Web-based programming with Java

CSDL NÂNG CAO

ThS. Nguyễn Nghiệm
0913.745.789 - NghiemN@fpt.edu.vn

NỘI DUNG

- Giới thiệu CSDL EShopV10
- Câu lệnh SELECT
  - Sắp xếp tập kết quả truy vấn
  - Lọc dữ liệu
  - Kết nối nhiều bảng
  - Thống kê số liệu
- JDBC nâng cao
  - PreparedStatement
  - CallableStatement
  - Transaction

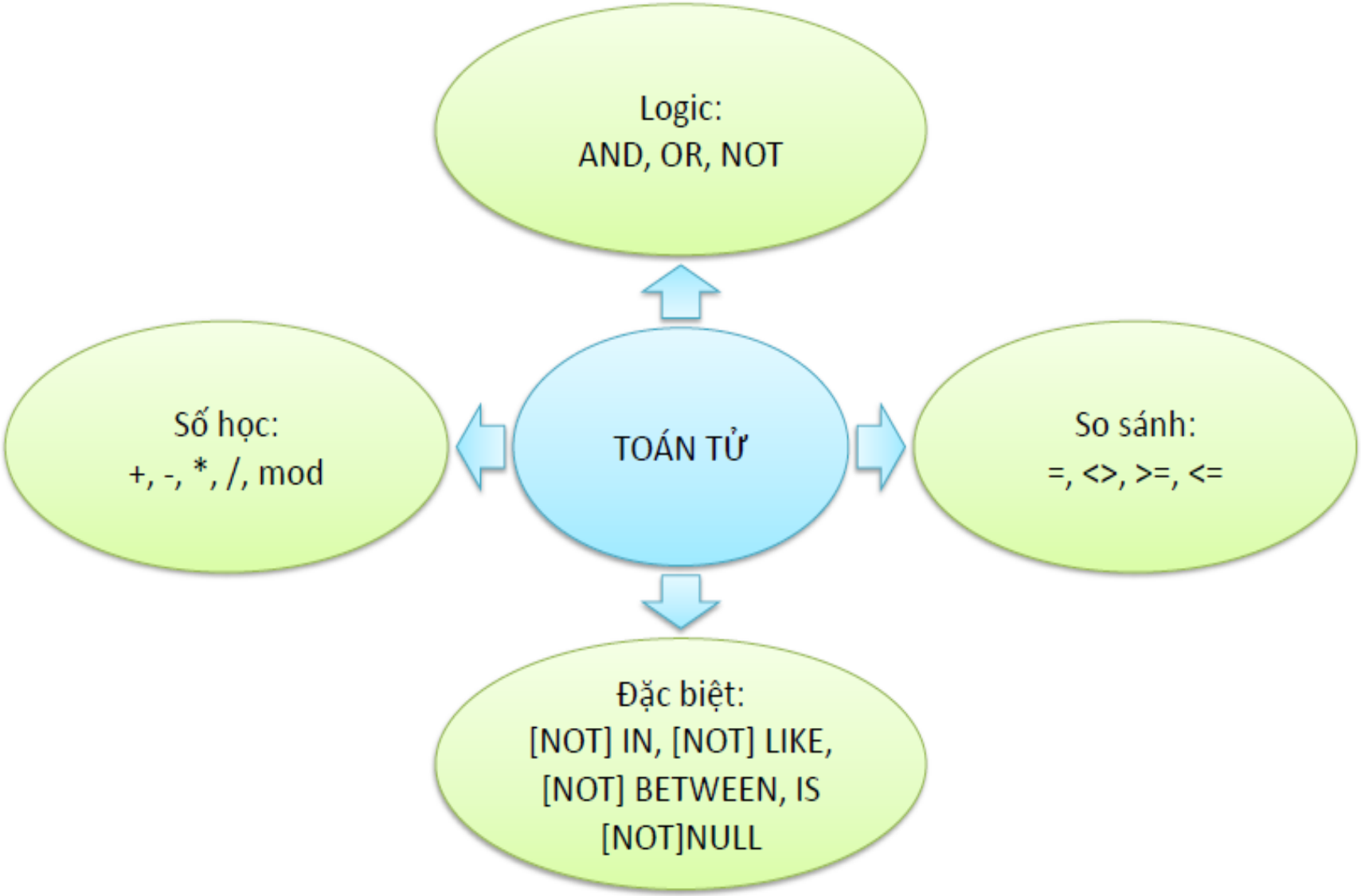
CÂU LỆNH TRUY VẤN

```
SELECT <danh sách cột>
       FROM <bảng>
              JOIN <bảng 1> ON <ĐK JOIN>
              JOIN <bảng 2> ON <ĐK JOIN>...
       WHERE <ĐK lọc dữ liệu>
       GROUP BY <biểu thức nhóm>
              HAVING<ĐK lọc nhóm>
       ORDER BY <biểu thức sắp sếp> [ASC] | DESC
```

Ví dụ

```
SELECT * FROM Courses
        WHERE Schoolfee <= 3000000
                AND YEAR(StartDate)=2017
                AND Status=0
        ORDER BY Schoolfee
```

TOÁN TỬ


Logic:
AND, OR, NOT
Số học:
+, -, *, /, mod
TOÁN TỬ
So sánh:
=, <>, >=, <=
Đặc biệt:
[NOT] IN, [NOT] LIKE,
[NOT] BETWEEN, IS
[NOT]NULL

TOÁN TỬ ĐẶC BIỆT

- [NOT] LIKE
  - **%**: đại diện một nhóm ký tự bất kỳ
  - _: đại diện một ký tự bất kỳ
  - **[xyz]:** một ký tự thuộc nhóm x, y hoặc z
  - **[^xyz]**: một ký tự không thuộc nhóm x, y và z

Name **LIKE** N'Nguyễn**%**'

Name **LIKE** '**%**anh**%**': Chứa chuỗi 'anh' ?

Name **LIKE** '**%**Tuấn': Kết thúc bởi chuỗi 'Tuấn' ?

Name **LIKE** 'Pha**[nm]%**': Bắt đầu bởi 'Phan' hoặc 'Pham'?

Name **LIKE** '**%**AB_C**%**': Chứa chuỗi 'ABxC', với x là ký tự bất kỳ?

TOÁN TỬ ĐẶC BIỆT

- [NOT] IN
  - SELECT* FROM Products WHERE
    - Id **IN** (1005, 1007, 1018, 1025)
    - CategoryId **IN** (SELECT Id FROM Categories)
- [NOT] BETWEEN
  - SELECT* FROM Courses WHERE
    - StartDate **BETWEEN** ’12/31/2016’ AND ’12/31/2017’
    - Schoolfee NOT **BETWEEN** 20 **AND** 80
- IS [NOT] NULL
  - SELECT * FROM Products WHERE
    - StartDate **IS NULL**

HÀM XỬ LÝ THỜI GIAN

- Truy vấn theo thời gian
  - SELECT* FROM Courses WHERE
    - StartDate < **GetDate**()
    - **Year**(StartDate) = 2014
    - **DatePart**(Quarter, StartDate) = 3
- Hàm xử lý thời gian thường dùng
  - **GetDate**() lấy thời gian hiện tại
  - **DatePart**(<thành phần>, <thời gian>): lấy thành phần thời gian
  - **Year**(<thời gian>): lấy năm
  - **Month**(<thời gian>): lấy tháng
  - **Hour**(<thời gian>): lấy giờ
  - **Minute**(<thời gian>): lấy phút
  - **Second**(<thời gian>): lấy giây

KẾT NỐI NHIỀU BẢNG

```
SELECT
    c.Name AS Loại,
    p.Name AS [Hàng Hóa],
    p.UnitPrice AS [Đơn giá]
FROM Categories AS c
    INNER JOIN Products AS p ON c.Id = p.CategoryId
```

Results | Messages

Loại	Hàng Hóa	Đơn giá
Laptop	Dell 1014	2000
Laptop	Acer Xyz	700
Mobile	iPhone ...	1500

PHÂN LOẠI KẾT NỐI

INNER JOIN

```
SELECT * FROM Categories AS c
    INNER JOIN Products AS p
    ON c.Id = p.CategoryId
```

LEFT OUTER JOIN

```
SELECT * FROM Categories AS c
    LEFT OUTER JOIN Products AS p
    ON c.Id = p.CategoryId
```

RIGHT OUTER JOIN

```
SELECT * FROM Categories AS c
    RIGHT OUTER JOIN Products AS p
    ON c.Id = p.CategoryId
```

FULL OUTER JOIN

```
SELECT * FROM Categories AS c
    FULL OUTER JOIN Products AS p
    ON c.Id = p.CategoryId
```

INNER JOIN

```
SELECT
    c.Name AS Loại,
    p.Name AS [Hàng Hóa],
    p.UnitPrice AS [Đơn giá]
FROM Categories AS c
    INNER JOIN Products AS p ON c.Id = p.CategoryId
```

	Loại	Hàng Hóa	Đơn giá
1	Laptop	Dell 1014	2000
2	Laptop	Acer Xyz	700
3	Mobile	iPhone ...	1500

LEFT OUTER JOIN

```
SELECT
    c.Name AS Loại,
    p.Name AS [Hàng Hóa],
    p.UnitPrice AS [Đơn giá]
FROM Categories AS c
    LEFT OUTER JOIN Products AS p ON c.Id = p.CategoryId
```

	Loại	Hàng Hóa	Đơn giá
1	Fashion	NULL	NULL
2	Laptop	Dell 1014	2000
3	Laptop	Acer Xyz	700
4	Mobile	iPhone 5S	1500

THỐNG KÊ – GROUP BY

- Tổng hợp số liệu theo từng nhóm.
  - Nhóm - GROUP BY
  - Tổng hợp thông tin theo nhóm
    - SUM()
    - AVG()
    - COUNT()
    - MIN()
    - MAX()

```
SELECT
      Name AS [Tên HH],
      COUNT(Id) AS [Số Lượng],
      MAX(UnitPrice) AS [Giá Cao Nhất],
      AVG(UnitPrice) AS [Giá Trung Bình],
      MAX(ProductDate) AS [Hàng Mới Nhất]
FROM Products
GROUP BY Name
```

Results | Messages

	Tên HH	Số Lượng	Giá Cao Nhất	Giá Trung Bình	Hàng Mới Nhất
1	Acer Xyz	1	700	700	2010-03-09
2	Dell 1014	1	2000	2000	2013-01-01
3	iPhone 5S	1	1500	1500	2013-09-25

THỐNG KÊ - HAVING

```
SELECT
     Name AS [Tên HH],
     COUNT(Id) AS [Số Lượng],
     MAX(UnitPrice) AS [Giá Cao Nhất],
     AVG(UnitPrice) AS [Giá Trung Bình],
     MAX(ProductDate) AS [Hàng Mới Nhất]
FROM Products
GROUP BY Name
HAVING MAX(UnitPrice) > 1000
```

JDBC Nâng cao

- PreparedStatement
  - Thay thế Statement
- Callable Statement
  - Làm việc với stored procedure
- Transaction
  - Điều khiển khối giao dịch thao tác dữ liệu

PREPARED STATEMENT

```
Class.forName(driver);
Connection connection = DriverManager.getConnection(url, user, pass);
String sql = "INSERT INTO Categories(Name, NameVN) VALUES(?, ?)";
PreparedStatement preparedStatement = connection.prepareStatement(sql);
preparedStatement.setString(1, "Remote Control");
preparedStatement.setString(2, "Điều khiển từ xa");
preparedStatement.executeUpdate();
connection.close();
```

Ưu điểm của PreparedStatement

- Làm việc được với dữ liệu nhị phân
- Tránh lỗi dấu nháy đơn
- Unicode
- Tránh được hack bằng SQL injection
- Mã viết rõ ràng
- Nhanh hơn khi thực hiện câu lệnh nhiều lần

PREPARED STATEMENT

```
Class.forName(driver);
Connection connection = DriverManager.getConnection(url, user, pass);
String sql = "SELECT * FROM Products WHERE UnitPrice BETWEEN ? AND ?";
PreparedStatement preparedStatement = connection.prepareStatement(sql);
preparedStatement.setDouble(1, 5);
preparedStatement.setDouble(2, 10);
ResultSet resultSet = preparedStatement.executeQuery();
while (resultSet.next()) {
        String name = resultSet.getString("Name");
        double unitPrice = resultSet.getDouble("UnitPrice");
        System.out.println(name + " = "+ unitPrice);
}
connection.close();
```

STORED PROCEDURE

- PROC là một thủ tục thực hiện một nhiệm vụ cụ thể nào đó.
- Tham số của PROC gồm 2 loại
  - Tham số vào (INPUT): cung cấp dữ liệu đủ để PROC hoạt động.
  - Tham số ra (OUTPUT): nắm giữ kết quả tính toán trong PROC.

STORED PROCEDURE

```
CREATE PROC spInsertCustomer
(
    @Id NVARCHAR(50),
    @Password NVARCHAR(50),
    @FullName NVARCHAR(50),
    @Email NVARCHAR(50),
    @Photo NVARCHAR(50) = 'User.png',
    @Activated BIT = 0
)
AS
BEGIN
    INSERT INTO Customers(Id, Password,
        Fullname, Email, Photo, Activated)
    VALUES(@Id, @Password, @Fullname,
        @Email, @Photo, @Activated)
END
```

```
EXEC spInsertCustomer 'teonv', '123',
        N'Nguyễn Văn Tèo', 'teonv@gmail.com'
```

STORED PROCEDURE

```
CREATE PROC spInsertCategory
(
    @Name NVARCHAR(50),
    @NameVN NVARCHAR(50),
    @Id INT OUTPUT --lấy mã tự tăng
)
AS
BEGIN
    INSERT INTO Categories(Name, NameVN)
        VALUES(@Name, @NameVN)
    SELECT @Id=@@IDENTITY --gán mã tự tăng
END
```

```
DECLARE @Id INT
EXEC spInsertCategory 'Computer', 'Máy tính', @Id OUTPUT
PRINT @Id
```

Stored Procedure

```
CREATE PROC spSearchByPrice
(
    @MinPrice FLOAT,
    @MaxPrice FLOAT
)
AS
BEGIN
    SELECT * FROM Products
    WHERE UnitPrice BETWEEN @MinPrice AND @MaxPrice
END
```

```
EXEC spSearchByPrice 10, 80
```

CALLABLE STATEMENT

```
String sql = "{call spInsertCategory (?, ?)}";
CallableStatement cstmt = con.prepareCall (sql);
cstmt.registerOutParameter(2, Types.INTEGER);
cstmt.setString(1, "Nokia");
cstmt.executeUpdate();
int MaLoai = cstmt.getInt(2);
```

```
CREATE PROCEDURE spInsertCategory
(
    @Name NVARCHAR(50),
    @Id INT OUT
)
AS
BEGIN
    INSERT INTO Categories(Name) VALUES(@Name)
    SET @Id = @@IDENTITY --số tự tăng vừa sinh ra
END
```

TRANSACTION

```
try {
        // hủy chế độ auto-commit
        conn.setAutoCommit(false);
        <thực thi khối lệnh thao tác dữ liệu>
        conn.commit();
}
catch(SQLException ex) {
        // hủy bỏ các câu lệnh đã thực hiện
        conn.rollback();
}
//  đặt lại chế độ auto-commit
con.setAutoCommit(true);
```

TRANSACTION

```
try
{
    // hủy chế độ auto-commit
    con.setAutoCommit(false);
    String sql1 = "Delete Student where Rollno = 3";
    PreparedStatement delStud = con.prepareStatement(sql1);
    // thực thi câu lệnh 1
    delStud.executeUpdate();
    String sql2 = "Delete Results where Rollno = 3";
    PreparedStatement delResult = con.prepareStatement(sql2);
    // thực thi câu lệnh 2
    delResult.executeUpdate();
    // thực hiện chuyển giao
    con.commit();
}
catch(SQLException ex)
{
    // hủy bỏ các câu lệnh đã thực hiện
    con.rollback();
}
//  đặt lại chế độ auto-commit
con.setAutoCommit(true);
```

TÓM TẮT

- Câu lệnh SELECT
- Kết nối nhiều bảng
- Tổng hợp - Thống kê
- PreparedStatement
- CallableStatement