Java - Conexoes e CRUD em Banco de Dados

Este artigo descreve como realizar conexões e operações CRUD (Create, Read, Update, Delete) em diferentes tipos de bancos de dados:

- Relacional,
- Não Relacional,
- em Memória ,
- Nuvem

utilizando Java nas versões 8, 11 e 17 no NetBeans.

Cada banco de dados será configurado com uma classe separada para a conexão, e as operações CRUD serão feitas de forma modular para cada tipo de banco.

Estrutura da Tabela e Configuração do Banco de Dados

A tabela usuarios, que será utilizada em todos os exemplos, é criada utilizando o seguinte script SQL

```
CREATE DATABASE IF NOT EXISTS meu_banco;

USE meu_banco;

CREATE TABLE IF NOT EXISTS usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    idade INT
);
```

Para projetos Maven, adicione as dependências necessárias no arquivo pom.xm, caso esteja utilizando um projeto Java simples, basta adicionar os .jar necessários às bibliotecas do projeto.

Banco de Dados Relacional: MySQL

Dependência Maven (mysql-connector)

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.30</version>
</dependency>
```

Classe de Conexão com MySQL

Classe MySQLConnection.java

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

public class MySQLConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/meu_banco";
    private static final String USER = "usuario";
    private static final String PASSWORD = "senha";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

Classe CRUD para MySQL

Classe MySQLCRUD.java

```
import java.sql.*;

public class MySQLCRUD {

    public void createUser(String nome, String email, int idade) {
        String sql = "INSERT INTO usuarios (nome, email, idade) VALUES (?, ?, ?)";
        try (Connection conn = MySQLConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, nome);
            pstmt.setString(2, email);
            pstmt.setInt(3, idade);
            pstmt.executeUpdate();
            System.out.println("Usuário inserido com sucesso no MySQL.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void readUsers() {
        String sql = "SELECT * FROM usuarios";
```

```
        try (Connection conn = MySQLConnection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                System.out.printf("ID: %d, Nome: %s, Email: %s, Idade: %d%n",
                        rs.getInt("id"), rs.getString("nome"),
                        rs.getString("email"), rs.getInt("idade"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void updateUser(int id, int novaIdade) {
        String sql = "UPDATE usuarios SET idade = ? WHERE id = ?";
        try (Connection conn = MySQLConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, novaIdade);
            pstmt.setInt(2, id);
            pstmt.executeUpdate();
            System.out.println("Usuário atualizado com sucesso no MySQL.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void deleteUser(int id) {
        String sql = "DELETE FROM usuarios WHERE id = ?";
        try (Connection conn = MySQLConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, id);
            pstmt.executeUpdate();
            System.out.println("Usuário deletado com sucesso no MySQL.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Banco de Dados Não Relacional: MongoDB

Dependência Maven (mongodb-driver-sync)

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>4.5.0</version>
</dependency>
```

Classe de Conexão com MongoDB

Classe MongoDBConnection.java

```
import com.mongodb.client.MongoClient;
```

```
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoDatabase;

public class MongoDBConnection {
    private static final String URI = "mongodb://localhost:27017";
    private static final String DATABASE_NAME = "meu_banco";

    public static MongoDatabase getConnection() {
        MongoClient mongoClient = MongoClients.create(URI);
        return mongoClient.getDatabase(DATABASE_NAME);
    }
}
```

Classe CRUD para MongoDB

Classe **MongoDBCRUD.java**

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;

public class MongoDBCRUD {
    private static final String COLLECTION_NAME = "usuarios";

    public void createUser(String nome, String email, int idade) {
        MongoDatabase database = MongoDBConnection.getConnection();
        MongoCollection<Document> collection = database.getCollection(COLLECTION_NAME);

        Document user = new Document("nome", nome)
                .append("email", email)
                .append("idade", idade);
        collection.insertOne(user);
        System.out.println("Usuário inserido com sucesso no MongoDB.");
    }

    public void readUsers() {
        MongoDatabase database = MongoDBConnection.getConnection();
        MongoCollection<Document> collection = database.getCollection(COLLECTION_NAME);

        for (Document doc : collection.find()) {
            System.out.println(doc.toJson());
        }
    }

    public void updateUser(String nome, int novaIdade) {
        MongoDatabase database = MongoDBConnection.getConnection();
        MongoCollection<Document> collection = database.getCollection(COLLECTION_NAME);

        Document query = new Document("nome", nome);
        Document update = new Document("$set", new Document("idade", novaIdade));
        collection.updateOne(query, update);
        System.out.println("Usuário atualizado com sucesso no MongoDB.");
    }

    public void deleteUser(String nome) {
        MongoDatabase database = MongoDBConnection.getConnection();
        MongoCollection<Document> collection = database.getCollection(COLLECTION_NAME);
```

```
        collection.deleteOne(new Document("nome", nome));
        System.out.println("Usuário deletado com sucesso no MongoDB.");
    }
}
```

Banco de Dados em Memória: H2

Dependência Maven (h2)

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>2.1.210</version>
</dependency>
```

Classe de Conexão com H2

Classe H2Connection.java

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

public class H2Connection {
    private static final String URL = "jdbc:h2:mem:meu_banco";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL);
    }
}
```

Classe CRUD para H2

Classe H2CRUD.java

```
import java.sql.*;

public class H2CRUD {

    public void createUser(String nome, String email, int idade) {
        String sql = "INSERT INTO usuarios (nome, email, idade) VALUES (?, ?, ?)";
        try (Connection conn = H2Connection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, nome);
            pstmt.setString(2, email);
            pstmt.setInt(3, idade);
            pstmt.executeUpdate();
            System.out.println("Usuário inserido com sucesso no H2.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
    public void readUsers() {
        String sql = "SELECT * FROM usuarios";
        try (Connection conn = H2Connection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                System.out.printf("ID: %d, Nome: %s, Email: %s, Idade: %d%n",
                                  rs.getInt("id"), rs.getString("nome"),
                                  rs.getString("email"), rs.getInt("idade"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void updateUser(int id, int novaIdade) {
        String sql = "UPDATE usuarios SET idade = ? WHERE id = ?";
        try (Connection conn = H2Connection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, novaIdade);
            pstmt.setInt(2, id);
            pstmt.executeUpdate();
            System.out.println("Usuário atualizado com sucesso no H2.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void deleteUser(int id) {
        String sql = "DELETE FROM usuarios WHERE id = ?";
        try (Connection conn = H2Connection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, id);
            pstmt.executeUpdate();
            System.out.println("Usuário deletado com sucesso no H2.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Banco de Dados em Nuvem: PostgreSQL no Heroku

Dependência Maven (postgresql)

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.3.8</version>
</dependency>
```

Classe de Conexão com PostgreSQL no Heroku

Classe PostgresConnection.java

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

public class PostgresConnection {
    private static final String URL = "jdbc:postgresql://<heroku_host>:<port>/<database>";
    private static final String USER = "<usuario>";
    private static final String PASSWORD = "<senha>";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }
}
```

Classe CRUD para PostgreSQL no Heroku

Classe PostgresCRUD.java

```
import java.sql.*;

public class PostgresCRUD {

    public void createUser(String nome, String email, int idade) {
        String sql = "INSERT INTO usuarios (nome, email, idade) VALUES (?, ?, ?)";
        try (Connection conn = PostgresConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, nome);
            pstmt.setString(2, email);
            pstmt.setInt(3, idade);
            pstmt.executeUpdate();
            System.out.println("Usuário inserido com sucesso no PostgreSQL.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void readUsers() {
        String sql = "SELECT * FROM usuarios";
        try (Connection conn = PostgresConnection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                System.out.printf("ID: %d, Nome: %s, Email: %s, Idade: %d%n",
                        rs.getInt("id"), rs.getString("nome"),
                        rs.getString("email"), rs.getInt("idade"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void updateUser(int id, int novaIdade) {
```

```
        String sql = "UPDATE usuarios SET idade = ? WHERE id = ?";
        try (Connection conn = PostgresConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, novaIdade);
            pstmt.setInt(2, id);
            pstmt.executeUpdate();
            System.out.println("Usuário atualizado com sucesso no PostgreSQL.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void deleteUser(int id) {
        String sql = "DELETE FROM usuarios WHERE id = ?";
        try (Connection conn = PostgresConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setInt(1, id);
            pstmt.executeUpdate();
            System.out.println("Usuário deletado com sucesso no PostgreSQL.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Neste artigo, exploramos como criar conexões com diversos tipos de banco de dados, incluindo MySQL, MongoDB, H2 e PostgreSQL em nuvem, utilizando Java 8, 11 e 17 no NetBeans.

Além disso, realizamos operações CRUD (Create, Read, Update, Delete) para cada um desses bancos, mostrando a versatilidade e as melhores práticas para integração de sistemas com diferentes tecnologias.

EducaCiência FastCode para a comunidade