

## BÀI 14. LẬP TRÌNH VỚI CSDL TRONG JAVA

---

1

### Nội dung

- Các khái niệm cơ bản về CSDL quan hệ
- Cú pháp SQL cơ bản
- Lập trình với CSDL trong Java

2

## 1. KHÁI NIỆM CƠ BẢN VỀ CSDL QUAN HỆ

---

3

## Các khái niệm cơ bản

- CSDL là một tập hợp các dữ liệu có mối liên hệ logic và được lưu trữ theo một mô hình nào đó
- Hệ quản trị CSDL là hệ thống phần mềm cho phép tạo lập, quản lý và cung cấp các thao tác để làm việc với CSDL
- CSDL quan hệ là mô hình mà trong đó dữ liệu được tổ chức dưới dạng bảng:
  - Cột: các trường( hoặc thuộc tính). Mỗi trường được xác định trên miền xác định của trường
  - Dòng: các bản ghi dữ liệu
- Các bảng trong CSDL quan hệ có liên kết với nhau bởi các trường

4

## Mô hình dữ liệu quan hệ - Ví dụ

Bảng tbl\_product

productID	productName	category	supplier	price
LAP001	HP Pavillon G50	Laptop	FPT	467
MOU103	Logitech M185	Mouse	SV House	12
LAP012	Fujitsu LH530	Laptop	FPT	520

Bảng tbl\_bill

billID	customer	date
1123	Bui Trong Tung	24/8/2014
1124	Nguyen Ha Dong	01/09/2014

Bảng tbl\_bill\_line

billID	productID	quantity
1123	LAP012	1
1123	MOU103	3
1124	LAP012	2

5

## Các khái niệm cơ bản

- Khóa K là tập các trường sao cho mỗi giá trị của K xác định duy nhất một bộ dữ liệu trong bảng
  - Trên một bảng có thể có nhiều khóa, nhưng chỉ chọn một khóa làm khóa chính
- Nếu K là khóa thì mọi  $K^* \supset K$  là khóa.  $K^*$  được gọi là siêu khóa
- K là khóa tối thiểu nếu mọi  $K' \subset K$  không phải là khóa
  - Khóa chính thường là khóa tối thiểu
- Khóa ngoài  $K_f$  nếu  $K_f$  là khóa chính trên một bảng khác

6

## Ví dụ

- Bảng tbl\_product:
  - Khóa chính: productID
  - Siêu khóa: (productID,productName)...
- Bảng tbl\_bill:
  - Khóa chính: billID
- Bảng tbl\_bill\_line
  - Khóa chính: (billID, productID)
  - Khóa ngoài: productID, billID

7

## 2. CÚ PHÁP SQL CƠ BẢN

---

8

## SQL là gì

- SQL (Structured Query Language): ngôn ngữ truy vấn trên hệ quản trị CSDL
- Là ngôn ngữ phổ biến nhất, được hầu hết các hệ quản trị CSDL hỗ trợ
- SQL bao gồm:
  - Ngôn ngữ định nghĩa dữ liệu: tạo bảng, mối liên hệ giữa các bảng, quy tắc, ràng buộc áp dụng lên dữ liệu
  - Ngôn ngữ thao tác dữ liệu: thêm, sửa, xóa, tìm kiếm
  - Ngôn ngữ điều khiển dữ liệu: phân quyền người dùng...
- Hệ quản trị CSDL MySQL: [mysql.com](http://mysql.com)

9

## Quy tắc định danh và kiểu dữ liệu

- Định danh:
  - Chữ cái, chữ số, dấu \_
  - Tối đa: 32 ký tự
  - Không phân biệt chữ hoa, chữ thường
- Kiểu dữ liệu:
  - CHAR(n): xâu có độ dài n ký tự
  - VARCHAR(n): xâu có độ dài tối đa n ký tự
  - NCHAR(n): hỗ trợ Unicode
  - NVARCHAR: hỗ trợ Unicode
  - Int, SmallInt, Float, Real, Double
  - Date: định dạng YYYY-MM-DD
  - DateTime: định dạng YYYY-MM-DD HH:MM:SS
  - ...

10

## Một số câu lệnh cơ bản

- `SHOW DATABASES;`
  - Liệt kê các CSDL
- `CREATE DATABASE IF NOT EXISTS Ten_CSDL;`
  - Tạo mới CSDL
- `CREATE DATABASE Ten_CSDL;`
- `DROP DATABASE IF EXIST Ten_CSDL;`
  - Xóa CSDL
- `DROP DATABASE Ten_CSDL;`
- `USE Ten_CSDL;`
  - Truy cập CSDL để thao tác

11

## Tạo bảng

### • Cú pháp

```
CREATE TABLE [IF NOT EXISTS] tenBang(
    tenCot1 KieuDuLieu [NOT NULL] [DEFAULT value]... ,
    tenCot2 KieuDuLieu [NOT NULL] [DEFAULT value]... ,
    .../
    PRIMARY KEY (...)
    [CONSTRAINT tenRangBuoc kieuRangBuoc menhDe]
) [DEFAULT CHARSET = charSet];
```

### • Giải thích

- [...] : có thể có hoặc không cần trong câu lệnh
- NOT NULL : trường này phải có giá trị trên các bản ghi
- DEFAULT : thiết lập giá trị mặc định
- PRIMARY KEY : thiết lập khóa chính
- CONSTRAINT : thiết lập ràng buộc
- DEFAULT CHARSET: Bảng mã mặc định

12

## Một số kiểu ràng buộc

- Ràng buộc khóa ngoài

```
CONSTRAINT tenRangBuoc FOREIGN KEY (...)
REFERENCES tenBangThamChieu (...)
[ON DELETE RESTRICT | CASCADE]
[ON UPDATE RESTRICT | CASCADE]
```

- Giải thích:

- FOREIGN KEY : thiết lập khóa ngoài
- REFERENCES : tham chiếu trên bảng tham chiếu (bảng cha)
- UPDATE : thay đổi khóa
- DELETE : xóa khóa
- RESTRICT : cấm thay đổi/xóa khóa ngoài trên bảng cha nếu bảng con còn có bản ghi chứa khóa
- CASCADE : cho phép thay đổi/xóa khóa ngoài trên bảng cha, cập nhật lại trên bảng con

13

## Một số kiểu ràng buộc(tiếp)

- Ràng buộc giá trị của trường

```
CONSTRAINT tenRangBuoc CHECK bieuThucKiemTra
```

- Giải thích: ràng buộc này bắt buộc giá trị gán cho một trường trên các bản ghi phải thỏa mãn bieuThucKiemTra

14

## Tạo bảng - Ví dụ

Bảng tbl\_product

productID	productName	category	supplier	price
LAP001	HP Pavillon G50	Laptop	FPT	467
MOU103	Logitech M185	Mouse	SV House	12
LAP012	Fujitsu LH530	Laptop	FPT	520

Bảng tbl\_bill

billID	customer	phone	date
1123	Bui Trong Tung	0988888888	24/8/2014
1124	Nguyen Ha Dong	0912345678	01/09/2014

Bảng tbl\_bill\_line

billID	productID	quantity
1123	LAP012	1
1123	MOU103	3
1124	LAP012	2

15

## Tạo bảng – Ví dụ

```
CREATE TABLE IF NOT EXISTS tbl_product(
    productID          CHAR(6)          NOT NULL ,
    productName        VARCHAR(30)      NOT NULL UNIQUE,
    category           VARCHAR(20)      NOT NULL,
    supplier            VARCHAR(30),
    price              INT UNSIGNED NOT NULL DEFAULT 0,
    PRIMARY KEY(productID)
);
```

```
CREATE TABLE IF NOT EXISTS tbl_bill(
    billID            INT UNSIGNED          NOT NULL AUTO_INCREMENT,
    customer          VARCHAR(30)          NOT NULL,
    phone             VARCHAR(15),
    date              DATE                  NOT NULL,
    PRIMARY KEY(billID)
);
```

16



## Tạo bảng – Ví dụ

```
CREATE TABLE IF NOT EXISTS tbl_bill_line(
    billID          INT UNSIGNED NOT NULL,
    productID       CHAR(6)      NOT NULL,
    quantity        INT UNSIGNED NOT NULL DEFAULT 1,
    PRIMARY KEY(billID, productID),
    CONSTRAINT fk_billID FOREIGN KEY(billID) REFERENCES
        tbl_bill(billID) ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_productID FOREIGN KEY(billID) REFERENCES
        tbl_product(productID)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT chk_quantity CHECK (quantity > 0)
);
```

17

## Thay đổi cấu trúc bảng

### • Cú pháp

```
ALTER TABLE tenBang
    ADD COLUMN tenCot KieuDuLieu; --Thêm cột
```

```
ALTER TABLE tenBang
    DROP COLUMN tenCot KieuDuLieu; --Xóa cột
```

```
ALTER TABLE tenBang
    CHANGE COLUMN tenCot TO KieuDuLieuMoi;
```

```
ALTER TABLE tenBang
    ADD CONSTRAINT ...; --Thêm ràng buộc
```

```
ALTER TABLE tenBang
    DROP CONSTRAINT tenRangBuoc; --Xóa ràng buộc
```

```
ALTER TABLE tenBang
    DROP FOREIGN KEY tenRangBuoc; --Dùng cho MySQL
```

18

## Tìm kiếm trên CSDL

- Cú pháp

```
SELECT dsCot          --danh sách cột cần lấy dữ liệu
FROM dsBang           --danh sách bảng chứa dữ liệu
[WHERE bieuThuc]      --biểu thức điều kiện dữ liệu cần lấy
[GROUP BY cot]
[HAVING bieuThucHV]
[ORDER BY cot ASC|DEC]
[UNION|INTERSECT|MINUS]...;
```

19

## Truy vấn trên 1 bảng – Ví dụ

- Hiển thị thông tin toàn bộ hóa đơn từ bảng tbl\_bill

```
SELECT *
FROM tbl_bill;
```

- Hiển thị tên các sản phẩm và giá từ bảng tbl\_product

```
SELECT productName, price
FROM tbl_product;
```

- Hiển thị danh sách các sản phẩm có giá trên 100 và do FPT cung cấp

```
SELECT productID, productName, category, price
FROM tbl_product
WHERE (price > 100) AND (supplier = 'FPT');
```

20

## Các phép toán trên biểu thức điều kiện

- Các phép so sánh: =, <>, >, <, >=, <=
- Các phép toán logic: NOT, AND, OR, XOR
- Các phép toán phạm vi:
  - BETWEEN...AND...: nằm trong khoảng giá trị liên tục
  - NOT BETWEEN...AND...: nằm ngoài khoảng giá trị
  - IN (value1, value2,...): nằm trong tập giá trị
  - NOT IN(...): không nằm trong tập
  - LIKE: đối sánh chuỗi ký tự:
    - \_: một ký tự bất kỳ
    - %: một chuỗi bất kỳ
    - Ví dụ:
      - abc% : chuỗi hợp lệ nếu bắt đầu là abc
      - %abc: chuỗi hợp lệ nếu kết thúc là abc
      - a\_b%: chuỗi hợp lệ nếu bắt đầu là a, tiếp theo là một ký tự bất kỳ, sau đó đến ký tự b

21

## Tìm kiếm loại trừ các bản ghi trùng nhau

- Sử dụng từ khóa DISTINCT
- Ví dụ: hiển thị danh sách các nhà cung cấp
- Hiển thị danh sách các nhà cung cấp mặt hàng Laptop

```
SELECT DISTINCT supplier
FROM tbl_product;
```

```
SELECT DISTINCT supplier
FROM tbl_product
WHERE category = 'Laptop';
```

22

## Phân nhóm kết quả tìm kiếm

- Phân nhóm: sử dụng GROUP BY. Ví dụ

```
SELECT *
FROM tbl_product
GROUP BY category;
```

- Kết quả: chỉ có bản ghi đầu tiên mỗi nhóm được hiển thị

productID	productName	category	supplier	price
LAP001	HP Pavillon G50	Laptop	FPT	467
MOU103	Logitech M185	Mouse	SV House	12

- Cột dùng để phân nhóm phải xuất hiện trên mệnh đề SELECT

23

## Thống kê kết quả tìm kiếm

- Đếm: COUNT

- COUNT(\*): số bản ghi trả về
- COUNT(tenCot): số bản ghi mà có giá trị của trường tenCot khác NULL

- Ví dụ: đếm số mặt hàng

```
SELECT COUNT(*) AS 'Count'
FROM tbl_product;
```

Kết quả

Count
3

- Ví dụ: đếm số mặt hàng của mỗi nhà sản xuất

```
SELECT supplier, COUNT(*)
FROM tbl_product
GROUP BY supplier;
```

supplier	COUNT(*)
FPT	2
SV House	1

24

## Thống kê kết quả tìm kiếm

- Tìm giá trị lớn nhất, nhỏ nhất, trung bình: MAX, MIN, AVG
- Tính tổng các giá trị: SUM
- Ví dụ

```
SELECT MAX(price), MIN(price), AVG(price)
FROM tbl_product;
```

MAX(price)	MIN(price)	AVG(price)
520	12	333.0000

- Kết hợp GROUP BY: Ví dụ

```
SELECT supplier, MAX(price), MIN(price), AVG(price)
FROM tbl_product
GROUP BY supplier;
```

supplier	MAX(price)	MIN(price)	AVG(price)
FPT	520	467	493.5000
SV House	12	12	12.0000

25

## Mệnh đề HAVING

- Tương tự như WHERE nhưng cho phép kết hợp với GROUP BY và thực hiện trên các hàm COUNT, MAX...
- Ví dụ: Hiển thị danh sách các nhà cung cấp các mặt hàng có giá tối thiểu lớn hơn 400

```
SELECT supplier, MIN(price) as 'min'
FROM tbl_product
GROUP BY supplier
HAVING min > 400;
```

supplier	Min
FPT	467

26

## Sắp xếp kết quả tìm kiếm

- Sử dụng từ khóa ORDER BY. Mặc định sắp xếp tăng dần
- Sắp xếp tăng dần: ASC
- Sắp xếp giảm dần: DESC

```
SELECT *
FROM tbl_product
ORDER BY supplier DESC, price;
```

productID	productName	category	supplier	price
MOU103	Logitech M185	Mouse	SV House	12
LAP001	HP Pavillon G50	Laptop	FPT	467
LAP012	Fujitsu LH530	Laptop	FPT	520

27

## Sắp xếp kết quả tìm kiếm (tiếp)

- Kết hợp các mệnh đề
- Ví dụ: hiển thị các nhà cung cấp theo số chủng loại mặt hàng giảm dần

```
SELECT supplier, COUNT(*) AS 'count'
FROM tbl_product
GROUP BY supplier
ORDER BY Count DESC;
```

supplier	Count
FPT	2
SV House	1

28

## Truy vấn trên nhiều bảng

- Cú pháp

```
SELECT dsCot          -- Danh sách các trường cần lấy
FROM dsBang           -- Danh sách bảng
WHERE bieuThuc        -- Biểu thức điều kiện
...;
```

- Ví dụ: Hiển thị danh sách các mặt hàng đã bán được trên hóa đơn số 1123

```
SELECT productName, category, quantiy
FROM tbl_product, tbl_bill_line
WHERE (billID = 1123) AND (tbl_product.productID =
tbl_bill_line.productID);
```

29

## Truy vấn trên nhiều bảng (tiếp)

- Hiển thị lịch sử mua hàng của khách hàng Bui Trong Tung

```
SELECT tbl_bill_line.billID, date, productName, quantiy
FROM tbl_product, tbl_bill, tbl_bill_line
WHERE (customer = 'Bui Trong Tung') AND
(tbl_product.productID = tbl_bill_line.productID) AND
(tbl_bill.billID = tbl_bill_line_billID);
```

30

## Truy vấn lồng nhau

- Lồng trên mệnh đề WHERE: Kiểm tra thành viên tập hợp(IN, NOT IN)
- Ví dụ: Đưa ra tên của các nhà cung cấp đồng thời 2 mặt hàng Laptop và chuột (Mouse)

```
SELECT DISTINCT supplier
FROM tbl_product
WHERE category = 'Laptop' AND supplier IN (
    SELECT supplier
    FROM tbl_product
    WHERE category = 'Mouse');0
```

- Ví dụ: Đưa ra tên của các nhà cung cấp đồng thời mặt hàng Laptop nhưng không cung cấp chuột (Mouse)

```
SELECT DISTINCT supplier
FROM tbl_product
WHERE category = 'Laptop' AND supplier NOT IN (
    SELECT supplier
    FROM tbl_product
    WHERE category = 'Mouse');
```

31

## Truy vấn lồng nhau (tiếp)

- Lồng trên mệnh đề WHERE: Kiểm tra sự tồn tại của bản ghi (EXISTS, NOT EXISTS)
- Ví dụ: Đưa ra thông tin các mặt hàng đã bán ít nhất một lần

```
SELECT *
FROM tbl_product
WHERE EXISTS (SELECT productID
              FROM tbl_bill_line
              WHERE tbl_bill_line.productID =
                  tbl_product.productID);
```

- Ví dụ: Đưa ra thông tin các mặt hàng chưa bán được lần nào

```
SELECT *
FROM tbl_product
WHERE NOT EXISTS (SELECT productID
                  FROM tbl_bill_line
                  WHERE tbl_bill_line.productID =
                      tbl_product.productID);
```

32



## Truy vấn lồng nhau (tiếp)

- Lồng trên mệnh đề WHERE: sử dụng với các phép toán tập hợp (>ALL, >=ALL, <ALL, <=ALL)
- Ví dụ: Đưa ra thông tin mặt hàng có giá thấp nhất

```
SELECT *
FROM tbl_product
WHERE price >= ALL(SELECT price from tbl_product);
```

- Ví dụ: Đưa ra thông tin mặt hàng có giá cao nhất

```
SELECT *
FROM tbl_product
WHERE price <= ALL(SELECT price from tbl_product);
```

33

## Thêm bản ghi mới vào bảng

- Cú pháp:

```
INSERT INTO tenBang VALUES
(giaTri11, giaTri12,...)
(giaTri21, giaTri22,...)
...;
```

- Danh sách các giá trị trong cặp dấu ( ) phải phù hợp với các cột trong bảng về thứ tự và kiểu. Những cột không có giá trị cụ thể thì nhận giá trị DEFAULT
- Ví dụ: thêm 1 bản ghi vào bảng tbl\_bill

```
INSERT INTO tbl_bill VALUES
(1125, 'Nguyen Van An', '0903123123', '2014-10-23');
```

- Sử dụng INSERT IGNORE INTO: tránh lỗi gây ra khi bản ghi mới đưa vào bảng có giá trị trường khóa chính trùng với các bản ghi đã có

34

## Xóa bản ghi trong bảng

- Cú pháp:

```
DELETE FROM tenBang
WHERE dieuKienXoa;
```

- Ví dụ: xóa những mặt hàng trong bảng tbl\_product mà chưa bán được lần nào

```
DELETE FROM tbl_product
WHERE NOT EXISTS (SELECT productID
                  FROM tbl_bill_line
                  WHERE tbl_bill_line.productID =
                        tbl_product.productID);
```

35

## Sửa đổi giá trị của dữ liệu

- Cú pháp

```
UPDATE tenBang SET
tenCot1 = giaTriMoi,
tenCot2 = giaTriMoi,...
WHERE dieuKienSua;
```

- Ví dụ: Sửa thông tin mặt hàng “FUJITSU LH530” có nhà cung cấp mới là Viettel với giá 500

```
UPDATE tbl_product SET
supplier = 'Viettel',
price = 500
WHERE productName = 'FUJITSU LH530';
```

36

### 3. CƠ BẢN VỀ LẬP TRÌNH CSDL TRONG JAVA

---

37

## JDBC

- Java Database Conectivity: cung cấp các cách thức để kết nối và tương tác với CSDL
- JDBC có thể làm việc với tất cả các hệ quản trị CSDL
- JDBC cần có driver của CSDL để làm việc
- Cài đặt driver của MySQL trên Eclipse:
  - <http://dev.mysql.com/downloads/file.php?id=453815>
  - Chọn "No thanks, just start my download."
  - Giải nén vào một thư mục nào đó
  - Chép file mysql-connector-java-5.1.{xx}-bin.jar vào thư mục "C:\program files\java\jdk1.7.0\_{xx}\jre\lib\ext")
- Lưu ý: cần khởi động MySQL server để chương trình có thể kết nối và thao tác trên CSDL

38

## Sử dụng MySQL trên Eclipse

- Bước 1: Trên Project cần sử dụng MySQL, tạo thư mục mới, ví dụ MySqlConnection
- Bước 2: Chép file mysql-connector-java-5.1.{xx}-bin.jar vào thư mục trên
- Bước 3: Chọn file .jar ở trên, chuột phải, chọn Build Path → Add to Build Path
- Bước 4: Viết và chạy project như thông thường

39

## Các bước lập trình tương tác CSDL

- Bước 1: Tạo đối tượng Connection để kết nối tới CSDL

```
Connection conn = DriverManager.getConnection(
    "jdbc:mysql://localhost:8888/DBName", user, pass)
```

Trong đó:

- localhost: giữ nguyên hoặc thay bằng địa chỉ của máy chạy MySQL server
- DBName: thay bằng tên CSDL cần xử lý
- user và pass: tài khoản trên MySQL
- Bước 2: Tạo đối tượng Statement để tương tác với CSDL
- Bước 3: Tạo xâu chứa câu truy vấn
- Bước 4, 5: slide sau
- Bước 6: Đóng kết nối

```
conn.close();
```

40

## Bước 4: Thực thi truy vấn

Có 4 cách thực thi truy vấn bằng cách gọi các phương thức từ đối tượng Statement:

- `ResultSet executeQuery(String queryStr)`
  - Thực hiện truy vấn tìm kiếm, kết quả trả về lưu trong ResultSet
- `int executeUpdate(String updateStr)`
  - Thực hiện các câu lệnh thay đổi trên CSDL
  - Trả về số bản ghi đã thay đổi
- `boolean execute(String sqlStr)`
  - Thực thi câu lệnh SQL bất kỳ, kết quả truy vấn phức hợp
  - Ít dùng
- `int[] executeBatch()`
  - Thực thi nhiều câu truy vấn cùng lúc
  - Trả về mảng chứa số bản ghi đã thay đổi của các câu lệnh SQL

41

## Bước 5: Xử lý kết quả truy vấn

- Kết quả truy vấn được đặt trong đối tượng ResultSet
  - Với các phương thức thực thi câu truy vấn không trả về một ResultSet, có thể sử dụng phương thức `getResultSet()` của đối tượng Statement để lấy kết quả truy vấn
- Các phương thức trên ResultSet:
  - `boolean first()`: chuyển tới bản ghi đầu tiên. Trả về false nếu kết quả trả về không có bản ghi nào
  - `boolean last()`: chuyển tới bản ghi cuối cùng. Trả về false nếu kết quả trả về không có bản ghi nào
  - `boolean next()`: chuyển sang bản ghi tiếp theo. Trả về false nếu không có
  - `boolean previous()`: chuyển sang bản ghi trước đó. Trả về false nếu không có
  - `getType(String fieldName)`: lấy giá trị của trường `fieldName`, thay Type bằng kiểu dữ liệu tương ứng

42

## Truy vấn với PreparedStatement

```
PreparedStatement pstmt =
    conn.prepareStatement(sqlStr);
```

- Tái sử dụng câu lệnh SQL
  - Trong câu lệnh SQL `sqlStr`, sử dụng dấu ? cho các giá trị được tham số hóa
  - Mỗi lần thiết lập câu truy vấn cần truyền các tham số tương ứng:
 

```
setType(int seq, Type value)
```
  - Trong đó:
    - `seq`: là thứ tự của tham số trong câu lệnh SQL (đánh số từ 1)
    - `Type`: thay bằng kiểu dữ liệu tương ứng
  - Thực thi truy vấn `executeUpdate()`, `executeQuery()`. Lưu ý: không cần truyền đối số là câu truy vấn
- Truy vấn theo lô:
  - `void addBatch()`: thêm câu truy vấn vào lô
  - `int[] executeBatch()`: thực hiện lô truy vấn

43

## Ví dụ -

```
private static void queryProduct(Statement stmt) throws
    SQLException{
    System.out.println("Products in store:");
    String queryStr = "SELECT * FROM tbl_product;";
    ResultSet rs = stmt.executeQuery(queryStr);
    //Check empty result
    if(!rs.first())System.out.println("Have no record!");
    else{
        //display result if not empty
        do{
            String id = rs.getString("productID");
            String name = rs.getString("productName");
            String category = rs.getString("category");
            String supplier = rs.getString("supplier");
            int price = rs.getInt("price");
            System.out.println(id + ", " + name + ", " + category
                               + ", " + supplier + ", " + price);
        }while(rs.next());
    }
}
```

44

## Ví dụ (tiếp)

```
private static void addProduct(Connection conn){
    System.out.println("Added some products into store");
    String queryStr = "INSERT IGNORE INTO tbl_product
        VALUES(?,?,?, ?,?);";
    try(PreparedStatement addStmt =
        conn.prepareStatement(queryStr)
    ){
        addStmt.setString(1, "LAP003");
        addStmt.setString(2, "Macbook Pro 2014");
        addStmt.setString(3, "Laptop");
        addStmt.setString(4, "FPT");
        addStmt.setInt(5, 1300);
        addStmt.addBatch(); //add the statement for batch
        addStmt.setString(1, "LAP004");
        addStmt.setString(2, "Dell Vostro 3650");
        addStmt.setInt(5, 560);
        addStmt.addBatch();
        addStmt.executeBatch();
    }catch(SQLException e){e.printStackTrace();}
}
```

45

## Ví dụ (tiếp)

```
private static void delProduct(Statement stmt) throws
    SQLException{
    System.out.println("Delete all laptop from FPT!");
    String delStr = "DELETE FROM tbl_product WHERE supplier
        = 'FPT' AND category = 'Laptop';";

    int rtCode = stmt.executeUpdate(delStr);
    System.out.println("Number of deleted products: " +
        rtCode);
}
```

46

## Ví dụ (tiếp)

```
public static void main(String[] args){
    //try-with-resource to create connection to MySQL
    try(Connection conn = DriverManager.getConnection(
        "jdbc:mysql://localhost:8888/store", "root", "123456");
        Statement stmt = conn.createStatement();
    ){
        queryProduct(stmt);
        delProduct(stmt);
        queryProduct(stmt);
        addProduct(conn);
        queryProduct(stmt);
    }catch(SQLException e){
        e.printStackTrace();
    }
}
```

47

48