ghi trong **Categories** thì trong **Products** ta cần gán lại một giá trị khác chẳng hạn là y (y khác x) cho tất cả các bản ghi mà có **category\_id = x**. Giao tác này được trình bày như sau:

Delete(Categories.id = x)

begin

read(products.category\_id = x)

if (products.category\_id = x) then begin

products.category\_id = y

write(category\_id = y)

end

end

Ở đây nếu việc cập nhập lại cho các bản ghi tương ứng trong lược đồ **Products** không được tiến hành thì CSDL sẽ rơi vào trang thái không nhất quán, vi phạm ràng buộc tham chiếu vì lúc này một số bản ghi product sẽ nói về một category không tồn tại trong CSDL.

Một giao tác phải chuyển CSDL từ trạng thái nhất quán này sang trạng thái nhất quán khác, mặc dù trong quá trình thực hiện giao tác, có những thời điểm CSDL tạm thời ở trạng thái không nhất quán.

Rất có thể trong quá trình thực hiện dãy hành động của một giao tác, một sự cố nào đó xảy ra và việc thực hiện giao tác không thành công. Như vậy, khi thực hiện một giao tác, chúng ta có một trong hai kết quả: giao tác được **thực hiện thành công** hoặc **thực hiện không thành công**.

* Trong trường hợp thứ nhất chúng ta nói là giao tác đã được **chuyển giao (committed)**, CSDL đã được chuyển sang một trạng thái mới, nhất quán.
* Trong trường hợp thứ hai chúng ta nói rằng giao tác bị **hủy bỏ (aborted)**, khi giao tác bị hủy bỏ thì CSDL phải được **khôi phục (trả về)** trạng thái nhất quán của nó trước khi bắt đầu thực hiện giao tác làm cho nó bị hủy. Những giao tác bị hủy bỏ như vậy được gọi là những giao tác bị **phục hồi (rollback)**

Một giao tác đã được chuyển giao sẽ không thể bị hủy bỏ. Nếu chúng ta nhận thấy giao tác này đã thực hiện một sai sót thì chỉ còn một cách là thực hiện một giao tác khác điều chỉnh lại gọi là giao tác bù, hủy đi tác động của các giao tác trước đó đã gây ra sai sót.

Các hệ QTCSDL không thể tự hiểu được rằng những phép cập nhập nào sẽ được nhóm lại thành một đơn vị logic công việc hay một giao tác-transaction, bởi vậy người dùng phải tự định biên cho một giao tác bằng các từ khóa **BEGIN TRANSACTION**, **COMMIT**, **ROLLBACK** theo quy định của các ngôn ngữ DML. Trong trường hợp không có sự định biên cho các giao tác thì hệ thống sẽ coi toàn bộ chương trình là một giao tác, hệ QTCSDL sẽ tự động thực hiện **COMMIT** khi chương trình thực hiện không có một lỗi nào xuất hiện và **ROLLBACK** trong tình huống ngược lại.

## 2. Các tính chất của giao tác:

Để đảm bảo tính nhất quán của dữ liệu, các hệ QTCSDL cần đảm bảo các tính chất sau đây cho giao tác, tính chất **ACID**

* **Tính nguyên tố (Atomicity):** toàn bộ các thao tác trong cùng một giao tác đều được thực hiện hoặc không một thao tác nào trong chúng được thực hiện. Một giao tác là một đơn vị công việc không thể phân chia
* **Tính nhất quán (Consistency):** một giao tác phải chuyển CSDL từ một trạng thái nhất quán này sang một trạng thái nhất quán khác
* **Tính cô lập (Isolation):** Các giao tác phải được thực hiện một cách độc lập với nhau. Nói một cách khác những tác động của giao tác này sẽ không thể thấy được đối với những giao tác khác, khi giao tác này chưa thực hiện xong hoàn toàn
* **Tính bền vững (Durability):** những thay đổi của CSDL do tác động của một giao tác thành công là bền vững, không bị mất đi

mở rộng hơn một tí: chúng ta sẽ để cập đến kiến trúc của một hệ CSDL, ở trong kiến trúc này bộ quản lí giao tác (**transaction manager**) thay mặt các chương trình ứng dụng điều phối các giao tác. Nó liên hệ với bộ lập lịch, nơi có trách nhiệm thực hiện một chiến lược điều khiển tương tranh. Nếu có một sự cố trong quá trình thực hiện giao tác thì tính nhất quán của CSDL cần được bảo toàn, đây là trách nhiệm của bộ quản lí khôi phục dữ liệu. Bộ quản lí vùng nhớ đệm có nhiệm vụ chuyển giao dữ liệu qua lại giữ các thiết bị nhớ thứ cấp và bộ nhớ chính.