# BÀI 2:
# Lập trình CSDL với JDBC

# Mục tiêu bài học

- Giới thiệu về JDBC
- Các class/interface trong JDBC
- Các bước kết nối và thao tác với CSDL trong ứng dụng Java
- Sử dụng thủ tục trong JDBC
- Thực hành

# Giới thiệu về JDBC

- Các ứng dụng Java không thể giao tiếp trực tiếp với CSDL, vì một CSDL chỉ có thể dịch và hiểu các câu lệnh SQL, CSDL không thể hiểu được các câu lệnh của ngôn ngữ Java.
- Ta cần một cơ chế để có thể dịch các câu lệnh Java thành các câu lệnh SQL.
- Kiến trúc JDBC cung cấp cơ chế để thực hiện giao tiếp giữa ứng dụng Java và CSDL.

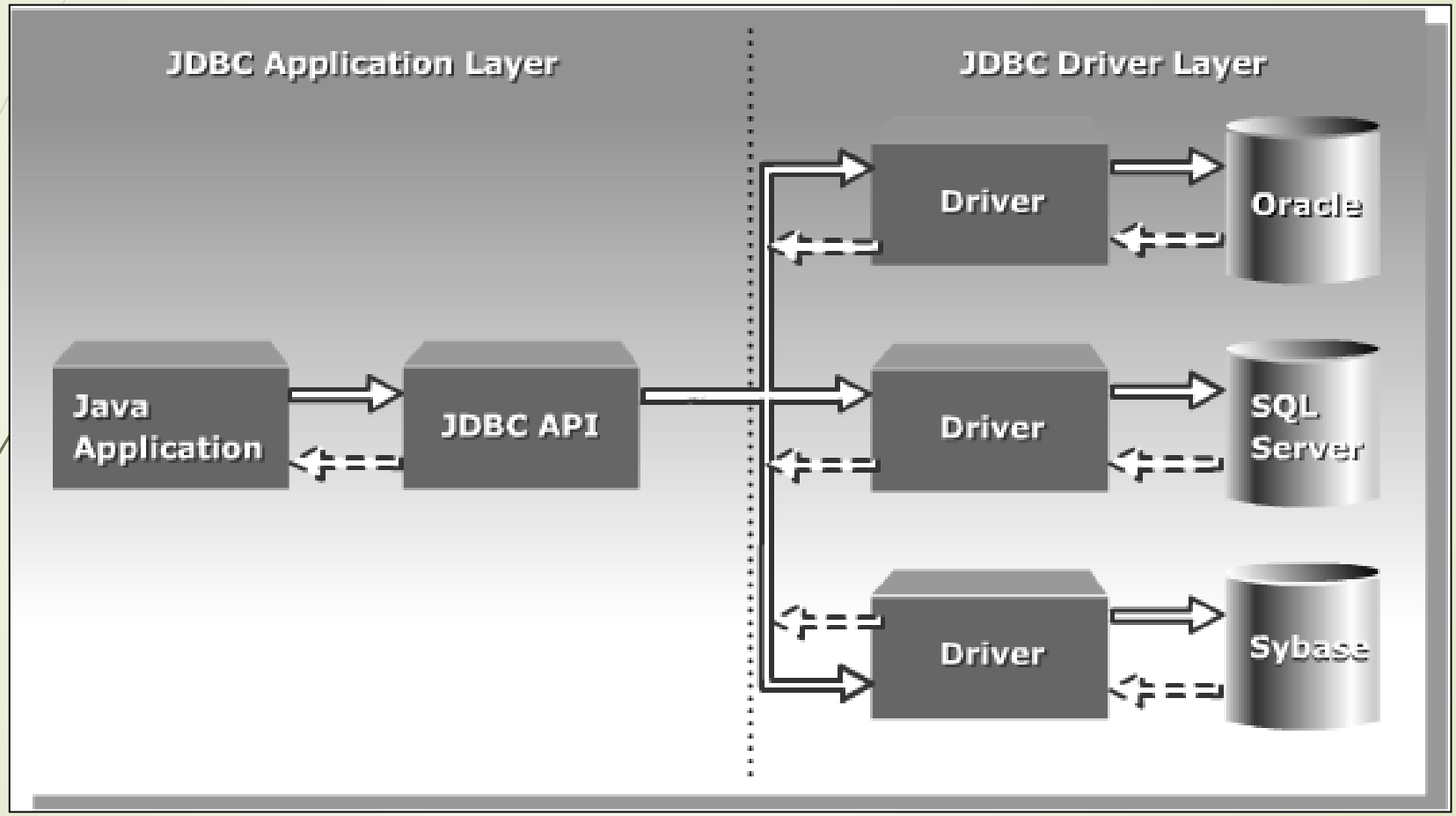
# Kiến trúc JDBC

- Kiến trúc JDBC bao gồm 2 layer như sau:

| JDBC application layer | JDBC driver layer |
| --- | --- |
| • Bao gồm ứng dụng Java sử dụng thư viện JDBC để tương tác với JDBC drivers. | • Hoạt động như một interface giữa ứng dụng Java và CSDL.<br>• Chứa một driver, VD như SQL Server driver hoặc Oracle driver, cái cho phép thiết lập kết nối đến CSDL. |

# Kiến trúc JDBC

- Kiến trúc JDBC.

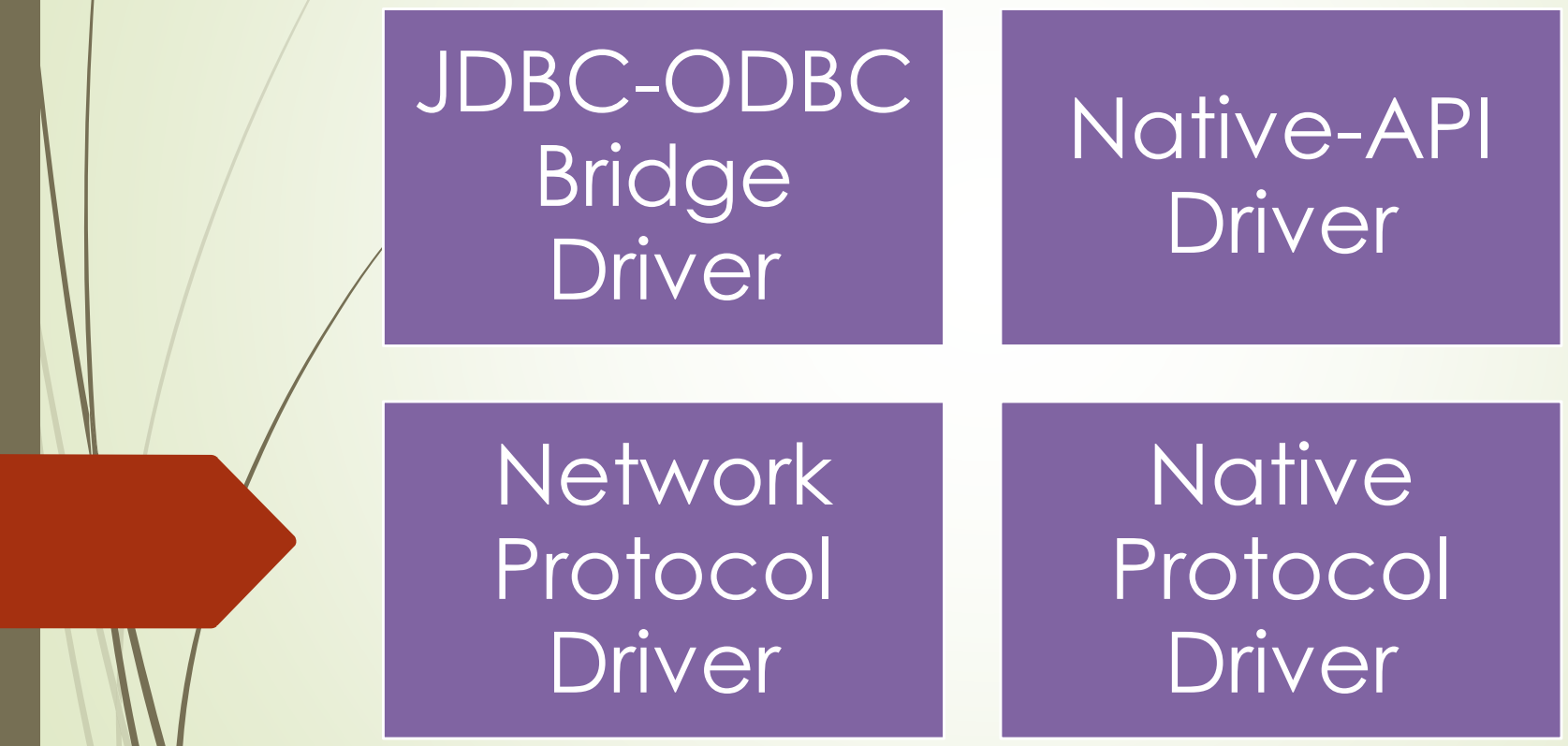


# JDBC API

- Ta cần sử dụng database drivers và JDBC API trong quá trình phát triển ứng dụng Java để có thể truy vấn hay lưu trữ dữ liệu trong CSDL.
- Các lớp/interface trong JDBC API:
  - Nằm trong các gói `java.sql` và `javax.sql`.
  - Thực hiện một số tác vụ, chẳng hạn như thiết lập và đóng kết nối đến CSDL, gửi request đến CSDL, truy vấn dữ liệu và cập nhật dữ liệu trong CSDL.

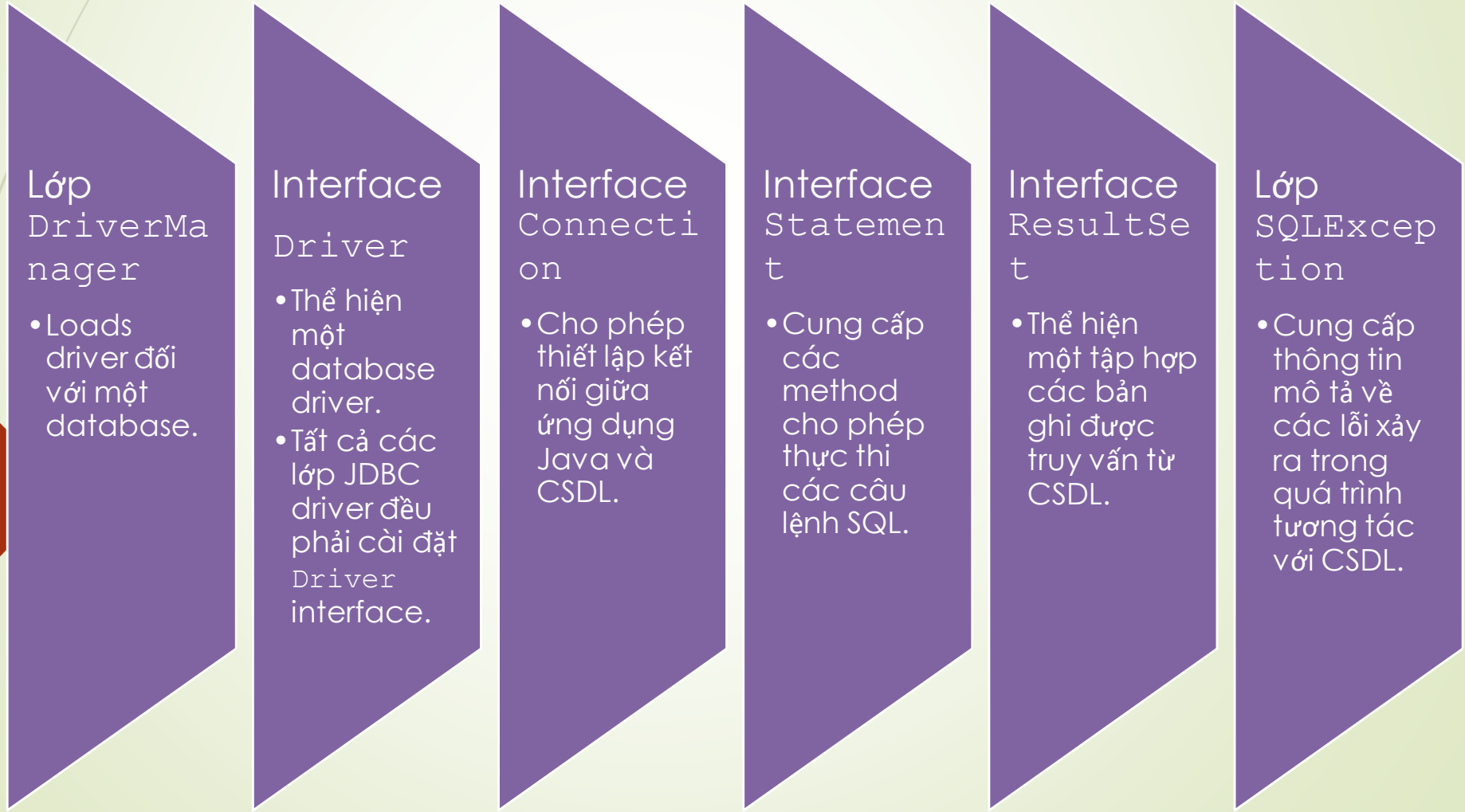
# Các kiểu JDBC Drivers

- JDBC hỗ trợ các kiểu drivers sau (4 kiểu):



# Giới thiệu JDBC API

- Các lớp và interfaces được sử dụng phổ biến trong JDBC API:



# Giới thiệu JDBC API

- Các bước để kết nối và thao tác với CSDL

# Loading a Driver

- Bước đầu tiên để phát triển một ứng dụng JDBC là load và đăng ký register JDBC driver bằng cách sử dụng driver manager.
- Có thể load và đăng ký driver bằng cách:

Sử dụng method `forName()` của lớp `java.lang.Class`.

Sử dụng static method `registerDriver()` của lớp `DriverManager`.

# Kết nối đến CSDL

- Để thiết lập kết nối từ ứng dụng Java đến CSDL, ta cần khởi tạo một đối tượng `Connection`.
- Ta có thể tạo nhiều đối tượng `Connection` objects trong một ứng dụng Java để có thể truy cập và thao tác với nhiều CSDL.
- Lớp `DriverManager` cung cấp method `getConnection()` để khởi tạo một đối tượng `Connection`.
- Method `getConnection()` có 3 phiên bản như sau:

```
public static Connection getConnection(String url)
```

```
public static Connection getConnection(String url,
                                       Properties info)
```

```
public static Connection getConnection(String url,
                                       String user, String password)
```

# Kết nối đến CSDL

- Đoạn mã minh họa khởi tạo kết nối đến CSDL bằng phương thức `getConnection()` với một tham số:

```
String url =
"jdbc:sqlserver://localhost;user=MyUserName;passw
ord=password@123";
Connection con =
DriverManager.getConnection(url);
```

- Cú pháp của JDBC URL được truyền cho method `getConnection()`:

```
<protocol>:<subprotocol>:<subname>
```

- Một JDBC URL sẽ bao gồm 3 thành phần sau:
  - `protocol`
  - `subprotocol`
  - `subname`

# Tạo và thực thi các câu lệnh JDBC

- Ta cần tạo một đối tượng `Statement` để gửi các yêu cầu đến CSDL và nhận kết quả trả về.
- Đoạn mã minh họa khởi tạo một đối tượng `Statement`:

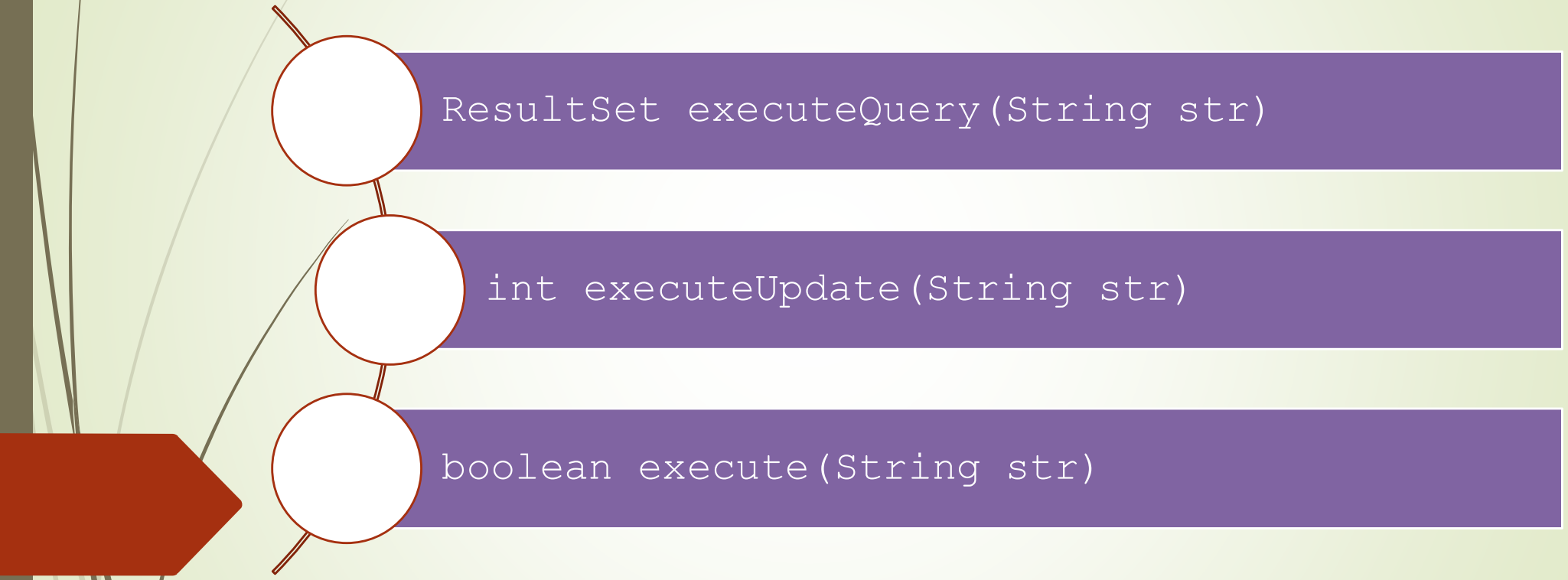
```
Connection
con=DriverManager.getConnection("jdbc:sqls
erver://sqlserver01;databaseName=Library;u
ser=user1;password=password#1234");
Statement stmt = con.createStatement();
```

- Các câu lệnh SQL statements không chứa các runtime parameters còn được gọi là các static SQL statements.

- Ta có thể gửi các câu lệnh SQL cho một CSDL bằng cách sử dụng đối tượng `Statement`.

# Tạo và thực thi các câu lệnh JDBC

- Interface `Statement` chứa các phương thức sau để gửi câu lệnh static SQL đến CSDL:



# Tạo và thực thi các câu lệnh JDBC

- Ta có thể sử dụng các câu lệnh DML, `INSERT`, `UPDATE`, và `DELETE` trong các ứng dụng Java để cập nhật dữ liệu trong bảng CSDL.
- Ta cũng có thể sử dụng các câu lệnh Data Definition Language (DDL), như `CREATE`, `ALTER`, và `DROP`, trong các ứng dụng Java để định nghĩa hoặc thay đổi cấu trúc của các database objects.
- Ta có thể truy vấn dữ liệu trong bảng bằng cách sử dụng lệnh `SELECT`. Lệnh `SELECT` được thực thi bằng phương thức `executeQuery()` và trả về một tập bản ghi, được lưu trữ trong một đối tượng `ResultSet`.
- Đoạn code minh họa truy vấn dữ liệu từ bảng `Authors`:

```
String str = "SELECT * FROM Authors";
Statement stmt = con.createStatement();
ResultSet rs =
stmt.executeQuery(str);
```

# Tạo và thực thi các câu lệnh JDBC

- Ta có thể chèn các dòng vào bảng bằng cách thực thi lệnh `INSERT`.
- Sau đó ta gọi method `executeUpdate()` để thực thi câu lệnh SQL.
- Đoạn code minh họa chèn dữ liệu (cứng) vào bảng `Authors`:

```
String str = " INSERT INTO Authors (au_id,
au_name, phone, address, city, state, zip)
VALUES ('A004', 'Ringer Albert', '8018260752',
' 67 Seventh Av. ', 'Salt Lake City', 'UT',
'100000078')";
Statement stmt = con.createStatement();
int count = stmt.executeUpdate(str);
```

# Tạo và thực thi các câu lệnh JDBC

- Ta có thể cập nhật dữ liệu trong bảng bằng cách sử dụng câu lệnh `UPDATE`.
- Code minh họa cập nhật dữ liệu trong bảng `Authors`:

```
String str = "UPDATE Authors SET
address='10932 Second Av.' where au_id=
'A001'";
Statement stmt = con.createStatement();
int count = stmt.executeUpdate(str);
```

# Tạo và thực thi các câu lệnh JDBC

- Để xóa dữ liệu trong bảng, ta sử dụng lệnh `DELETE`.
- Code minh họa xóa dữ liệu trong bảng `Authors` :

```
String str = "DELETE FROM Authors WHERE
au_id='A005'";
Statement stmt = con.createStatement();
int count = stmt.executeUpdate(str);
```

# Tạo và thực thi các câu lệnh JDBC

- Ta có thể dùng lệnh `CREATE TABLE` để tạo ra một bảng mới trong CSDL.
- Ví dụ:

```
String str="CREATE TABLE Publishers"
+"(pub_id VARCHAR(5),"
+"pub_name VARCHAR(50),"
+"phone INTEGER,"
+"address VARCHAR(50), "
+"city VARCHAR(50), "
+"ZIP VARCHAR(20))";
Statement stmt=con.createStatement();
stmt.execute(str);
```

# Tạo và thực thi các câu lệnh JDBC

- Để sửa cấu trúc bảng, chẳng hạn như thêm sửa xóa các cột trong bảng, ta dùng lệnh `ALTER TABLE`.
- Code minh họa thêm cột vào bảng `Books`:

```
String str="ALTER TABLE Books ADD price
INTEGER";
Statement stmt=con.createStatement();
stmt.execute(str);
```

- Để xóa bảng, ta dùng lệnh `DROP TABLE`.

# Xử lý SQL Exceptions

- Gói `java.sql` cung cấp lớp `SQLException`, lớp này kế thừa từ lớp `java.lang.Exception`.
- Lớp `SQLException` cung cấp các thông tin mô tả về lỗi như sau:
  - Error message
  - Error code
  - SQL state
- Các phương thức của lớp `SQLException`:

```
int getErrorCode()
```

```
String getSQLState()
```

```
SQLException getNextException()
```

# Xử lý Result Sets

- Khi ta thực thi câu truy vấn (select) để lấy dữ liệu trong bảng CSDL về chương trình Java, kết quả của câu truy vấn sẽ là một tập hợp các bản ghi, được lưu trong một đối tượng `ResultSet`.
- Một đối tượng `ResultSet` khi được khởi tạo sẽ duy trì một con trỏ (cursor) để phục vụ cho việc theo dõi vị trí hiện tại, và duyệt qua các dòng dữ liệu được lưu trữ trong `ResultSet`.
- Mặc định cursor sẽ trỏ đến vị trí trước bản ghi đầu tiên trong `ResultSet`.

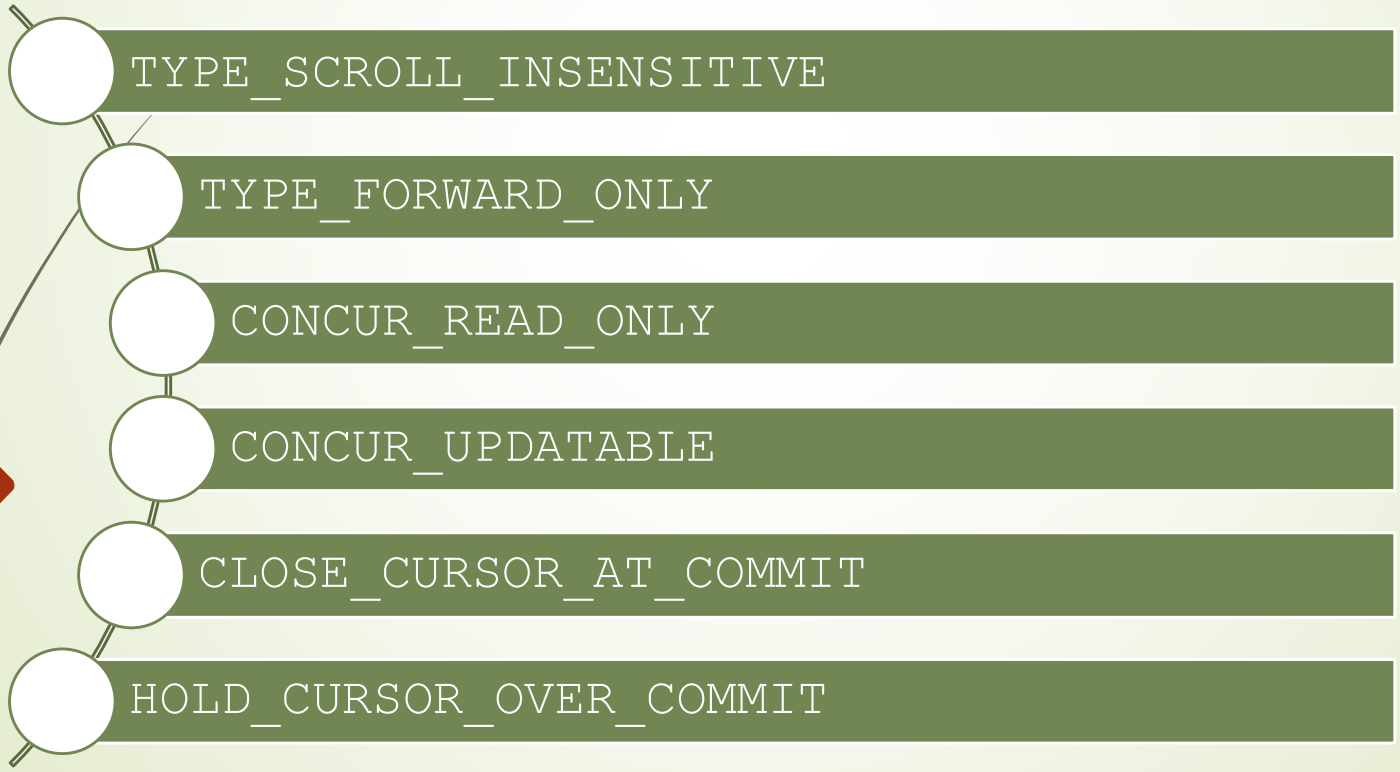
# Các kiểu Result Sets

- Có nhiều kiểu ResultSet khác nhau:

| Read only | Forward only | Scrollable | Updatable |
|---|---|---|---|
| • Chỉ cho phép đọc các dòng trong `ResultSet`. | • Chỉ cho phép duyệt qua các bản ghi trong resultset theo một chiều từ đầu về cuối. | • Cho phép cuộn xuôi ngược giữa các bản ghi trong result set. | • Cho phép cập nhật các bản ghi trong resultset. |

# Các kiểu Result Sets

- Ta có thể thiết lập kiểu của `ResultSet` trong khi gọi phương thức `createStatement()` của đối tượng `Connection`.
- Các hằng số của interface `ResultSet`:



# Các phương thức của ResultSet Interface

- Các phương thức phổ biến của `ResultSet`:
  - `boolean first()`
  - `boolean isFirst()`
  - `void beforeFirst()`
  - `boolean isBeforeFirst()`
  - `boolean last()`
  - `boolean isLast()`
  - `void afterLast()`
  - `boolean isAfterLast()`
  - `boolean previous()`
  - `boolean absolute(int i)`
  - `boolean relative(int i)`

# Các phương thức của ResultSet Interface

- Ta có thể tạo ra một scrollable result set.
- Ví dụ minh họa tạo ra một read-only scrollable result set:

```
Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIV
E, ResultSet.CONCUR_READ_ONLY);
ResultSet rs=stmt.executeQuery ("SELECT * FROM
Authors");
```

- Đoạn code minh họa kiểm tra xem liệu result set cursor có phải đang ở vị trí trước bản ghi đầu tiên hay không:

```
if(rs.isBeforeFirst()==true)
System.out.println("Result set cursor is before the
first row in the result set");
```

- Để di chuyển giữa các dòng trong result set, ta có thể sử dụng các method first(), next(), previous(), last(), hoặc sử dụng các method relative(), `absolute()`.

# Các phương thức của ResultSet Interface

- JDBC cho phép ta tạo ra các updatable result set, để cho phép ta cập nhật các dòng trong result set, rồi đồng bộ với dữ liệu trong bảng.
- Một số methods được sử dụng với updatable result set:
  - `void updateRow()`
  - `void insertRow()`
  - `void deleteRow()`
  - `void updateString(int columnIndex, String x)`
  - `void updateString(String  columnLabel, String x)`
  - `void updateInt(int columnIndex, int x)`
  - `void updateInt(String columnLabel, int x)`

# Các phương thức của PreparedStatement Interface

- `PreparedStatement` interface kế thừa các methods sau:
  - `ResultSet executeQuery()`
  - `int executeUpdate()`
  - `boolean execute()`
- Ta có thể sử dụng ví dụ như sau để tạo một đối tượng PreparedStatement:

```
stat=con.prepareStatement("SELECT * FROM Authors
WHERE au_id = ?");
```

- Phương thức `prepareStatement()` của đối tượng `Connection` sẽ cần truyền tham số là một câu lệnh SQL.

# Các phương thức của PreparedStatement Interface

- Có thể thiết lập các giá trị cho prepared statement bằng cách gọi các phương thức `setXXX()`, trong đó `XXX` là kiểu dữ liệu của tham số. Ví dụ:

```
stat.setString(1,"A001");
ResultSet result=stat.executeQuery();
```

- Một số methods của `PreparedStatement` interface:
  - `void setByte(int index, byte val)`
  - `void setBytes(int index, byte[] val)`
  - `void setBoolean(int index, boolean val)`
  - `void setDouble(int index, double val)`
  - `void setInt(int index, int val)`
  - `void setLong(int index, long val)`
  - `void setFloat(int index, float val)`
  - `void setShort(int index, short val)`
  - `void setString(int index, String val)`

# Truy vấn dữ liệu

- Ví dụ minh họa truy vấn dữ liệu trong bảng `Books` bằng cách sử dụng đối tượng `PreparedStatement`:

```
String str = "SELECT * FROM Books WHERE au_id = ?";
PreparedStatement ps= con.prepareStatement(str);
ps.setString(1, "A001");
ResultSet rs=ps.executeQuery();
while(rs.next())
{
     System.out.println(rs.getString(1) + " " +
rs.getString(2));
}
```

# Thêm bản ghi

- Ví dụ sử dụng đối tượng `PreparedStatement` để chèn thêm một bản ghi mới vào bảng `Authors`:

```
String str = "INSERT INTO Authors (au_id, au_name)
VALUES (?,?)";
PreparedStatement ps = con.prepareStatement(str);
ps.setString(1, "1001");
ps.setString(2, "Abraham White");
int rt=ps.executeUpdate();
```

# Cập nhật và xóa bản ghi

- Ví dụ cập nhật dữ liệu trong bảng `Authors` bằng đối tượng `PreparedStatement`:

```
String str = "UPDATE Authors SET state= ? WHERE
city= ? ";
PreparedStatement ps = con.prepareStatement(str);
ps.setString(1, "CA");
ps.setString(2, "Oakland");
int rt=ps.executeUpdate();
```

- Ví dụ xóa dữ liệu trong bảng `Authors` bằng cách sử dụng `PreparedStatement`:

```
String str = "DELETE FROM Authors WHERE au_name= ?
"; PreparedStatement ps = con.prepareStatement
(str); ps.setString(1, "Abraham White");
int rt=ps.executeUpdate();
```

# Tạo và gọi các Stored Procedures trong JDBC

- Gói `java.sql` cung cấp `CallableStatement` interface:
  - Chứa các phương thức để cho phép gọi các thủ tục lưu trong CSDL (database stored procedures).
  - Được kế thừa từ `PreparedStatement` interface.

# Tạo Stored Procedures

- Stored procedures:
  - Có thể được tạo bằng các ứng dụng Java, sử dụng JDBC.
  - Có thể thuộc về 1 trong 2 loại, parameterized và non parameterized.
- Ta có thể sử dụng method `executeUpdate()` để thực thi câu lệnh `CREATE PROCEDURE`.
- Đoạn code minh họa việc tạo thủ tục không có tham số:

```
String str = "CREATE PROCEDURE authors_info "
+"AS "
+ "SELECT au_id,au_name "
+ "FROM authors "
+ "WHERE city = 'Oakland' "
+ "ORDER BY au_name";
Statement stmt=con.createStatement();
int rt=stmt.executeUpdate(str);
```

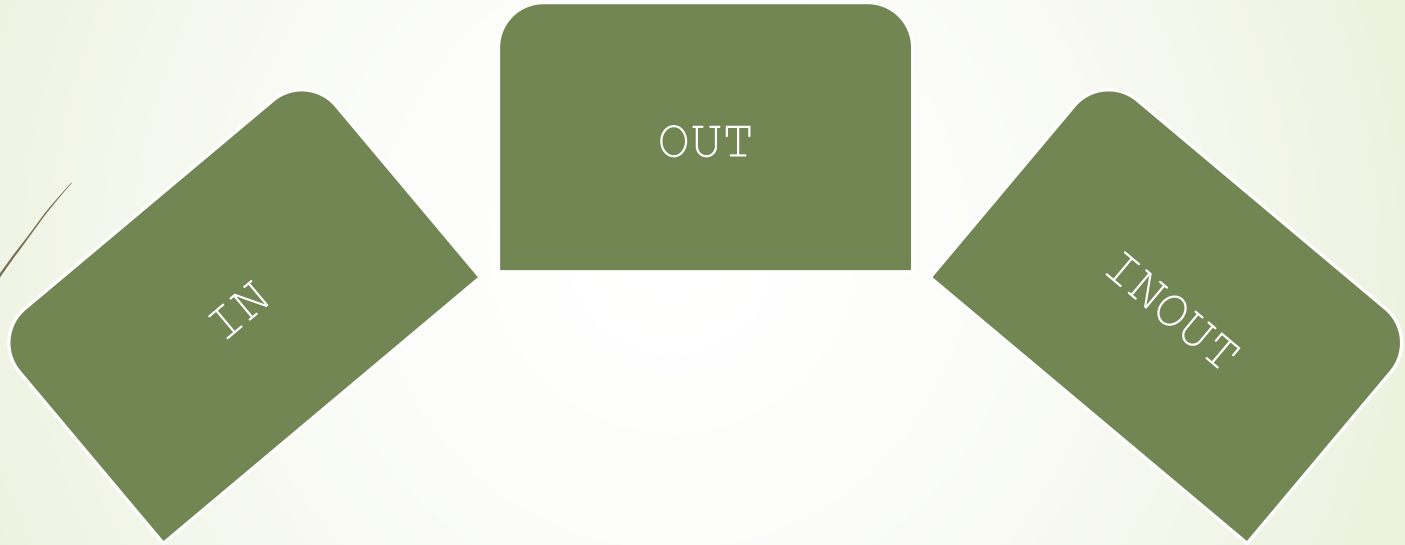
# Tạo Stored Procedures

- Đoạn code minh họa tạo một parameterized stored procedure:

```
str = " CREATE PROCEDURE authors_info_prmtz
@auth_id varchar(15) ,@auth_name varchar(20)
output, @auth_city varchar(20) output,@auth_state
varchar(20) output "
+ " AS "
+ " SELECT @auth_name=au_name, @auth_city=city,
@auth_state=state "
+ " FROM authors "
+ " WHERE au_id=@auth_id ";
Statement stmt=con.createStatement();
int rt=stmt.executeUpdate(str);
```

# Tạo Stored Procedures

- Tham số (parameter) của một stored procedure có thể thuộc 1 trong 3 dạng sau:



# Gọi một Stored Procedure không có tham số

- `Connection` interface cung cấp phương thức `prepareCall()` được dùng để khởi tạo một đối tượng `CallableStatement`.
- Có 3 phiên bản như sau:
  - `CallableStatement prepareCall(String str)`
  - `CallableStatement prepareCall(String str, int resSetType, int resSetConcurrency)`
  - `CallableStatement prepareCall(String str, int resSetType, int resSetConcurrency, int resSetHoldability)`
- Cú pháp để gọi thủ tục không có tham số:

```
exec <procedure_name>
```

# Gọi một Stored Procedure không có tham số

- Đoạn code minh họa gọi thủ tục:

```
String str = "exec authors_info";
    CallableStatement cstmt = con.prepareCall(str);
    ResultSet rs = cstmt.executeQuery();
    while (rs.next())
{
        System.out.println(" Author Id : " +
rs.getString(1) + "\t");
        System.out.println(" Author Name : " +
rs.getString(2) + "\t");
    }
```

# Gọi một Stored Procedure có tham số

- Thủ tục cho phép ta khai báo các tham số đầu vào (IN parameter), cũng như các tham số đầu ra (OUT parameter)
- Cú pháp để nhận kết quả tham số đầu ra từ thủ tục:

```
{[? =] call <procedure_name>
[<parameter1>,<parameter2>, ., <parameterN>]}
```

- Cú pháp để gọi một stored procedure với tham số:

```
{ call <procedure_name>(?) };
```

- Cú pháp để thiết lập giá trị cho `IN` parameter:

```
<CallableStatement_object>.setInt(<value>);
```

# Gọi một Stored Procedure có tham số

- Phương thức `registerOut()` được dùng để đăng ký một output parameters.
- Cú pháp của phương thức `registerOut()` :
  - `registerOut(int index, int stype)`
  - `registerOut(int index, int stype, int scale)`

# Tổng kết

- Giới thiệu về JDBC
- Các class/interface trong JDBC
- Các bước kết nối và thao tác với CSDL trong ứng dụng Java
- Sử dụng thủ tục trong JDBC
- Thực hành