Học Procedure MySQL

**Tạo Stored Procedure**

DELIMITER $$

CREATE PROCEDURE procedureName()

BEGIN

   /\*Xu ly\*/

END $$

DELIMITER ;

**Trong đó**:

* Dòng đầu tiên DELIMITER $$ dùng để phân cách bộ nhớ lưu trữ thủ tục Cache và mở ra một ô lưu trữ mới. Đây là cú pháp nên bắt buộc bạn phải nhập như vậy
* Dòng CREATE PROCEDURE procedureName() dùng để khai báo tạo một Procedure mới, trong đó procedureName chính là tên thủ tục còn hai từ đầu là từ khóa.
* BEGINvà END; $$ dùng để khai báo bắt đầu của Procedure và kết thúc Procedure
* Cuối cùng là đóng lại ô lưu trữ DELIMITER ;

**Gọi Stored Procedure**

* Tạo xong rồi bây giờ làm thế nào để gọi đến Store này? Đơn giản để gọi tới Store  nào thì ta chỉ cần dùng cú pháp như sau:

|  |  |
| --- | --- |
| 1 | CALL storeName(); |

**Xem danh sách Stored Procedure trong hệ thống**

Nếu bạn không sử dụng SqlYog thì rất khó để quản lý Procedure vì không nhìn thấy được nó. Yên tâm vì trong MYSQL có hỗ trợ một số lệnh hiển thị dánh sách Stored Procedure bằng cách chạy lệnh sau:

|  |  |
| --- | --- |
| 1 | **show procedure status;** |

**Sửa Stored Procedure đã tạo**

Trong Mysql không cung cấp lệnh sửa Stored nên thông thường chúng ta sẽ chạy lệnh tạo mới. Tuy nhiên  có một lưu ý đó là nếu như bạn đã chạy lệnh tạo Procedure một lần rồi, sau đó bạn edit và chạy lại thì ngay lập tức sẽ bị báo lỗi ngay vì trùng tên. Để giải quyết vấn đề này thì chúng ta sẽ dùng lệnh Drop để xóa đi Procedure đó và tạo lại:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | DELIMITER $$    DROP PROCEDURE IF EXISTS `GetAllProducts`$$    CREATE PROCEDURE `GetAllProducts`()  BEGIN     SELECT \*  FROM products;  END$$    DELIMITER ; |

Và một lưu ý nữa là khi bạn dùng với quyền User nào thì Store đó sẽ có quyền thực hiện trong phạm vi của User đó. Ví dụ bạn không có quyền edit mà bạn tạo mới một Procedure Edit thì khi chạy sẽ bị báo lỗi ngay. Chính vì vậy thông thường khi edit bạn phải thêm người định nghĩa nó như sau:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | DELIMITER $$    DROP PROCEDURE IF EXISTS `GetAllProducts`$$    CREATE DEFINER=`root`@`localhost` PROCEDURE `GetAllProducts`()  BEGIN     SELECT \*  FROM products;  END$$    DELIMITER ; |

Dòng chữ DEFINER=`root`@`localhost` chính là tên người đã tạo ra nó.

# Truyền tham số vào Mysql Stored Procedure

 Procedure trong MYSQL thì sẽ tồn tại ba loại đó là **tham số IN**, **tham số OUT**và **tham số INOUT**

## Các loại tham số trong Mysql Stored Procedure

Chúng ta có hai loại tham số chính trong Procedure đó là:

* IN: Đây là chế độ mặc định (*nghĩa là nếu bạn không định nghĩa loại nào thì nó sẽ hiểu là IN*). Khi bạn sử dụng mức này thì nó sẽ được bảo vệ an toàn, có nghĩa là sẽ không bị thay đổi nếu như trong Procedure có tác động đến
* OUT: Chế độ này nếu như trong Procedure có tác động thay đổi thì nó sẽ thay đổi theo. Nhưng có điều đặc biệt là dù trước khi truyền vào mà bạn gán giá trị cho biến đó thì vẫn sẽ không nhận được  vì mặc định nó luôn hiểu giá trị truyền vào là NULL.
* INOUT: Đây là sự kết hợp giữa IN và OUT. Nghĩa là **có thể gán giá trị trước** và **có thể bị thay đổi** nếu trong Procedure có tác động tới

## Tham số loại IN trong Mysql Stored Procedure

Như trình bày ở trên tham số này sẽ được bảo vệ và không bị thay đổi trong quá trình sử dụng trong Procedure.

**Ví dụ**: Viết Store lấy chi tiết sản phẩm theo ID

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | DELIMITER $$  DROP PROCEDURE IF EXISTS getById $$  CREATE PROCEDURE getById(IN idVal INT(11))  BEGIN      SELECT \* FROM products WHERE id = idVal;  END; $$  DELIMITER; |

Chạy Procedure này:

|  |  |
| --- | --- |
| 1 | CALL getById(1); |

## Tham số loại OUT trong Mysql Stored Procedure

Loại out nếu trong quá trình thực thi mà Procedure có tác động đến tham số này thì bên ngoài nó ảnh hưởng theo. Khi nhận tham số này thì Procedure sẽ hiểu đó là giá trị NULL nên dù bạn có gán giá trị cho biến trước khi truyền vào nó vẫn lấy NULL.

* Biến truyền vào phải có chữ @ đằng trước, ví dụ @title

**Ví dụ**: Truyền tham số title kiểu OUT vào Procedure và đổi giá trị cho nó, sau đó bên ngoài Procedure hiển thị giá trị của title.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | DELIMITER $$  DROP PROCEDURE IF EXISTS changeTitle $$  CREATE PROCEDURE changeTitle(OUT title VARCHAR(255))  BEGIN      SET title = 'Hoc lap trinh online tai freetuts.net';  END; $$  DELIMITER; |

Bây giờ ta gọi Procedure này như sau:

|  |  |
| --- | --- |
| 1  2  3 | CALL changeTitle(@title);    SELECT @title; |

**Như vậy ra rút ra kết luận như sau**:

* Khi truyền tham số dạng OUT mục đích là lấy dữ liệu trong Proedure và sử dụng ở bên ngoài. (Giống dạng Function)
* Khi truyền tham số vào dạng OUT phải có chữ @ đằng trước biến
* Hoạt động giống tham chiếu nên biến truyền vào dạng OUT không cần định nghĩa trước, chính vì vậy khởi đầu nó có giá trị NULL

DELIMITER $$

DROP PROCEDURE IF EXISTS changeinfor $$

CREATE PROCEDURE changeinfor(OUT ip VARCHAR(255) CHARSET utf8 COLLATE utf8\_unicode\_ci)

BEGIN

SET ip = 'Giá trị đã được thay đổi';

END $$

CREATE TABLE IF NOT EXISTS `members` (

  `us\_id` INT(11) NOT NULL AUTO\_INCREMENT,

  `us\_username` VARCHAR(30) COLLATE utf8\_unicode\_ci DEFAULT NULL,

  `us\_password` VARCHAR(32) COLLATE utf8\_unicode\_ci DEFAULT NULL,

  `us\_level` TINYINT(1) DEFAULT '0',

  PRIMARY KEY (`us\_id`)

) ENGINE=INNODB  DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci AUTO\_INCREMENT=4 ;

* **CHARSET utf8 và COLLATE utf8\_unicode\_ci: khi khai báo cùng lúc với khai báo biến thì ta sẽ không sử dụng dấu ‘=’, còn sau khi khai báo biến rồi mới khai báo thì ta sẽ sử dụng dấu ‘=’ ví dụ: CHARSET = utf8 COLLATE = utf8\_unicode\_ci**
* **Trong cùng một Procedure ta có thể truyền vào nhiều loại biến (IN, OUT, INOUT)**

## Tham số dạng INOUT trong Mysql Stored Procedure

INOUT là sự kết hợp giữa IN và OUT, nghĩa là:

* Nó có thể được định nghĩa trước và gán gia trị trước rồi truyền vào Procedure, điều này với dạng OUT thì không thể được nhưng IN thì được.
* Sau khi thực thi xong nếu trong Procedure có tác động đến thì ảnh hưởng theo. Điêu này dạng IN không được nhưng OUT thì không được.

**Ví dụ**: Tạo Procedure

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | DELIMITER $$    DROP PROCEDURE IF EXISTS counter $$    CREATE PROCEDURE counter(INOUT number INT(11))  BEGIN      SET number = number + 1;  END; $$  DELIMITER; |

**Gọi sử dụng**:

|  |  |
| --- | --- |
| 1  2  3 | SET @counter = 1;  CALL counter(@counter);  SELECT @counter; |

Và kết quả là 2.

Nhưng nếu ta dùng dạng OUT thì kết quả sẽ là NULL. Lý do là bên trong **có tăng lên 1** nhưng nó lấy giá trị truyền vào dạng OUT là NULL nên 1 + NULL sẽ là NULL.

|  |
| --- |
| DELIMITER $$    DROP PROCEDURE IF EXISTS `docSo`$$    CREATE PROCEDURE `docSo`(IN a INT(11))  BEGIN        DECLARE message VARCHAR(255);        CASE a          WHEN 0 THEN              SET message = 'KHONG';          WHEN 1 THEN              SET message = 'MOT';          WHEN 2 THEN              SET message = 'HAI';          WHEN 3 THEN              SET message = 'BA';          WHEN 4 THEN              SET message = 'BON';          WHEN 5 THEN              SET message = 'NAM';          WHEN 6 THEN              SET message = 'SAU';          WHEN 7 THEN              SET message = 'BAY';          WHEN 8 THEN              SET message = 'TAM';          WHEN 9 THEN              SET message = 'CHIN';          ELSE              SET message = 'KHONG TIM THAY';        END CASE;        SELECT message;    END$$    DELIMITER ; |

Chạy thủ tục:

CALL docSo(1); -- MOT

CALL docSo(2); -- HAI

**Với hàm khai báo biến thì ta phải khai báo:**

**CREATE PROCEDURE `docSo`(IN a INT(11))**

## 1. Khai báo biến trong MySql Stored Procedure

Để định nghĩa một biến mới ta dùng cú pháp :

|  |  |
| --- | --- |
| 1 | DECLARE variable\_name datatype(size) DEFAULT default\_value |

## Gán giá trị cho biến trong MySql Stored Procedure

SET variable\_name = 'value';

**Ví dụ: Gán giá trị thông qua lệnh SELECT**

|  |  |
| --- | --- |
| 1  2  3  4 | DECLARE total\_products INT DEFAULT 0    SELECT COUNT(\*) **INTO** total\_products  FROM products |

**Trong ví dụ này thì trước tiên nó sẽ thực hiện câu truy vấn SQL đếm tổng số record và sau đó gán vào biến total\_products bằng lệnh (COUNT(\*) INTO total\_products).**

## 4. Phạm vi hoạt động của biến

Nếu như bạn định nghĩa một biến bên trong phần thân của Procedure (*giữa BEGIN và END*) thì đó ta gọi là biến cục bộ của Procedure. Bạn có thể định nghĩa nhiều biến trong một Procedure.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | DELIMITER $$  DROP PROCEDURE IF EXISTS tinhTong $$  CREATE PROCEDURE tinhTong()  BEGIN      DECLARE a INT (11) DEFAULT 0;      DECLARE b INT (11) DEFAULT 0;      DECLARE tong INT (11) DEFAULT 0;        SET a = 200;      SET b = 300;      SET tong = a + b;        SELECT tong;    END; $$  DELIMITER; |

Ở procedure này tôi đã định nghĩa các biến a,b,tong và tính toán trên đó. Bạn chạy câu sql trên và sau đó gọi nó bằng lệnh call tinhTong thì nó sẽ cho kết quả là 500.

**Ví dụ**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | DELIMITER $$  DECLARE tong INT (11) DEFAULT 0;  DROP PROCEDURE IF EXISTS tinhTong $$  CREATE PROCEDURE tinhTong()  BEGIN      DECLARE a INT (11) DEFAULT 0;      DECLARE b INT (11) DEFAULT 0;        SET a = 200;      SET b = 300;      SET tong = a + b;        SELECT tong;    END; $$  DELIMITER; |

**Nếu một biến được khai báo bên ngoài Procedure thì bên trong Procedure sẽ không nhận được và sẽ thông báo lỗi.**

# Câu lệnh if else trong MYSQL

**Cú pháp mệnh đề if - else trong MYSQL như sau**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | IF if\_expression THEN      commands     ELSEIF elseif\_expression THEN      commands     ELSE      commands  END IF; |

# Câu lệnh CASE trong MYSQL

## Cú pháp CASE trong MySql

**Cú pháp của lệnh CASE như sau**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CASE  case\_expression     WHEN when\_expression\_1 THEN commands     WHEN when\_expression\_2 THEN commands     ...     ELSE commands  END CASE; |

**Trong đó**:

* WHEN dịch tiếng anh nghĩa là KHI, tức là **KHI MỆNH ĐỀ ĐIỀU KIỆN ĐÚNG** thì  sẽ thực thi câu lệnh bên trong nó.
* ELSE chính là trường hợp cuối cùng nếu như các điều kiện WHEN ở trên không đúng thì nó sẽ được chạy. Nếu để ý kĩ thì bạn thấy đây chính là lệnh default trong **Switch Case** của các ngôn ngữ lập trình khác.

Ngoài ra lệnh CASE còn sử dụng trong câu lệnh select

* SELECT
* us\_username,
* CASE us\_gender
* WHEN 1 THEN 'Nam'
* ELSE 'Nu'
* END AS gender;
* FROM members

**Lệnh CASE trong câu SELECT hơi khác chút xíu đó là chỉ có lệnh END thay vì END CASE như trong Procedure.Các bạ lưu ý điểm này để viết câu truy vấn cho đúng nhé.**

# Vòng lặp while trong MySql

Trong MySql có ba vòng lặp chính đó là vòng lặp while, vòng lặp repeat, vòng lặp với lệnh LOOP và LIEVE.

## Cú pháp vòng lặp while trong MYSQL

WHILE expression DO

   Statements

END WHILE;

Ví dụ:

DELIMITER $$

DROP PROCEDURE IF EXISTS loopWhile$$

CREATE PROCEDURE loopWhile(

    IN a INT(11),

    IN b INT(11)

)

BEGIN

    -- Chuoi in ra man hinh--

        DECLARE str  VARCHAR(255) DEFAULT '';

        WHILE (a <= b) DO

        SET  str = CONCAT(str,a,',');

                SET  a = a + 1;

        END WHILE;

        SELECT str;

END$$

DELIMITER ;

**Chạy thử**:

|  |  |
| --- | --- |
| 1 | CALL loopWhile(1,10); |

# **Sử dụng view, trigger và procedure trong mysql**

**1. Bảng ảo View**

* View là bảng ảo giúp giới hạn truy cập một số cột dòng trên các bảng dữ liệu.
* Views đảm bảo tính bảo mật bằng các giới hạn truy cập sau:
  + Các hàng trên một bảng
  + Các cột trên một bảng
  + Các hàng và cột trên một bảng
  + Các hàng giữa các bảng sử dụng phép joins
* Thuận lợi khi sử dụng view
  + Cung cấp dữ liệu cần thiết cho người dùng
  + Che dấu đi sự phức tạp của dữ liệu
  + Tổ chức dữ liệu từ nhiều tài nguyên không đồng nhất
  + Giảm kích cỡ của đối tượng
* Cú pháp tạo view

CREATE VIEW ViewName

AS

//Query

**Query là một truy vấn mysql, lưu ý khi sử dụng câu lệnh select:**

o Không sử dụng biến local, user hay session

o Những lệnh dịch sẵn cũng không thể sử dụng

o Bảng tạm hay các bảng ảo khác cũng không được sử dụng làm nguồn dữ liệu cho câu truy vấn này.

* Ví dụ tạo view

CREATE VIEW SalePerOrder

AS

SELECT orderNumber,

SUM (quantityOrdered \* priceEach) total

FROM orderDetails

GROUP by orderNumber

ORDER BY total DESC

Sử dụng view:

SELECT total

FROM salePerOrder

WHERE orderNumber = 1000

* Ưu nhược điểm của view

1. Ưu điểm
   * An ninh – Cho phép người quản trị CSDL (DBA) cung cấp cho người sử dụng chỉ những dữ liệu cần thiết
   * Đơn giản hóa – Các view cũng có thể sử dụng để ẩn và sử dụng lại các truy vấn phức tạp
2. Nhược điểm
   * Không thể chứa mệnh ñề COMPUTE hoặc COMPUTE BY
   * Vì tạo ra bảng tạm nên view sẽ làm tốn tài nguyên, chậm hệ thống.
   * Không thể chứa mệnh ñề ORDER BY trừ khi có TOP n
   * Không thể chứa mệnh ñề INTO
   * Không thể chứa các table tạm
   * Có tối đa 1024 cột

* Xóa view

Có thể xóa view từ database sử dụng cú pháp:

DROP VIEW view\_name

**2. Trigger**

* Khái niệm về trigger

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.

* Lợi ích khi sử dụng trigger

Bởi vì 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...

* Cách sử dụng

**1.Tạo trigger**

DELIMITER $$

CREATE TRIGGER trigger\_name BEFORE | AFTER INSERT |UPDATE | DELETE ON tablename

FOR EACH ROW

BEGIN

…….

END

**Ví dụ:**

DELIMITER $$

CREATE TRIGGER trig2 AFTER UPDATE ON thanhvien

FOR EACH ROW

BEGIN

INSERT INTO thanhvien

VALUES (6,'ABC','DEF',1);

END $$

**2.Xóa trigger**

DROP TRIGGER tablename.trigger\_name

**3.Sửa, xem thông tin trigger:**

ALTER TRIGGER, SHOW CREATE TRIGGER, hoặc SHOW TRIGGER STATUS

**4.Để hiển thị các trigger:**

* Hiển thị tất cả các trigger

SELECT \* FROM Information\_Schema.Trigger

WHERE Trigger\_schema = 'database\_name' AND Event\_object\_table = 'table\_name';

* Xem nội dung trigger

SELECT \* FROM Information\_Schema.Trigger

WHERE Trigger\_schema = 'database\_name' AND

Trigger\_name = 'trigger\_name';

* Xem tất cả trigger của một bảng dữ liệu

SELECT \* FROM Information\_Schema.Trigger

WHERE Trigger\_schema = 'database\_name' AND

Event\_object\_table = 'table\_name';

Kết nối đến MYSQL

Cài đặt trong cmd: pip install mysql-connector-python

import mysql.connector

from mysql.connector import Eror

# mo ket noi toi Database

db = mysql.connector.connect("localhost","testuser","test123","TESTDB" )

# chuan bi mot doi tuong cursor boi su dung phuong thuc *cursor()*

cursor = db.cursor()

# Thuc thi truy van SQL boi su dung phuong thuc *execute()*.

cursor.execute("SELECT VERSION()")

# Lay mot hang boi su dung phuong thuc *fetchone()*.

data = cursor.fetchone()

print "Database version : %s " % data

# ngat ket noi voi server

db.close()

* **import mysql.connector :** Dòng này nhập mô-đun Python kết nối MySQL trong chương trình của bạn để bạn có thể sử dụng API của mô-đun này để kết nối MySQL.

**from mysql.connector import Error**

* **Đối tượng lỗi kết nối mysql**  được sử dụng để hiển thị cho chúng tôi một lỗi khi chúng tôi không kết nối được Cơ sở dữ liệu hoặc nếu có bất kỳ lỗi cơ sở dữ liệu nào khác xảy ra trong khi làm việc với cơ sở dữ liệu. Ví dụ ER\_ACCESS\_DENIED\_ERROR khi tên người dùng hoặc mật khẩu bị sai.
* **mysql.connector.connect()**
* Sử dụng chức năng này, chúng tôi có thể kết nối Cơ sở dữ liệu MySQL, chức năng này chấp nhận bốn tham số bắt buộc: **Máy chủ, Cơ sở dữ liệu, Người dùng và Mật khẩu** mà chúng tôi đã thảo luận.
* Hàm **connect () đã** thiết lập kết nối tới cơ sở dữ liệu MySQL từ ứng dụng python và trả về một đối tượng MySQLConnection. Sau đó, chúng ta có thể sử dụng đối tượng MySQLConnection để thực hiện các hoạt động khác nhau trên Cơ sở dữ liệu MySQL.

**conn.is\_connected()**

is\_connected()là chức năng của lớp MySQLConnection mà qua đó chúng ta có thể xác minh là ứng dụng python của chúng ta được kết nối với MySQL.

**connection.cursor()**

* Phương thức này trả về một đối tượng con trỏ. Sử dụng một đối tượng con trỏ, chúng ta có thể thực hiện các truy vấn SQL.
* Lớp MySQLCthon khởi tạo các đối tượng có thể thực thi các hoạt động như các câu lệnh SQL.   
  Các đối tượng con trỏ tương tác với máy chủ MySQL bằng cách sử dụng đối tượng MySQLConnection.

**cursor.close()**

* Sử dụng phương thức đóng của con trỏ, chúng ta có thể đóng đối tượng con trỏ. Khi chúng ta đóng đối tượng con trỏ, chúng ta không thể thực thi bất kỳ câu lệnh SQL nào.
* **connection.close()**
* Cuối cùng, chúng tôi sẽ đóng kết nối cơ sở dữ liệu MySQL bằng cách sử dụng hàm close () của lớp MySQLConnection.

**Commit cac thay doi vao trong Database**

**connection**.**commit()**

Tóm tắt:

**Để kết nối đến cơ sở DL MYSQL ta cần import một module hỗ trợ việc kết nối đó là mysql.connector**

**Ta khởi tạo một đối tượng kết nối (mysqlconnection)**

**Ví dụ: con = mysql.connector.connect(host = ‘’,database = ‘’,user = ‘’, password = ‘’)**

**Hàm connect() cần 4 tham số: host, database, user, password.**

import MySQLdb

# mo ket noi toi Database

db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# chuan bi mot doi tuong cursor boi su dung phuong thuc *cursor()*

cursor = db.cursor()

# Xoa bang neu no da ton tai boi su dung phuong thuc *execute()*.

cursor.execute("DROP TABLE IF EXISTS SINHVIEN")

# Tao mot bang

sql = """CREATE TABLE SINHVIEN (

HO CHAR(20) NOT NULL,

TEN CHAR(20),

TUOI INT,

GIOITINH CHAR(1),

HOCPHI FLOAT )"""

cursor.execute(sql)

# ngat ket noi voi server

db.close()

**Nếu một kết nối được thành lập, thì một đối tượng Connection được trả về và được lưu giữ vào trong db, nếu không db được thiết lập là None. Tiếp đó, đối tượng db được sử dụng để tạo đối tượng cursor, mà tiếp đó được sử dụng để thực thi các truy vấn SQL. Cuối cùng, trước khi thoát ra, nó bảo đảm rằng kết nối tới Database được đóng và các resource được giải phóng.**

import mysql.connector

from mysql.connector import Error

try:

con = mysql.connector.connect(host='localhost ', database = 'member', user = 'root', password = '')

if con.is\_connected():

con\_info = con.get\_server\_info()

print ('Đã kết nối: %s' %con\_info)

cursor = con.cursor()

cursor.execute('select \* From thanhvien where level = 2')

record = cursor.fetchall()

for index in record:

print(index)

except Error as e:

print('Có lỗi phát sinh %s' %e)

finally:

con.close()

cursor.execute('select \* From thanhvien where level = 2')

record = cursor.fetchall()

**Câu lệnh trên đây trả về một list các phần tử và mỗi phần tử là một tuple bao gồm các trường dữ liệu của bảng csdl**