**Stored procedure và trigger trong mysql**  
**A.Stored procedure in**[**MySQL**](http://thegioilaptrinh.net/viewforum.php?f=9)  
**I.Giới thiệu Stored procedure**  
  
**1. Định nghĩa**  
  
Stored procedure được định nghĩa như một tập các khai báo SQL được viết bởi người dùng, được lưu trữ ngay trong cơ sở dữ liệu (database) và sau đó được triệu gọi bởi một program, một trigger hay thậm chí là một Stored procedure khác.  
Một Stored procedure có tính đệ qui. Msql không hỗ trợ tốt tính năng đệ qui nên phải từ Mysql 5.0 trở lên mới hỗ trợ Stored procedure.  
  
**2. Phân loại**  
  
Stored procedure có thể chia thành 3 nhóm như sau:  
System Stored Procedure: Là những Stored procedure chứa trong Matster database và thường bắt đầu bằng tiếp đầu ngữ sp\_. Các Stored procedure này thuộc loại built-in và chủ yếu dùng trong việc qunr lý database và security.  
Ví dụ: bạn có thể kiểm tra tất cả các processes đang được sử dụng bởi user DomainName\Administrators bạn có thể dung:

Mã:

Sp\_who @loginame=’DomainName\Administrator’&#46;

· Local Stored procedure: đây là loại thường dùng nhất. Chúng được chứa trong user database và thường được viết để thực hiện một công việc nào đó. Thông thường người ta nói đến Stored procedure là nói tới loại này. Loại này thường được viết bởi DBA hoặc programmer.  
· Extended Stored procedure: đây là một loại Stored procedure sử dụng một chương trình ngoại vi vốn được compiled thành một DLL để mở rộng chức năng hoạt động của SQL Server. Loại này thương được bắt đầu bằng tiếp ngữ xp\_.  
Ví dụ: xp\_sendmail dung để gửi mail cho một người nào đó.  
  
**II.Chương trình Stored procedure**  
  
**1.Chương trình Stored procedure đầu tiên**  
  
Stored procedure đầu tiên khá đơn gian. Nó chỉ có chức năng lấy thông tin về tất cả các sản phẩm được lưu trong bảng products.  
Ví dụ:

Mã:

DELIMITER //

CREATE PROCEDURE GetAllProducts()

BEGIN

SELECT \* FROM products;

END//

DELIMITER;

Dòng đầu tiên ta thấy là DELIMITER //. Dòng lệnh này không liên quan tới Stored procedure. Nó được dùng để thay đổi cú pháp phân cách chuẩn từ dấu “;” thành dấu khác.  
Sau từ khóa END bạn lại sử dụng dấu phân cách // để ra hiệu cho trình biên dịch biết là đã hết stored procedure. Dòng cuối cùng đưa cú pháp phân cách trở về trạnh thái nguyên, tức dấu “;”.  
  
**2.Khai báo một Stored procedure mới**  
  
Để tạo 1 stored procedure mới, bạn sử dụng phát biểu CREATE PROCEDURE. Sau phát biểu CREATE PROCEDURE là tên của stored procedure mà bạn muốn đặt.  
Phần thân của stored procedure được bắt đầu bởi cặp từ khóa BEGIN và END. Các câu lệnh SQL được đặt trong block này. Chúng ta sẽ phân tích chi tiết trong các phần sau. Đến đây, bạn đã hoàn tất việc tạo 1 stored procedure.

Mã:

CREATE PROCEDURE sp\_name(danh\_sach\_tham\_so)

BEGIN

END

Nếu có tham số thì phải khai báo trong dấu ngoặc đơn còn không thì bỏ trống (), mặc định của tham số là IN.  
- Tên thủ tục: phải theo qui tắc định danh và không vượt quá 128 ký tự.  
- Danh sách tham số: khai báo mỗi tham số tối thiểu phải bao gồm 2 phần:  
  
Tên tham số bắt đầu bởi dấu @  
Kiểu dữ liệu của tham số  
  
Vi dụ:

Mã:

CREATE PROCEDURE p

(IN thamso integer)

BEGIN

DECLARE bien1 CHAR (20);

IF thamso = 27 THEN

SET bien1 = ‘hoa hong’;

ELSE

SET bien1= ‘hoa’;

END IF;

INSERT INTO bang1 VALUES (bien1)

END

**3. Triệu gọi Stored procedure**  
  
Để triệu gọi 1 stored procedure, chúng ta sử dụng câu lệnh SQL sau:

Mã:

CALL STORED\_PROCEDURE\_NAME()

Ví dụ:

Mã:

Call p

**III. Biến trong Stored procedure**  
  
**1. Khai báo biến**  
  
Biến trong Stored procedure được sử dụng để lưu trữ kết quả tức thời  
Cú pháp khai báo biến:

Mã:

DECLARE variable\_name datatype(size) DEFAULT default\_value;

Trong đó:  
  
Ngay sau từ khóa DECLARE là tên biến, và tên biến phải tuân theo qui ước đặt tên biến. Tên biến không nên trùng với tên của table hay colume trong database.  
Kiểu dữ liệu của biến, đó là một trong những kiểu dữ liệu mà MySQL hỗ trợ sẵn: Int, varchar, datetime…  
Kích cỡ của biến  
Giái trị: khi bạn khai báo một biến nó tự động khởi tạo giá trị mặc định là null. Bạn có thể thay đổi giá trị mặc định cho biến bằng cách sử dụng DEFAULT.  
  
Ví dụ

Mã:

DECLARE tong\_tien INT DEFAULT 0

**Chú ý:**Có thể dùng 1 từ khóa DECLARE để khai báo 2 hay nhiều biến có cùng kiểu dữ liệu:

Mã:

DECLARE x, y INT DEFAULT 1

Hai biến x và y với cùng kiểu dữ liệu int, giá trị mặc định (của cả 2) là 1.  
  
**2. Gán giá trị cho một biến**  
Sau khi khai báo biến, bạn có thể sử dụng chúng. Để gán 1 giá trị cho 1 biến, bạn dùng phát biểu SET như sau:

Mã:

DECLARE tong\_tien INT DEFAULT 0

SET tong\_tien = 10;

Biến tong\_tien lúc này có giá trị là 10.  
Ngoài cách dùng phát biểu SET, chúng ta cũng có thể sử dụng SELECT … INTO để gán kết quả của 1 truy vấn cho 1 biến:

Mã:

DECLARE tong\_tien INT DEFAULT 0

SELECT COUNT INTO tong\_tien

FROM Luong

Ở ví dụ trên, chúng ta khai báo 1 biến tong\_tienvà khởi tạo giá trị ban đầu bằng 0, sau đó sử dụng phát biểu SELECT … INTO để gán giá trị cho biến tong\_tien bằng tổng số tiền trong bảng Luong  
  
**3.Phạm vi của một biến**  
  
Biến có phạm vi hoạt động khác nhau, nó phụ thuộc vào vị trí ta khai báo biến:  
- Nếu khai báo biến bên trong 1 Stored procedure, biến đó sẽ hoạt động cho tới khi gặp từ khóa END của Stored procedure  
- Nếu bạn khai báo biến trong block BEGIN/END của Stored procedure, biens đó sẽ hoạt động trong phạm vi riêng của chúng. (khuyến cáo không nên làm).  
- Mỗi biến với ký tự “@” làm tiền tố được xem là biến thuộc phạm vi session, nó tồn tại khi nào session còn tồn tại.  
  
**IV. Parameters trong Stored Procedures**  
  
Hầu hết các Stored procedure đều có tham số. Tham số làm cho Stored procedure trở nên linh động và hữu dụng hơn. Trong MySQL, một tham số thuộc một trong 3 kiểu sau: IN, OUT và INOUT  
- IN: tham số vẫn giữ nguyên giá trị của nó sau khi ra khỏi Stored procedure  
- Out: cho phép Stored procedure thay đổi giá trị của tham số và trả nó về chương trình gọi.  
- INOUT: kết hợp cả hai IN và OUT. Bạn có thể truyền một tham số dạng này vào Stored procedure và nhận lại giá trị mới.  
  
**v Cú pháp khai báo tham số trong Stored procedure**  
  
**MODE param\_name param\_type(param\_size)**  
  
Trong đó:  
- Param\_name: tên của tham số, nó không nên trùng với tên của column hay table và phải tuân theo quy tắc đặt tên cho tham số.  
- Param\_type: kiểu của tham số  
- Param\_size: kích cỡ của tham số  
Mỗi param được ngăn cách bởi dấu “,” nếu stored procedure có nhiều hơn 1 param.  
Ví dụ:  
- Sử dụng tham số IN  
  
Giả sử ta có bảng nhanvien (MA\_id, tên, tuổi)  
Muốn lấy ra các nhân viên có tuổi lớn hơn bao nhiêu đó khi đó câu lệnh procedure như sau:

Mã:

DROP PROCEDURE IF EXISTS ‘test1’;

DELIMITER;;

CREATE DEFINER=’root’ @’%’ PROCEDURE ‘test1’(int age int)

BEGIN

SELECT \* FROM nhanvien WHERE tuoi>age;

END;;

DELIMITER;

Khi đó lệnh gọi procedure để lấy ra các nhân viên có tuổi lớn hơn 25 là:  
Call test1(25)  
  
- Sử dụng tham số OUT  
  
Lấy ra tổng số nhân viên có tuổi>một giá trị nào đấy

Mã:

DROP PROCEDURE IF EXISTS ‘test1’;

DELIMITER;;

CREATE DEFINER=’root’ @’%’ PROCEDURE ‘test1’(int age int, count rows\_count)

BEGIN

SELECT \* FROM nhanvien WHERE tuoi>age;

SET rows\_count = FOUND\_ROWS();

END;;

DELIMITER;

Khi đó lệnh gọi procedure để lấy ra tổng số nhân viên có tuổi lớn hơn 25 là  
Call test1(25, @rows\_count)  
Select @rows\_count  
  
**V. Các mệnh đề điều kiện trong Stored procedures**  
  
Mệnh đề điều kiện cho phép bạn thực thi các đoạn mã dựa trên giá trị của một hoặc nhiều biểu thức logic kết hợp. MySQL hỗ trợ 2 mệnh đề điều kiện là IF và CASE.  
  
**1.Mệnh đề IF**  
  
Cú pháp của mệnh đề IF:

Mã:

IF expression THEN commands

[ELSEIF expression THEN commands]

[ELSE commands]

END IF;

Các lệnh nằm trong khối IF, ELSEIF hoặc ELSE chỉ được thực thi khi mệnh đề có giá trị là TRUE. Một trong những cái "bẫy" hay mắc phải là biểu thức của mệnh đề IF có giá trị NULL, khi 1 biểu thức logic có giá trị NULL, nó không được xem là TRUE hay FALSE.  
  
Ví dụ

Mã:

CREATE PROCEDURE p

(IN thamso integer)

BEGIN

DECLARE bien1 CHAR (20);

IF thamso = 27 THEN

SET bien1 = ‘hoa hong’;

ELSE

SET bien1= ‘hoa’;

END IF;

INSERT INTO bang1 VALUES (bien1)

END

**2. Mệnh đề CASE**  
  
Khi có nhiều biểu thức logic được sử dụng với mệnh đề IF, mã chương trình của bạn sẽ rất khó đọc. Trong tình huống này, bạn có thể sử dụng mệnh đề CASE để làm cho chương trình trông sáng sủa hơn.  
  
Cú pháp của mệnh đề CASE:

Mã:

CASE

WHEN expression THEN commands

…

WHEN expression THEN commands

ELSE commands

END CASE;

**VI. Vòng lặp trong Stored procedure**  
**1.vòng lặp While**  
  
Cú pháp của vòng lặp while:

Mã:

WHILE expression DO

Statements

END WHILE

Đầu tiên, vòng lặp while kiểm tra biểu thức điều kiện, nếu điều kiện có giá trị true thì chương trình sẽ thực thi các lệnh trong vòng lặp. Vòng lặp while kiểm tra biểu thức trước khi thực thi các dòng lệnh.  
  
**2. Vòng lặp Reprat**  
Cú pháp của vòng lặp repeat:

Mã:

REPEAT

Statement;

UNTIL expression

END REPEAT

Dòng lệnh đầu tiên được thực thi, sau đó chương trình mới kiểm tra tới biểu thức logic. Nếu biểu thức logic có giá trị true, các dòng lệnh bên trong sẽ được lập lại liên tục cho đến khi biểu thức logic có giá trị false. Vì vòng lặp repeat kiểm tra biểu thức logic sau khi đã thực thi 1 lần dòng lệnh nên repeat được biết như vòng lặp tiền kiểm tra (post-test loop).  
  
**3. Vòng lặp loop, leave và iterate**  
  
Phát biểu leave cho phép bạn rời vòng lặp. Nó gần giống với phát biểu break trong các ngôn ngữ khác như Java, C#...  
Phát biểu Iterate cho phép bạn bắt đầu vòng lặp trở lại. Nó gần giống như phát biểu continue trong Java hay C#.  
MySql cũng hỗ trợ vòng lặp loop cho phép bạn thực thi dòng lệnh lặp 1 cách cơ động.  
  
**VII. Ưu điểm và nhược điểm của Stored procedure**  
  
Stored procedure có những thuận tiện cũng như những bất lợi. Khi phát triển ứng dụng, bạn nên đánh giá giữa ưu và nhược điểm để quyết định có nên sử dụng Stored procedure hay không.  
  
**1.Ưu điểm**  
  
· Stored procedure làm tăng khả năng thực thi của ứng dụng. Sau khi được tạo, Stored procedure sẽ được biên dịch và lưu trữ ngay trong database. Nên nó sẽ chạy nhanh hơn một lệnh SQL chưa biên dịch gửi trực tiếp từ ứng dụng.  
· Stored procedure làm giảm lưu lượng thong tin giao tiếp giữa ứng dụng và database server.  
· Stored procedure có thể tái sử dụng và chuyển sang bất cứ ứng dụng naog muốn sử dụng chúng.  
· Stored procedure cũng rất an toàn. Ví dụ trong ngân hang sử dụng Stored procedure và các function cho tất cả các tác vụ chung. Việc này cung cấp môi trường an toàn cao. Theo đó, các ứng dụng và người dung không thể truy cập trực tiếp vào các bảng dữ liệu nhưng có thể thực thi các Stored procedure xác định.  
  
**2. Nhược điểm của Stored procedure**  
  
· Stored procedure làm cho database server phải tốn nhiều tài nguyên về cả bộ nhớ lẫn xử lý.  
· Stored procedure chỉ chứa đựng các khai báo SQL, vì vậy rất khó có thể viết một procedure nhằm thực hiện các thao tác xử lý phức tạp như các ngôn ngữ khác làm được ở tầng ứng dụng như C#, Java, C++…  
· Bạn cũng không thể debug Stored procedure trong hầu hết các RDMBS và trong cả MySQL.  
· Việc viết bảo trì Stored procedure thường yêu cầu một loạt các kỹ năng chuyên biệt, nhiều khi không phải là kỹ năng của lập trình viên. Điều này dẫn đến các vấn đề trong cả khía cạnh phát triển ứng dụng và bảo trì sản phẩm.  
  
**B. Trigger in MySQL**  
  
**I.Trigger là gì**  
Trigger là một đối tượng được định danh trong CSDL và được gắn chặt với một sự kiện xảy ra trên một bảng nào đó ( điều này có nghĩa là nó sẽ được tự động thực thi khi xảy ra một sự kiện trên một bảng). Các sự kiện này bao gồm: chèn (Insert), xóa (Delete) hay cập nhật (Update) một bảng.  
  
**II.Tại sao lại sử dụng Trigger?**  
  
Trigger được thực thi tự động khi xuất hiện một hành động thay đổi trong bảng, nên người ta có thể ứng dụng Trigger để tạo ra các công việc tự động thay cho việc phải làm thủ công bằng tay như: kiểm tra dữ liệu, đồng bộ hóa dữ liệu, đảm bảo các mối quan hệ giữa các bảng.  
  
Dùng Trigger để thay đổi dây chuyền, nghĩa là có sự thay đổi ở một bảng thì các bảng khác có liên quan cũng thay đổi. ví dụ: trong bảng nhân viên mà xóa đi một nhân viên thì phòng của nhân viên đó sẽ giảm đi và công ty đó cũng giảm đi nhân viên. Ta có thể tạo một Trigger trên bảng để mỗi khi một nhân viên chuyển đi thì Trigger được kích hoạt và giảm đi số nhân viên của công ty.  
  
**III.Sử dụng Trigger trong MySQl**  
  
Chúng ta cần chú ý rằng các Trigger trong MySQL chỉ được kích hoạt thông qua các câu lệnh SQL, chúng không được kích hoạt bằng các thay đổi được thực thi bởi các hàm API trong bảng mà không thông qua việc truyền các câu lệnh SQL tới máy chủ MySQL.  
  
**1.Đặc điểm của Strigger**  
- Một Trigger có thể làm nhiều công việc khác nhau và có thể được kích hoạt bởi nhiều sự kiện.  
- Trigger không thể được tạo ra trên các hệ thống bảng.  
- Trigger chỉ có thể được tạo ra trên các hệ thống bảng  
- Khi một Trigger được kích hoạt thì data mới vừa được insert hoặc vừa mới được thay đổi sẽ được chứa trong bảng inser còn data vừa mới được Delete chứa trong bảng Delete. Đây là hai bảng tạm thời chỉ chứa trên bộ nhớ và chỉ có giá trị bên trong Trigger thôi.  
  
**2.Tạo một Trigger**  
Cú pháp câu lệnh tạo ra một Trigger như sau:

Mã:

CREATE

[DEFINER = { user | CURRENT\_USER }]

TRIGGER trigger\_name trigger\_time trigger\_event

ON table\_name FOR EACH ROW trigger\_statement

Trong đó:  
- Mệnh đề DEFINER xác định ngữ cảnh bảo mật sẽ được sử dụng khi kiểm tra quyền truy cập tại thời điểm Trigger được kích hoạt  
- Trigger\_name: Tên của Trigger  
- Trigger\_time: Xác định thời điểm thực thi của Trigger. Thời điểm này có thể là BEFORE ( trước) hoặc AFTER (sau) để chỉ báo rằng Trigger sẽ được kích hoạt trước hoặc sau mỗi dòng chinh sửa.  
- Trigger\_event: xác định kiểu câu lệnh sẽ kích hoạt Trigger. Nó chỉ có thể nhận các giá trị sau:  
  
+ INSERT: Trigger sẽ được kích họat khi có một dòng mới được chèn vào bảng thông qua các câu lệnh như: INSERT, LOAD DATA, và REPLACE.  
+ UPDATE: Trigger được kích họat khi có một dòng bị chỉnh sửa bởi câu lệnh UPDATE  
+ DELETE: Trigger được thực hiện khi một dòng bị xóa khỏi bảng. Chú ý rằng các câu lệnh DROP TABLE và TRUNCATE sẽ không kích họat được trigger này.  
- Tabel\_name: Tên bảng mà trigger sẽ họat động trên đó  
- Trigger\_statement: Câu lệnh sẽ được thực thi khi trigger được kích họat. Nếu như cần kích họat nhiều câu lệnh, chúng ta cần phải đưa chúng vào cấu trúc khối lệnh BEGIN ... END.  
  
Ví dụ  
Để làm rõ cách thức làm việc của Trigger chúng ta đi vào xét ví dụ sau:  
- Đầu tiên phải xây dụng một cơ sở dữ liệu gồm một số bảng.  
Ví dụ cơ sở dữ liệu imagegallery gồm 4 bảng được tạo như sau:

Mã:

mysql> create database imagegallery;

Query OK, 1 row affected (0.03 sec)

mysql> use imagegallery;

Database changed

mysql> create table users(id int(11) auto\_increment primary key,

user\_name varchar(30), password varchar(40), email varchar(15));

Query OK, 0 rows affected (0.00 sec)

mysql> create table images( id int(11) auto\_increment primary key,

cat\_id int(11), url varchar(200), title varchar(200), description text);

Query OK, 0 rows affected (0.00 sec)

mysql> show tables;

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

| Tables\_in\_imagegallery |

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

| images |

| users |

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

2 rows in set (0.00 sec)

Bây giờ ta tạo một Trigger mới (bangluu) kết hợp với bảng users. Trigger này sẽ thực hiện xóa từ bảng users và sau đó xóa tất cả các hàng trong bảng images có id giống với hàng đã xóa trong bảng users.

Mã:

CREATE TRIGGER bangluu AFTER DELETE ON users FOR EACH ROW

DELETE FROM images where id= Old.id

Cú pháp khác của Trigger  
  
Ta còn có cú pháp tạo một Trigger căn bản như sau:

Mã:

CREATE TRIGGER trigger\_name

ON table\_name or view\_name

FOR trigger\_class and trigger\_type(s)

AS Transact-SQL statements

Như vậy khi tạo ra một Trigger ta phải chỉ rõ là tạo ra Trigger trên table nào và được Trigger khi nào (Insert, Update hay Delete. Sau chữ AS là các câu lệnh SQL xử lý công việc.  
  
Việc sử dụng máy tính hay các cơ sở dữ liệu được làm việc trên máy tính là rất phổ biến. Nhưng không phải ai cũng giải quyết tốt các công việc một cách thuần thạo. Như vậy rất có thể sẽ có người xóa nhầm dữ liệu trong cơ sở dữ liệu. Việc này sẽ gây hậu quả không nhỏ nếu đó là dữ liệu cực quan trọng. Dùng Trigger chúng ta có thể lấy lại được dữ liệu đã bị xóa, vì chúng được lưu vào một bảng phụ khi chúng ta xóa hay làm bất kì công việc nào bằng Trigger.  
  
Ví dụ  
Giải sử ta có bảng categories trong cơ sở dữ liệu imagegallery ở trên. Để giám sát các hoạt động trên bảng này ta tạo ra một bảng giám sát có tên Audit có các cột giống hệt cột của categories và thêm 2 cột:  
-Audit\_Type : với các giá trị có thể là 'I','U','D' để ghi nhận bản ghi được Insert, Update hay Delete  
-Date\_Time\_Stamp : Data Type thuộc loại DateTime dùng để ghi nhận thời điểm xảy ra sự thay đổi  
Sau đó ta sẽ tạo ra 3 trigger dùng cho việc audit như sau:

Mã:

Insert Trigger

CREATE TRIGGER [AuditInsert categories]

ON [categories]

FOR Insert

AS

insert into audit select \*,'I’,getdate() From inserted

--Update Trigger

CREATE TRIGGER [AuditUpdatecategories]

ON [categories]

for UPDATE

AS

insert into audit select \*,'U',Getdate() from deleted

--Delete Trigger

CREATE TRIGGER [AuditDelete categories]

ON [categories]

FOR DELETE

AS

insert into audit select \*,'D',getdate() From deleted

**3. Danh sách, xóa và thay đổi trong Trigger**  
  
**a.Danh sách**  
  
Để có được một danh sách của Trigger trong cơ sở dữ liệu của bạn, bạn có thể sử dụng SQL statement hiển thị Strigger. Điều này sẽ trả về một danh sách của Strigger với tất cả các thuộc tính.  
  
**b.Xóa một Trigger**  
Để xáo một Trigger ta cần sử dụng DROP TRIGGER statement

Mã:

DROP TRIGGER trigger name;

**c.Thay đổi Trigger**  
Nếu bạn muốn thay đổi một Trigger bạn phải xóa nó đầu tiên, sau đó tái tạo lại nó

Mã:

DROP TRIGGER bangluu;

CREATE TRIGGER bangluu BEFORE DELETE ON users FOR EACH ROW

DELETE FROM images where id= Old.id