Persistência de Dados

Java JDBC
Java DB (Derby)
Java DAO

Rafael Vargas Mesquita

ROTEIRO

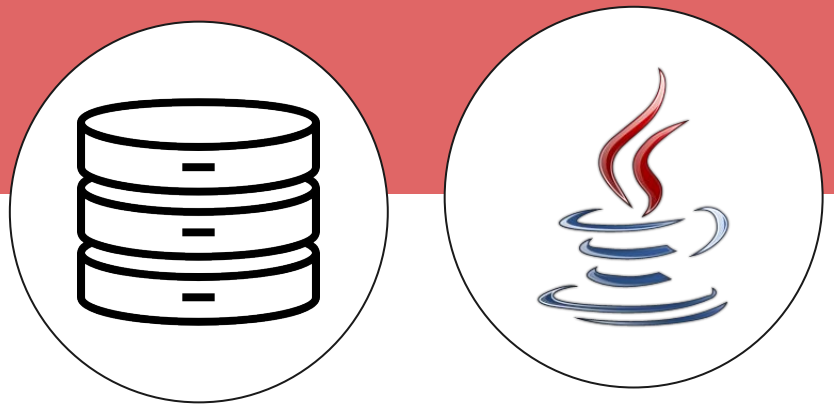
Java JDBC

Java DB (Derby)

Java JDBC (Códigos)

Java DAO

Persistência de Dados



Java JDBC

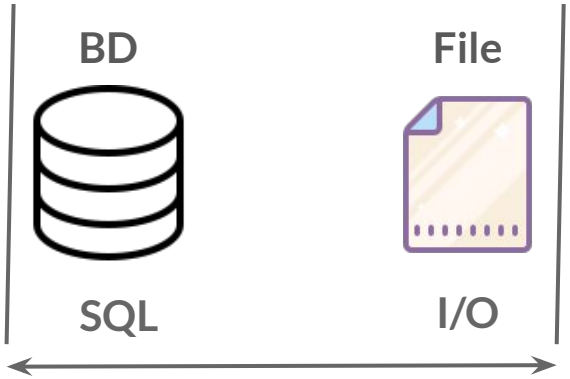
Rafael Mesquita, Prof.

Introdução

Muitos sistemas precisam **persistir dados** em algum ponto de funcionamento

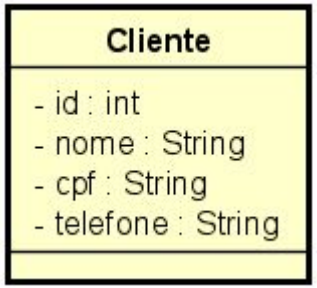
Dados de um sistema podem ser armazenados de diferentes formas

Persistência de dados



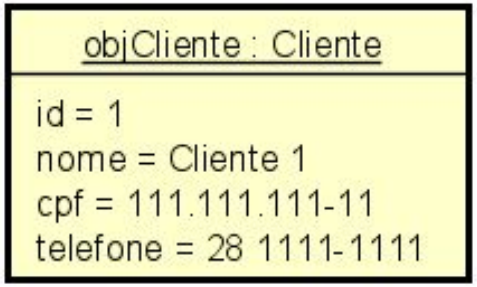
Introdução: Persistência de Dados (Banco de Dados)

Classe Cliente



Sem persistência

Objeto Cliente - Memória RAM



Com persistência

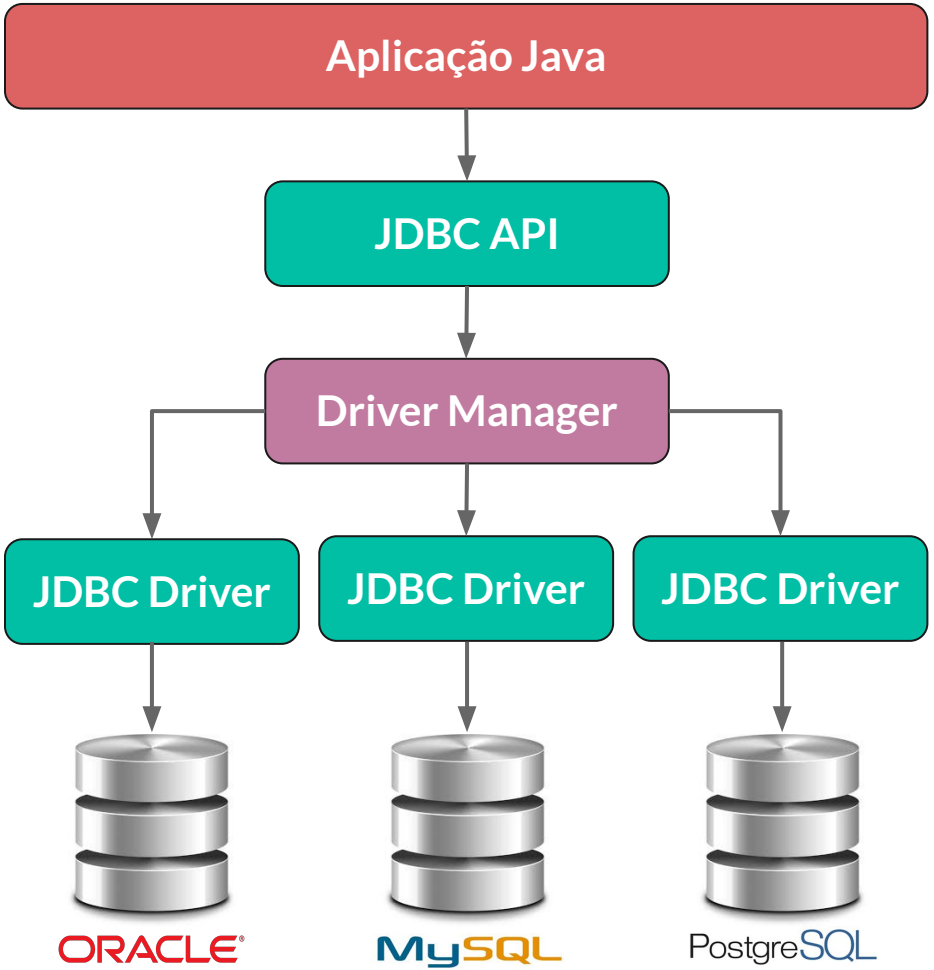
id	nome	cpf	telefone
1	Cliente 1	111.111.111-11	(28) 99999-8888

Tabela Clientes - Banco de Dados

JDBC

O que é?

- Uma biblioteca;
- Implementada em Java;
- Disponibiliza classes e interfaces para acesso a banco de dados;



JDBC

Principais classes e interfaces do pacote java.sql

- **DriverManager**, cria conexão com o banco de dados;
- **Connection**, mantém uma conexão aberta com o banco;
- **Statement**, gerencia e executa instruções SQL;
- **PreparedStatement**, gerencia e executa instruções SQL (parâmetros);
- **ResultSet**, recebe os dados obtidos em uma pesquisa ao banco.

Persistência de Dados

Java DB (Derby)

Rafael Mesquita, Prof.

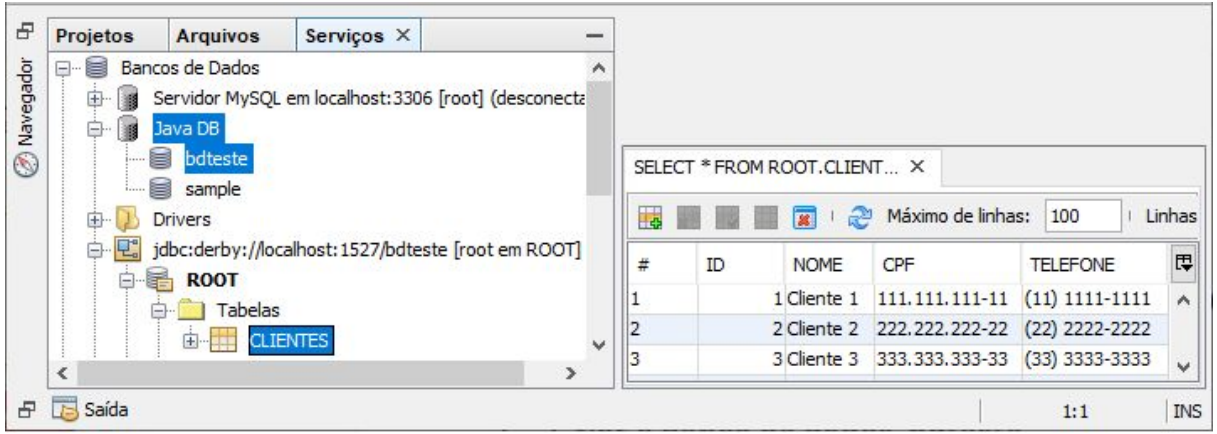
Banco de Dados Java DB (Derby)

O Java DB é uma distribuição da Sun com suporte do Apache Derby.

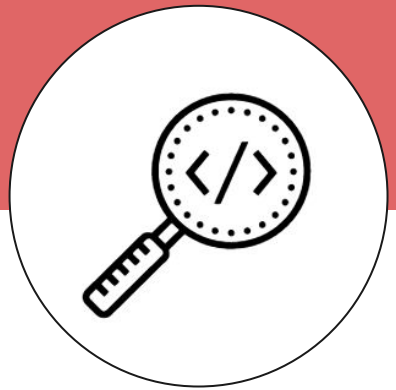
O Java DB é um servidor de banco de dados com base em padrões, seguro e totalmente transacional, escrito inteiramente em Java, e dá suporte total a SQL, JDBC API e à tecnologia Java EE.

Banco de Dados Java DB (Derby)

1. Instalar Java DB (Derby)
2. Criar o banco de dados ‘bdteste’
3. Executar o script para criação da tabela ‘clientes’



Persistência de Dados



Java JDBC (Códigos)

Rafael Mesquita, Prof.

JDBC: Códigos

Exemplo de códigos utilizando JDBC

- **Cliente**, exemplo de código com os dados de clientes;
- **MainSelect**, exemplo de código utilizando a cláusula SQL SELECT;
- **MainInsert**, exemplo de código utilizando a cláusula SQL INSERT;
- **MainUpdate**, exemplo de código utilizando a cláusula SQL UPDATE;
- **MainDelete**, exemplo de código utilizando a cláusula SQL DELETE;

https://github.com/ravarmes/jdbc-exemplos-java

JDBC: Classe Cliente

```java
public class Cliente {
    private Integer id;
    private String nome;
    private String cpf;
    private String telefone;

    public Cliente() {
    }

    public Cliente(String nome, String cpf, String telefone) {
        this.nome = nome;
        this.cpf = cpf;
        this.telefone = telefone;
    }

    public Cliente(Integer id, String nome, String cpf, String telefone) {
        this.id = id;
        this.nome = nome;
        this.cpf = cpf;
        this.telefone = telefone;
    }

    //Getters e Setters omitidos...

    @Override
    public String toString() {
        return "id=" + id + ", nome=" + nome + ", cpf=" + cpf + ", telefone=" + telefone;
    }

}
```

JDBC: Cláusula SELECT

```
public class MainSelect {
    public static void main(String[] args) throws SQLException, ClassNotFoundException {
        Class.forName("org.apache.derby.jdbc.ClientDriver");

        String DATABASE_URL = "jdbc:derby://localhost:1527/bdteste";
        String usuario = "root";
        String senha = "123";

        Co                                                              io, senha);

        St
        Pr
        Re

        wh
            Cliente cliente = new Cliente();
            cliente.setId(rs.getInt("id"));
            cliente.setNome(rs.getString("nome"));
            cliente.setCpf(rs.getString("cpf"));
            cliente.setTelefone(rs.getString("telefone"));
            System.out.println(cliente);
        }
    }
}
```

```
Saída - jdbc-exemplos-java (run)
run:
id=1, nome=Cliente 1, cpf=111.111.111-11, telefone=(11) 1111-1111
id=2, nome=Cliente 2, cpf=222.222.222-22, telefone=(22) 2222-2222
id=3, nome=Cliente 3, cpf=333.333.333-33, telefone=(33) 3333-3333
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

JDBC: Cláusula INSERT

```
public class MainInsert {
    public static void main(String[] args) throws SQLException, ClassNotFoundException {
        Class.forName("org.apache.derby.jdbc.ClientDriver");

        String DATABASE_URL = "jdbc:derby://localhost:1527/bdteste";
        String usuario = "root";
        String
        Connect                                                    suario, senha);
        String                                                      ?, ?)";
        Prepare

        Cliente cliente = new Cliente("Cliente 4", "444.444.444-44", "(44) 4444-4444");

        ps.setString(1, cliente.getNome());
        ps.setString(2, cliente.getCpf());
        ps.setString(3, cliente.getTelefone());
        ps.execute();
    }
}
```

Saída - jdbc-exemplos-java (run)

```
run:
id=1, nome=Cliente 1, cpf=111.111.111-11, telefone=(11) 1111-1111
id=2, nome=Cliente 2, cpf=222.222.222-22, telefone=(22) 2222-2222
id=3, nome=Cliente 3, cpf=333.333.333-33, telefone=(33) 3333-3333
id=4, nome=Cliente 4, cpf=444.444.444-44, telefone=(44) 4444-4444
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

JDBC: Cláusula UPDATE

```
public class MainUpdate {
    public static void main(String[] args) throws SQLException, ClassNotFoundException {
        Class.forName("org.apache.derby.jdbc.ClientDriver");

        String DATABASE_URL = "jdbc:derby://localhost:1527/bdteste";
        String usuario = "root";
        String
        Connec                                                          uario, senha);
        String                                                          =?";
        Prepar
        Client                                                          0000-0000");

        ps.setString(1, cliente.getNome());
        ps.setString(2, cliente.getCpf());
        ps.setString(3, cliente.getTelefone());
        ps.setInt(4, cliente.getId());
        ps.execute();
    }
}
```

Saída - jdbc-exemplos-java (run)

```
run:
id=1, nome=Cliente 0, cpf=000.000.000-00, telefone=(00) 0000-0000
id=2, nome=Cliente 2, cpf=222.222.222-22, telefone=(22) 2222-2222
id=3, nome=Cliente 3, cpf=333.333.333-33, telefone=(33) 3333-3333
id=4, nome=Cliente 4, cpf=444.444.444-44, telefone=(44) 4444-4444
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

JDBC: Cláusula DELETE

```
public class MainDelete {
    public static void main(String[] args) throws SQLException, ClassNotFoundException {
        Class.forName("org.apache.derby.jdbc.ClientDriver");

        String
        String
        String

        Connect                                                    usuario, senha);

        String
        PreparedStatement ps = connection.prepareStatement(sql);

        ps.execute();
    }
}
```

Saída - jdbc-exemplos-java (run)

```
run:
id=2, nome=Cliente 2, cpf=222.222.222-22, telefone=(22) 2222-2222
id=3, nome=Cliente 3, cpf=333.333.333-33, telefone=(33) 3333-3333
id=4, nome=Cliente 4, cpf=444.444.444-44, telefone=(44) 4444-4444
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Persistência de Dados

Java DAO

Rafael Mesquita, Prof.

Data Access Object (DAO) - Contexto

O acesso ao banco de dados depende fortemente do tipo de armazenamento e da tecnologia utilizada;

A interação entre as regras de negócio e as regras de armazenamento de dados influencia na qualidade do sistema.

DAO

Data Access Object (DAO) - Problema

Visão

Controle

C1

C2

Modelo

M1

M2

???

Persistência

BD

File

SQL / I-O

DAO

Data Access Object (DAO) - Solução

Visão

Controle

C1

C2

Modelo

M1

M2

DAO

DAO1

DAO2

Persistência

BD

File

SQL / I-O

Data Access Object (DAO) - Solução

Modelo: Cliente.java

```java
public class Cliente {
    private String nome;
    private String cpf;
    private String telefone;

    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getCpf() {
        return cpf;
    }
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public String getTelefone() {
        return telefone;
    }
    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

}
```

DAO: ClienteDAO.java

```java
public class ClienteDAO {

    private Connection connection;

    public boolean inserir(Cliente cliente){
        String sql = "INSERT INTO CLIENTES (nome, cpf, telefone) VALUES (?, ?, ?)";
        ...
    }

    public boolean alterar(Cliente cliente){
        String sql = "UPDATE CLIENTES SET nome=?, cpf=?, telefone=? WHERE ID=?";
        ...
    }

    public boolean remover(Cliente cliente){
        String sql = "DELETE FROM CLIENTES WHERE ID=1";
        ...
    }

    public List<Cliente> listar(Cliente cliente){
        String sql = "SELECT * FROM CLIENTES";
        ...
    }

    public Cliente buscar(Cliente cliente){
        String sql = "SELECT * FROM CLIENTES WHERE ID=?";
        ...
    }
}
```

Data Access Object (DAO) - Solução

Criar um objeto para gerenciar a obtenção e o armazenamento de dados;

Abstrair e encapsular todo o acesso às fontes de dados;

JDBC: Códigos (Padrão DAO)

Exemplo de códigos utilizando JDBC e padrão DAO

- **ClienteDAO**, exemplo de código com métodos de acesso ao BD;
- **MainSelect**, exemplo de código utilizando a cláusula SQL SELECT;
- **MainInsert**, exemplo de código utilizando a cláusula SQL INSERT;
- **MainUpdate**, exemplo de código utilizando a cláusula SQL UPDATE;
- **MainDelete**, exemplo de código utilizando a cláusula SQL DELETE;

https://github.com/ravarmes/jdbc-dao-java

ClienteDAO

```
public class ClienteDAO {

    private Connection connection;

    public ClienteDAO() {...11 linhas }

    public List<Cliente> listar() {...19 linhas }

    public boolean inserir(Cliente cliente) {...14 linhas }

    public boolean alterar(Cliente cliente) {...15 linhas }

    public boolean remover(Integer id) {...12 linhas }

}
```

ClienteDAO: Construtor (Conexão com BD)

```
public class ClienteDAO {

    private Connection connection;

    public ClienteDAO() {
        try {
            Class.forName("org.apache.derby.jdbc.ClientDriver");
            String DATABASE_URL = "jdbc:derby://localhost:1527/bdteste";
            String usuario = "root";
            String senha = "123";
            this.connection = DriverManager.getConnection(DATABASE_URL, usuario, senha);
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(ClienteDAO.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public List<Cliente> listar() {...19 linhas }

    public boolean inserir(Cliente cliente) {...14 linhas }

    public boolean alterar(Cliente cliente) {...15 linhas }

    public boolean remover(Integer id) {...12 linhas }

}
```

DAO

ClienteDAO: listar (SELECT)

```
public class ClienteDAO {
    private Connection connection;
    public ClienteDAO() {...11 linhas }

    public List<Cliente> listar() {
        String sql = "SELECT * FROM clientes";
        List<Cliente> retorno = new ArrayList<>();
        try {
            PreparedStatement stmt = connection.prepareStatement(sql);
            ResultSet resultado = stmt.executeQuery();
            while (resultado.next()) {
                Cliente cliente = new Cliente();
                cliente.setId(resultado.getInt("id"));
                cliente.setNome(resultado.getString("nome"));
                cliente.setCpf(resultado.getString("cpf"));
                cliente.setTelefone(resultado.getString("telefone"));
                retorno.add(cliente);
            }
        } catch (SQLException ex) {
            Logger.getLogger(ClienteDAO.class.getName()).log(Level.SEVERE, null, ex);
        }
        return retorno;
    }

    public boolean inserir(Cliente cliente) {...14 linhas }

    public boolean alterar(Cliente cliente) {...15 linhas }

    public boolean remover(Integer id) {...12 linhas }
}
```

DAO

ClienteDAO: inserir (INSERT)

```
public class ClienteDAO {

    private Connection connection;

    public ClienteDAO() {...11 linhas }

    public List<Cliente> listar() {...19 linhas }

    public boolean inserir(Cliente cliente) {
        String sql = "INSERT INTO clientes(nome, cpf, telefone) VALUES(?,?,?)";
        try {
            PreparedStatement stmt = connection.prepareStatement(sql);
            stmt.setString(1, cliente.getNome());
            stmt.setString(2, cliente.getCpf());
            stmt.setString(3, cliente.getTelefone());
            stmt.execute();
            return true;
        } catch (SQLException ex) {
            Logger.getLogger(ClienteDAO.class.getName()).log(Level.SEVERE, null, ex);
            return false;
        }
    }

    public boolean alterar(Cliente cliente) {...15 linhas }

    public boolean remover(Integer id) {...12 linhas }

}
```

DAO

ClienteDAO: alterar(UPDATE)

```
public class ClienteDAO {

    private Connection connection;

    public ClienteDAO() {...11 linhas }

    public List<Cliente> listar() {...19 linhas }

    public boolean inserir(Cliente cliente) {...14 linhas }

    public boolean alterar(Cliente cliente) {
        String sql = "UPDATE clientes SET nome=?, cpf=?, telefone=? WHERE id=?";
        try {
            PreparedStatement stmt = connection.prepareStatement(sql);
            stmt.setString(1, cliente.getNome());
            stmt.setString(2, cliente.getCpf());
            stmt.setString(3, cliente.getTelefone());
            stmt.setInt(4, cliente.getId());
            stmt.execute();
            return true;
        } catch (SQLException ex) {
            Logger.getLogger(ClienteDAO.class.getName()).log(Level.SEVERE, null, ex);
            return false;
        }
    }

    public boolean remover(Integer id) {...12 linhas }

}
```

DAO

ClienteDAO: remover (DELETE)

```
public class ClienteDAO {

    private Connection connection;

    public ClienteDAO() {...11 linhas }

    public List<Cliente> listar() {...19 linhas }

    public boolean inserir(Cliente cliente) {...14 linhas }

    public boolean alterar(Cliente cliente) {...15 linhas }

    public boolean remover(Integer id) {
        String sql = "DELETE FROM clientes WHERE id=?";
        try {
            PreparedStatement stmt = connection.prepareStatement(sql);
            stmt.setInt(1, id);
            stmt.execute();
            return true;
        } catch (SQLException ex) {
            Logger.getLogger(ClienteDAO.class.getName()).log(Level.SEVERE, null, ex);
            return false;
        }
    }

}
```

Main's

```java
public class MainSelect {
    public static void main(String[] args) {
        ClienteDAO clienteDAO = new ClienteDAO();
        List<Cliente> lista = clienteDAO.listar();
        for (Cliente cliente : lista)
            System.out.println(cliente);
    }
}
```

```java
public class MainInsert {
    public static void main(String[] args) {
        Cliente cliente = new Cliente("Cliente 4", "444.444.444-44", "(44) 4444-4444");
        ClienteDAO clienteDAO = new ClienteDAO();
        clienteDAO.inserir(cliente);
    }
}
```

```java
public class MainUpdate {
    public static void main(String[] args) {
        Cliente cliente = new Cliente(1, "Cliente 0", "000.000.000-00", "(00) 0000-0000");
        ClienteDAO clienteDAO = new ClienteDAO();
        clienteDAO.alterar(cliente);
    }
}
```

```java
public class MainDelete {
    public static void main(String[] args) {
        ClienteDAO clienteDAO = new ClienteDAO();
        clienteDAO.remover(1);
    }
}
```

Referências Bibliográficas

H. M. Deitel, P. J. Deitel. Java: Como Programar, **Capítulo 24 – Acesso a banco de dados com JDBC**, 10ª Edição. Pearson, 2016.

Obrigado.

Sobre mim

Rafael Mesquita, Prof.

Prof. Dr. Formado em Ciência da Computação pela Universidade Federal de Lavras