10 BAB X PENGAKSESAN DATABASE

10.1 IDENTITAS

Kajian

Relasi Antar Kelas 2, Exception Handling, Akses Database

Topik

Pengaksesan Basis Data MySQL

Kompetensi Utama

1. Mahasiswa memahami konsep pengaksesan basis data MySQL menggunakan bahasa java
2. Mahasiswa mampu menerapkan query SELECT menggunakan bahasa pemrograman java.
3. Mahasiswa mampu menerapkan query INSERT menggunakan bahasa pemrograman java.
4. Mahasiswa mampu menerapkan query DELETE menggunakan bahasa pemrograman java.
5. Mahasiswa mampu menerapkan query UPDATE menggunakan bahasa pemrograman java.

Lama Kegiatan Praktikum

1. Pertemuan Terbimbing : 1 x 120 menit
2. Kegiatan Mandiri : 0 x 120 menit

Parameter Penilaian

1. Tugas Awal/Pendahuluan
2. Jurnal Pengamatan

10.2 PERTANYAAN PENDAHULUAN

a. Apakah yang dimaksud dengan JDBC?
b. Langkah apa yang harus dilakukan untuk membuka koneksi antara Java-MySQL?
c. Apa yang dimaksud dengan ResultSet & interface PreparedStatement?

10.3 PRAKTIK

10.3.1 Exercise : Mengakses database MySQL

Latihan ini meliputi kegiatan untuk membuat class Koneksi berdasarkan tabel yang sudah ditentukan sebelumnya.

JDBC dibutuhkan untuk menghubungkan bahasa pemrograman java dengan database. JDBC merupakan singkatan dari Java DataBase Connectivity. JDBC merupakan driver untuk mengakses database. Analoginya seperti driver printer untuk menggunakan sebuah printer melalui computer. Driver JDBC sendiri merupakan koleksi class-class Java yang dikumpulkan dalam satu atau beberapa file .jar. JDBC yang digunakan berbeda-beda utuk setiap database yang digunakan.

Ada beberapa langkah yang harus dilakukan untuk mengakses database menggunakan bahasa java:

1. Import packages terkait. Untuk menggunakan JDBC package terkait harus diimport. Package yang biasanya diimport yaitu import java.sql.*
2. Mendaftarkan driver JDBC tersebut. Biasanya merupakan 1 kelas tersendiri sebagai inisialisasi driver yang digunakan sehingga bisa membuka kanal pengaksesan database.
3. Membuka koneksi. Caranya dengan menggunakan method DriverManager.getConnection(), hal ini merepresentasikan koneksi fisik dengan database.
4. Eksekusi query. Hal ini membutuhkan sebuah objek dengan tipe Statement untuk membangun dan men-submit SQL statement ke database.
5. Ekstrak data hasil eksekusi query. Tipe data hasil eksekusi query biasanya adalah ResultSet yang memiliki nilai record data dari database. Untuk meng-ekstrak ResultSet ke dalam bentuk Object (String dll), dibutuhkan method ResultSet.getXXX().
6. Clean up. Setelah mengakses data, sebaiknya dilakukan closing database.

Terkait langkah ke-4 (point d), objek bertipe Statement dibentuk dari 3 interface yang mengandung method untuk mengeksekusi query yang diberikan:

1. Statement: merupakan general-purpose pengaksesan database. Digunakan untuk mengeksekusi query yang static seperti “select * from nama_tabel”. Interface ini tidak menerima parameter.
2. PreparedStatement: Digunakan untuk mengeksekusi query dinamis dan memiliki input parameter. Jika query static harus dieksekusi berulang kali, penggunaan preparedStatement akan lebih efektif dibandingkan Statement.
3. CallableStatement: berfungsi untuk mengakses stored procedure dari database (procedure, function dll)

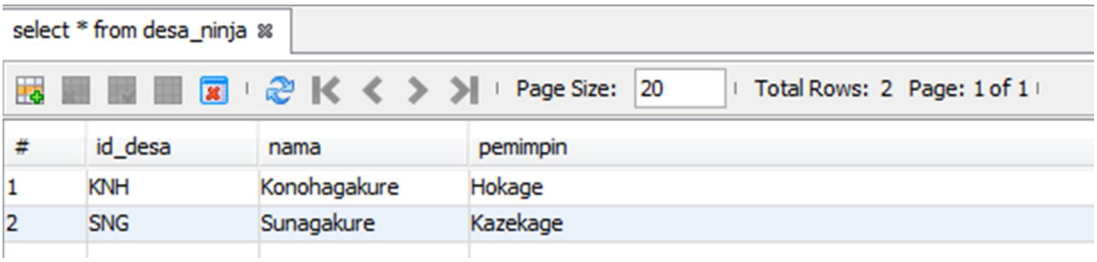
Untuk mengeksekusi query, terdapat 3 method yang digunakan (terdapat pada objek bertipe Statement dan PreparedStatement):

1. boolean execute(String SQL)
2. int executeUpdate(String SQL)
3. ResultSet executeQuery(String SQL)

10.3.1.1 Soal, Langkah Penyelesaian dan Solusi Lengkap

Permasalahan/Soal

Diketahui tabel sebagai berikut pada MySQL:



#	id_desa	nama	pemimpin
1	KNH	Konohagakure	Hokage
2	SNG	Sunagakure	Kazekage

Nama database: “si_desa”
Url database: “jdbc:mysql://localhost/si_desa”
Username: “root”
Password: “” <tidak ada password>
Driver JDBC: "com.mysql.jdbc.Driver"

Nama Table: desa_ninja
Atribut: id_desa (primary key), nama, pemimpin

Buatlah pengaksesan terhadap tabel di atas.

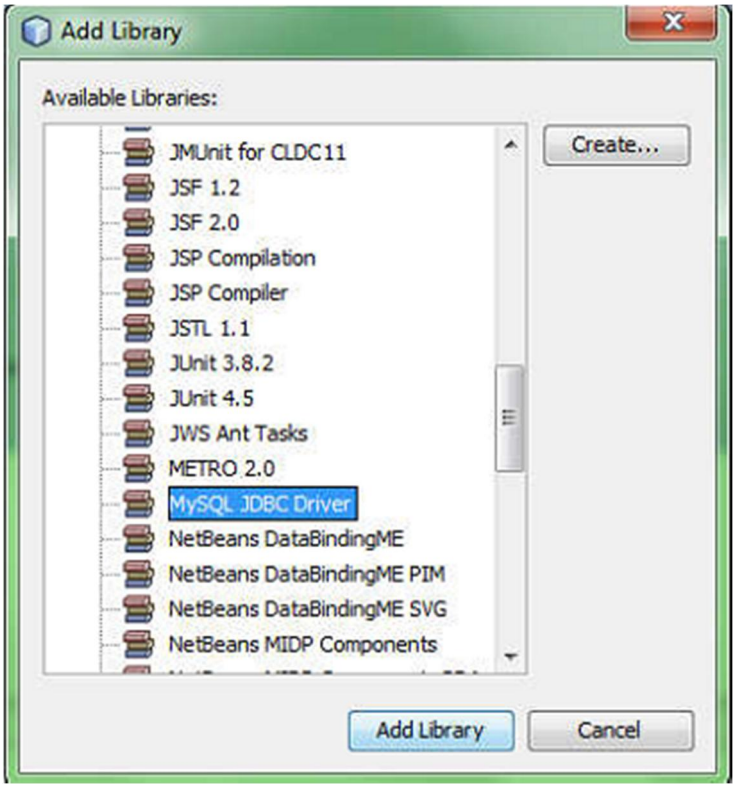
Langkah Penyelesaian

Untuk mengakses database, dibutuhkan 1 kelas untuk mewakilkan langkah a-c. Hal ini tidak harus dilakukan, tetapi agar memudahkan maintainance data, jika terdapat perubahan database yang digunakan atau perubahan informasi lain seperti username dan password, dibuat 1 kelas untuk merepresentasikan hal ini. Jadi kelas lain yang mengakses database cukup membuat objek dari kelas “Pool Connection” ini.

Informasi yang dibutuhkan untuk membuat kelas ini adalah nama database, username, password dan driver JDBC yang digunakan.

```
//Langkah pertama, import package terkait
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class KoneksiDB {
    // driver JDBC driver dan database URL

    private final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private final String DB_URL = "jdbc:mysql://localhost/si_desa";
    //  Database credentials
    private final String USER = "root";
    private final String PASS = "";
    private Connection conn = null;

    public void bukaKoneksi() {
        boolean flag = false;
        try {
            //Langkah ke-2: Registrasi JDBC
            Class.forName(JDBC_DRIVER);
        } catch (Exception e) {
            System.out.println(e.getMessage());
            flag = true;
        }
        if (!flag) {
            try {
                //Langkah ke-3: buka koneksi
                conn = DriverManager.getConnection(DB_URL, USER, PASS);
            } catch (SQLException e) {
                System.out.println(e.getMessage());
            }
        }
    }

    public Connection getConn() {
        return conn;
    }
}
```

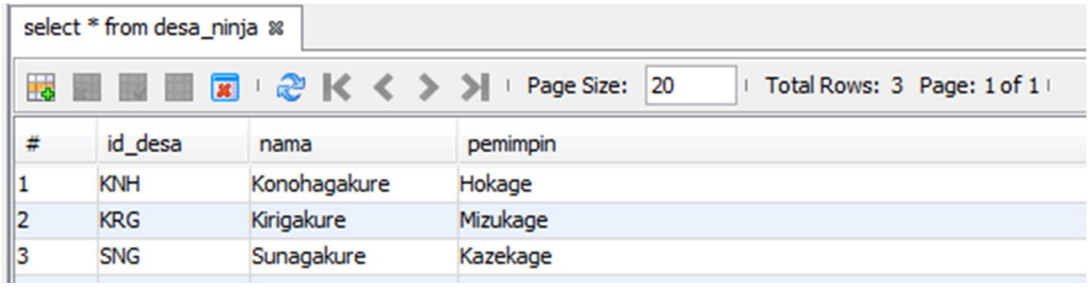
JDBC merupakan package class yang bukan bawaan dari bahasa Java. Sehingga library tersebut harus di-import terlebih dahulu ke project yang terkait. Caranya, klik kanan di bagian “Libraries” pada project yang mengakses database, pilih “Add Library”. Di sini, bisa ditambahkan Library MySQL ataupun Oracle. Untuk MySQL dapat ditambahkan “MySQL JDBC Driver” (jika menggunakan Netbeans).


Add Library
Available Libraries:
JMUnit for CLDC11
JSF 1.2
JSF 2.0
JSP Compilation
JSP Compiler
JSTL 1.1
JUnit 3.8.2
JUnit 4.5
JWS Ant Tasks
METRO 2.0
MySQL JDBC Driver
NetBeans DataBindingME
NetBeans DataBindingME PIM
NetBeans DataBindingME SVG
NetBeans MIDP Components
Create...
Add Library
Cancel

Cara mengakses database dengan menggunakan query DML (INSERT, UPDATE, dan DELETE) serta SELECT adalah sebagai berikut:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

/**
 *
 * @author Eja
 */
public class Main {

    public static void main(String[] args) {
        //bentuk objek dari class KoneksiDB
        KoneksiDB kdb = new KoneksiDB();
        kdb.bukaKoneksi();

        //ambil atribut Connection dari KoneksiDB
        Connection kon = kdb.getConn();

        //query INSERT
        String kueriInsert = "INSERT INTO desa_ninja"
                + "(id_desa, nama, pemimpin) VALUES"
                + "(?,?,?)";
        int rowAffect = 0;

        //persiapan kueri dari interface PreparedStatement
        PreparedStatement ps;
        try {
            ps = kon.prepareStatement(kueriInsert);
            ps.setString(1, "KRG");
            ps.setString(2, "Kirigakure");
            ps.setString(3, "Mizukage");
            rowAffect = ps.executeUpdate();
        } catch (SQLException ex) {
            System.out.println("Error: "+ex.getMessage());
        }
        if(rowAffect > 0){
            System.out.println("Kueri Berhasil Dieksekusi");
        }else{
            System.out.println("Kueri Gagal Dieksekusi");
        }
    }
}
```

Perhatikan bahwa dengan menggunakan PreparedStatement, terdapat persiapan query dengan karakter tanda tanya. Karakter ini akan diisi nilainya dengan setString (bisa juga setInt dan lainnya, tergantung tipe data). Setelah mengeksekusi class di atas, maka data pada database akan bertambah seperti yang ditampilkan pada gambar di bawah.


select * from desa_ninja

Page Size: 20 Total Rows: 3 Page: 1 of 1

#	id_desa	nama	pemimpin
1	KNH	Konohagakure	Hokage
2	KRG	Kirigakure	Mizukage
3	SNG	Sunagakure	Kazekage

Untuk Delete, dan Update lakukan hal yang sama dengan query berbeda. Untuk SELECT, ubah class di atas menjadi seperti yang ditampilkan berikut:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Main{
    public static void main(String[] args) {
        KoneksiDB kdb = new KoneksiDB();
        kdb.bukaKoneksi();

        //ambil atribut Connection dari KoneksiDB
        Connection kon = kdb.getConn();

         //query SELECT
        String kueriSelect = "SELECT * FROM desa_ninja";
        PreparedStatement ps;
        ResultSet rs;
        try {
            ps = kon.prepareStatement(kueriSelect);
            rs = ps.executeQuery();
            //menampilkan nilai dari ResultSet
            while(rs.next()){
                System.out.println("Id Desa: "+rs.getString(1));
                System.out.println("Nama Desa: "+rs.getString(2));
                System.out.println("Pemimpin: "+rs.getString(3));
            }
        } catch (SQLException e) {
            System.out.println("Error: "+e);
        }
    }

}
```

Perhatikan bahwa hasil dari pemanggilan executeQuery adalah ResultSet, dan ResultSet harus di-extract agar dapat ditampilkan. Extraksi ResultSet dapat menggunakan perulangan ataupun kondisional.