# Chương 6.
# Lập trình Cơ sở dữ liệu với Java
## (Database Programming with Java)

# 6.1. Tổng quan về Cơ sở dữ liệu quan hệ

# Khái niệm Dữ liệu và Cơ sở dữ liệu

- **Dữ liệu** (**data**): Là các thông tin của đối tượng (*ví dụ: người, vật, một khái niệm, sự việc*…) được lưu trữ trên máy tính. Có thể truy nhập vào dữ liệu để trích xuất ra các thông tin. Dữ liệu được mô tả dưới nhiều dạng khác nhau (các ký tự, ký số, hình ảnh, ký hiệu, âm thanh…). Mỗi cách mô tả như vậy gắn với một ngữ nghĩa nào đó.
- **CSDL** (**Database**) **:** Tập hợp dữ liệu được tổ chức có cấu trúc liên quan với nhau và được lưu trữ trong máy tính. CSDL được thiết kế, xây dựng cho phép người dùng lưu trữ dữ liệu, truy xuất thông tin hoặc cập nhật dữ liệu.
- **CSDL được tổ chức có cấu trúc**: Các dữ liệu lưu trữ có cấu trúc thành các **bản ghi** (*record*), các **trường dữ liệu** (*field*). Các dữ liệu lưu trữ có mối quan hệ (*relational*) với nhau. CSDL được cấu trúc để dễ dàng truy cập, quản lý và cập nhật dữ liệu.

# Mô hình dữ liệu quan hệ

Trong mô hình dữ liệu quan hệ, dữ liệu được biểu diễn dưới dạng bảng với các hàng và các cột:

- CSDL là tập hợp các bảng (còn gọi là quan hệ).
- Mỗi hàng là một bản ghi (record), còn được gọi là bộ (tuple).
- Mỗi cột là một thuộc tính, còn được gọi là trường (field)
- Dữ liệu trong hai bảng liên hệ với nhau thông qua các cột chung.

# Hệ quản trị Cơ sở dữ liệu

Hệ quản trị CSDL (DataBase Management System – DBMS) là các phần mềm giúp tạo các CSDL và cung cấp cơ chế lưu trữ, truy cập theo các mô hình CSDL.

*Ví dụ*:

- PostgreSQL
- SQL Server
- Microsoft Access
- Oracle

là các hệ quản trị CSDL điển hình cho mô hình quan hệ.

# Hệ quản trị Cơ sở dữ liệu quan hệ

- Hệ quản trị CSDL quan hệ (Relational DataBase Management System = RDBMS) là tập hợp các phần mềm cho phép tạo và thao tác với CSDL quan hệ. Nó là một dạng DBMS được sử dụng phổ biến nhất, trong đó tất cả dữ liệu được tổ chức chặt chẽ dưới dạng các bảng dữ liệu.
- Tất cả các thao tác trên CSDL đều diễn ra trên các bảng.
- Có nhiều đối tượng người dùng RDBMS như: quản trị CSDL, thiết kế CSDL, phân tích và thiết kế ứng dụng, cài đặt CSDL, người dùng cuối.

# 6.2. Câu lệnh SQL

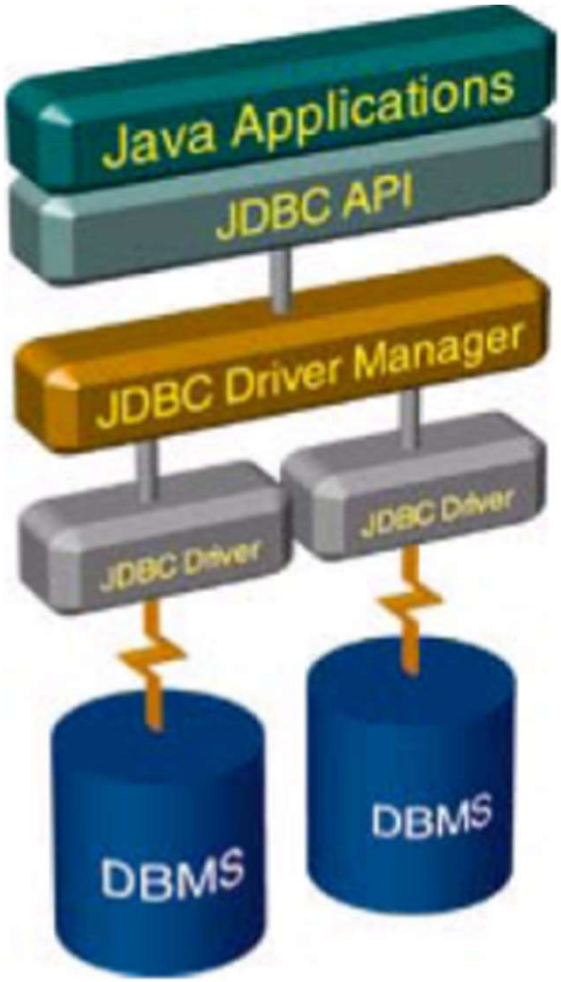
- CREATE
- SELECT
- INSERT
- UPDATE
- DELETE

# 6.3. Làm việc với JDBC

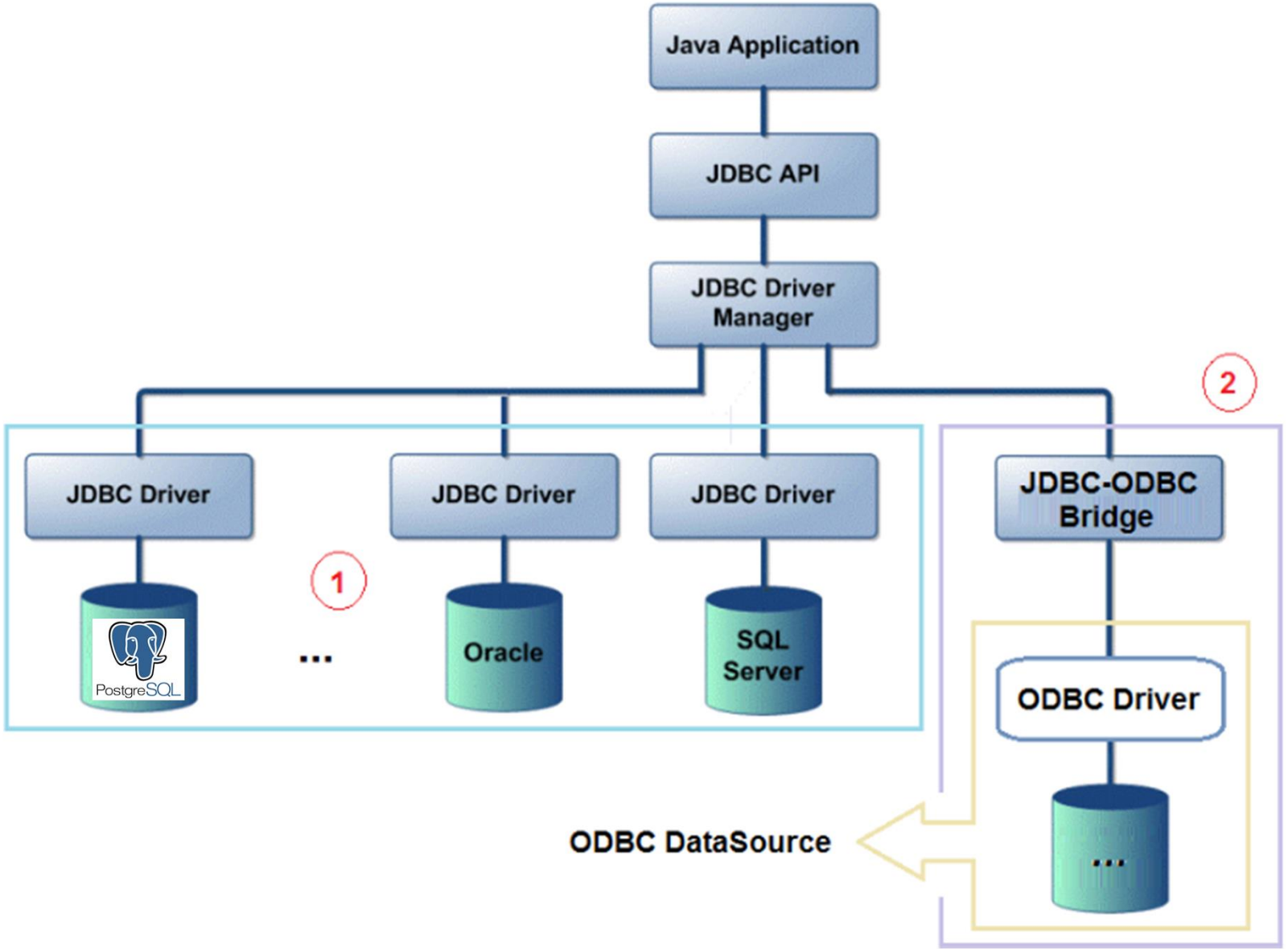
- **JDBC** (**J**ava **D**ata**B**ase **C**onnectivity)

  Cung cấp thư viện chuẩn để truy nhập CSDL quan hệ.

- JDBC bao gồm 2 phần:
  - JDBC API (Application Programming Interface)
  - JDBC Driver Manager, cung cấp các giao tiếp cụ thể với CSDL.
- Các lớp JDBC thuộc gói **java.sql**



# Làm việc với JDBC



# Các bước cơ bản sử dụng JDBC

1. Nạp các driver (Loading JDBC drivers).
2. Xác định URL của kết nối và thiết lập kết nối.
3. Khởi tạo một đối tượng Statement.
4. Thực hiện Statement.
5. Nhận dữ liệu từ kết quả thực thi.
6. Xử lý các kết quả.
7. Đóng kết nối.

# Loading JDBC drivers

☺ Sử dụng **Class.forName("DriverXYZ");**

*Ví dụ*:

- Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
- Class.forName("org.postgresql.Driver");
- Class.forName("oracle.jdbc.driver.OracleDriver");
- Class.forName(“com.mysql.jdbc.Driver");
- Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

**Lưu ý**: *Đối với SQL Server, kể từ SQLJDBC_4.0 trở về sau không cần nạp driver.*

# Thiết lập kết nối CSDL SQL Server

- **Các thông tin cần thiết để kết nối:**
  - URL
  - username, password
- **Cú pháp thiết lập kết nối:**

**Connection con = DriverManager.*getConnection*(url, “user”, “password”);**

- **Ví dụ:**

```
public static Connection getConnection() {
    String dbURL = "jdbc:sqlserver://localhost:1433; databaseName=Student Management;";
    String userName = "sa";
    String password = "12345";
    try {
        return DriverManager.getConnection(dbURL, userName, password);
    } catch (SQLException e) {
        System.out.println(e.getMessage());
        return null;
    }
}
```

# Đối tượng Statement

Một đối tượng **Statement** sẽ gửi câu lệnh SQL đến Database.

- Tạo một đối tượng **Statement**

```
Statement stmt = con.createStatement();
```

- Sử dụng phương thức **executeQuery** để thực hiện câu lệnh SELECT. Trả về là giá trị kiểu **ResultSet**

```
stmt.executeQuery( “SELECT * FROM XXX");
```

- Đối với các câu lệnh khởi tạo và chỉnh sửa bảng (DELETE, INSERT,…), ta sử dụng phương thức **executeUpdate**.

```
stmt.executeUpdate( "INSERT INTO XXX… “);
```

*Trả về giá trị kiểu int là số bản ghi được tác động*

# Đối tượng PreparedStatement

- **PreparedStatement** Interface là một interface con của Statement. Nó được sử dụng **để thực thi các truy vấn được tham số hóa**.
- Đối tượng PreparedStatement là một đối tượng mà biểu diễn một lệnh SQL được biên dịch trước. Tức là, một lệnh SQL được biên dịch trước và được lưu trữ trong một đối tượng PreparedStatement. Đối tượng này sau đó có thể được sử dụng để thực thi có hiệu quả Statement này **nhiều lần**.
- *Lợi thế*: sử dụng PreparedStatement giúp tăng hiệu suất, bởi vì truy vấn sẽ được biên dịch chỉ một lần.

# Sử dụng đối tượng PreparedStatement

**Bước 1**. Tạo 1 chuỗi truy vấn đến cơ sở dữ liệu dựa trên cấu trúc lệnh SQL đồng thời sử dụng ký tự "?" để xác định tham số muốn truyền vào.

**Bước 2**. Tạo đối tượng PreparedStatement

```
PreparedStatement pstmt = con.PreparedStatement("sql");
```

**Bước 3**. Truyền tham số truy vấn cho đối tượng ***PreparedStatement*** đã tạo ra ở bước trên thông qua ***setString***, ***setInt***, ***setFloat***, ***setBoolean***, …

**Bước 4**.

- Sử dụng phương thức **executeQuery** để thực hiện câu lệnh SELECT. Trả về là giá trị kiểu **ResultSet**

```
ResultSet result = pstmt.executeQuery( );
```

- Đối với các câu lệnh khởi tạo và chỉnh sửa bảng (DELETE, INSERT,…), ta sử dụng phương thức **executeUpdate()**. Trả về kiểu Int

```
pstmt.executeUpdate( );
```

# Sử dụng đối tượng CallableStatement

**CallableStatement được dùng khi muốn gọi stored procedures**

**Bước 1**. Tạo đối tượng CallableStatement

```
CallableStatement cstmt = con.prepareCall("{call procedureName(?, ?)}");
```

**Bước 2**. Truyền tham số truy vấn cho đối tượng **Callable*Statement*** đã tạo ra ở bước trên thông qua ***setString***, ***setInt***, ***setFloat***, ***setBoolean***, ...

**Bước 3**. Sử dụng phương thức **cstmt. Execute();**

# Đối tượng ResultSet

- Một đối tượng **ResultSet** là một bảng dữ liệu đại diện cho một tập hợp kết quả của cơ sở dữ liệu, được tạo ra bằng cách thực hiện một câu lệnh truy vấn cơ sở dữ liệu.
- Việc truy cập vào dữ liệu trong đối tượng **ResultSet** thông qua một con trỏ. Con trỏ này là một con trỏ trỏ đến một hàng dữ liệu trong **ResultSet**. Ban đầu, con trỏ ở trước hàng đầu tiên.

# Các kiểu của ResultSet

- **TYPE_FORWARD_ONLY**: Tập **ResultSet** không thể cuộn; con trỏ chỉ có thể di chuyển từ dòng đầu tiên đến dòng cuối cùng.
- **TYPE_SCROLL_INSENSITIVE**: Tập **ResultSet** có thể cuộn được; Con trỏ có thể cuộn tiến lùi đối với vị trí hiện tại, nhưng không nhạy cảm với các sự thay đổi dữ liệu.
- **TYPE_SCROLL_SENSITIVE**: Tập result có thể cuộn được; Con trỏ có thể cuộn tiến lùi đối với vị trí hiện tại, và nhạy cảm với sự thay đổi dữ liệu.
- Kiểu ngầm định của **ResultSet** là **TYPE_FORWARD_ONLY**.

# Các mức hoạt động của ResultSet

Có 2 mức hoạt động đồng thời **ResultSet** như sau:

- ✓ **CONCUR_READ_ONLY**: Xác định chế độ hoạt động đồng thời, kết quả lưu trong đối tượng **ResultSet** không được thay đổi.
- ✓ **CONCUR_UPDATABLE**: Xác định chế độ hoạt động đồng thời, kết quả lưu trong đối tượng **ResultSet** được thay đổi.

- Ngầm định là **CONCUR_READ_ONLY**.

# Sử dụng kiểu, hoạt động của ResultSet

- public Statement **createStatement**(int resultSetType, int resultSetConcurrency) throws SQLException
- public PreparedStatement **prepareStatement**(String sql, int resultSetType, int resultSetConcurrency) throws SQLException
- public CallableStatement **prepareCall**(String sql, int resultSetType, int resultSetConcurrency) throws SQLException

# Các phương thức di chuyển con trỏ trong tập ResultSet

| Method | Description |
|---|---|
| next() | di chuyển con trỏ đến dòng kế tiếp, trả về true nếu có dòng kế tiếp, false nếu đến cuối ResultSet. |
| previous() | di chuyển con trỏ đến dòng trước |
| first() | di chuyển con trỏ đến dòng đầu tiên |
| last() | di chuyển con trỏ đến dòng cuối cùng |
| beforeFirst() | di chuyển con trỏ đến vị trí trước dòng đầu tiên |
| afterLast() | di chuyển con trỏ đến vị trí sau dòng cuối cùng |
| getRow() | Trả về số của dòng hiện tại. Dòng đầu tiên là số 1, ... |
| absolute(int row) | di chuyển con trỏ đến dòng thứ row |
| relative(int rows) | di chuyển con trỏ tương đối với vị trí hiện tại của nó với số dòng là rows |

# Lấy giá trị của cột từ các hàng

- ResultSet có các phương thức getter để lấy giá trị cột từ hàng hiện tại.

  *Ví dụ*: **getString, getBoolean, getLong**, ...

- Ta có thể lấy giá trị bằng cách sử dụng các số chỉ số của cột hoặc bí danh hoặc tên của các cột.

- Sử dụng các chỉ số cột thường hiệu quả hơn. Các cột được đánh số từ 1 kể từ trái qua phải.

# Cập nhật các dòng trong các đối tượng ResultSet

- Với ResultSet được thiết lập là CONCUR_UPDATABLE, có thể cập nhật giá trị bằng cách sử dụng phương thức **`updateXXX()`**.
- Ta có thể cập nhật giá trị thông qua chỉ số hoặc tên của cột.
- Cuối cùng phải gọi phương thức **ResultSet.updateRow** để cập nhật cơ sở dữ liệu

# Cập nhật 1 dòng
# Updating a row

- Bước 1: Xác định vị trí con trỏ

```
Statement st =
    cn.createStatement(ResultSet.TYPE_SCROLL_SENSITI
    VE, ResultSet.CONCUR_UPDATABLE)
ResultSet rs = st.executeQuery("SELECT NAME,
    EMPLOEE_ID FROM EMPLOYEES");
rs.first();
```

- Bước 2: Cập nhật các cột

```
rs.updateInt(2,2345); //rs.update<Type>
```

- Bước 3: Committing the update

```
rs.updateRow();
```

# Chèn 1 dòng
# Inserting a row

- Step 1: Positioning the Cursor

```
Statement st =
    cn.createStatement(ResultSet.TYPE_SCROLL_SENSITI
    VE, ResultSet.CONCUR_UPDATABLE)
ResultSet rs = st.executeQuery("SELECT NAME,
    EMPLOEE_ID FROM EMPLOYEES");
rs.first();
```

- Step 2: Updating the columns

```
rs.update<Type>
```

- Step 3: inserting a row

```
rs.insertRow();
```

# Deleting a row

Step 1: Positioning the cursor

```
// Move the cursor to the last row of the result set
rs.last();
```

Step 2: Deleting the row

```
// Deleting the row from the result set
rs.deleteRow();
```

# Example: Inserting Rows

```
try {
      stmt = con.createStatement(
        ResultSet.TYPE_SCROLL_SENSITIVE,
   ResultSet.CONCUR_UPDATABLE);
      ResultSet uprs = stmt.executeQuery(
        "SELECT * FROM " + dbName + ".COFFEES");

      uprs.moveToInsertRow();

      uprs.updateString("COF_NAME", coffeeName);
      uprs.updateInt("SUP_ID", supplierID);
      uprs.updateFloat("PRICE", price);
      uprs.updateInt("SALES", sales);
      uprs.updateInt("TOTAL", total);

      uprs.insertRow();
      uprs.beforeFirst();

} catch (SQLException e ) {
```