

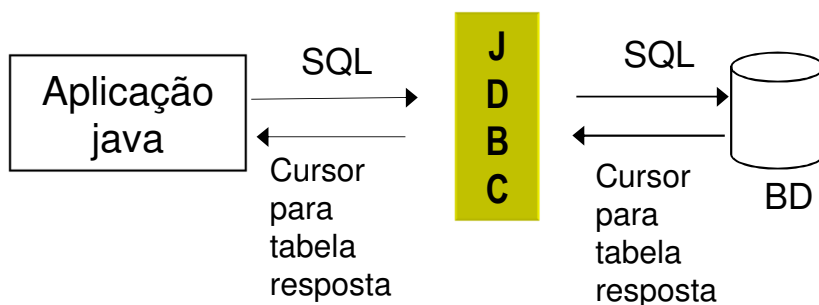
Persistência de Objetos

Fausto Maranhão Ayres

5 Persistência com SQL (JDBC)

JDBC (Java DataBase Connectivity)

JDBC é uma API para acesso a banco relacional via SQL



A idéia foi baseada no
ODBC do Windows

PASSO A PASSO

1. **Abrir/Fechar** a conexão com o BD
2. **Submeter** comando SQL dentro de uma transação
3. **Obter** os resultados (via cursor)

Abrir/Fechar Conexão

```
import java.sql.*;
```

```
try {
```

```
    Connection con = DriverManager.getConnection(  
        "jdbc:postgresql://localhost:5432/agenda",  
        "postgres",  
        "ifpb");
```

```
}
```

```
"jdbc:mysql://127.0.0.1:3306/bdaula"  
"jdbc:postgresql://localhost:5432/bdaula"  
"jdbc:oracle:thin:@maq.ifpb.edu.br:1521:bdaula"  
"jdbc:derby://127.0.0.1:1527/bdaula;create=true"
```

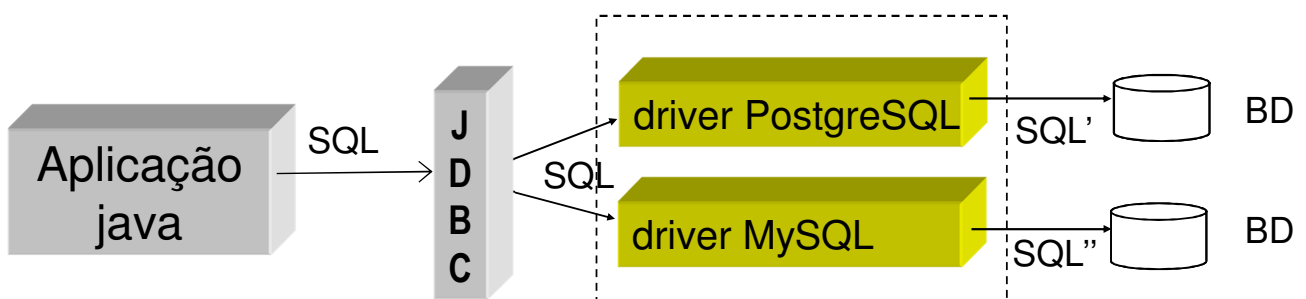
```
catch (SQLException e) {
```

```
    System.out.println("erro na conexao");
```

```
}
```

DRIVER

- Cada fabricante de SGBD fornece DRIVER
 - um arquivo .JAR, contendo .class para a comunicação com banco

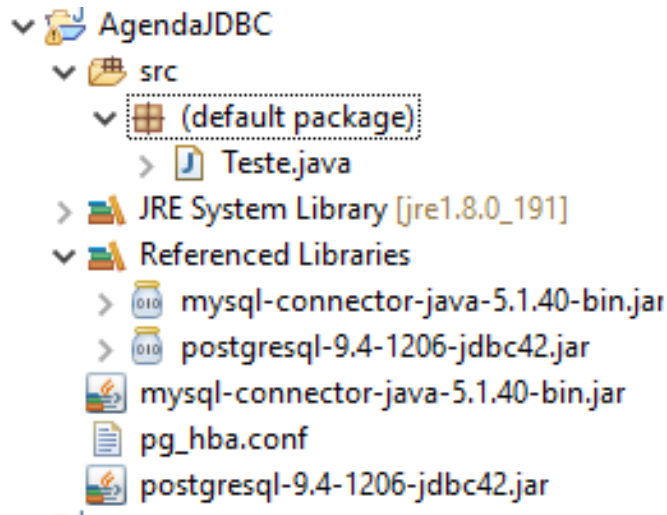


- Tem-se que baixar o driver no site do fabricante.

Download dos drivers

<https://jdbc.postgresql.org/download.html>

<https://dev.mysql.com/downloads/windows/installer/8.0.html>



Arquivos dos drivers
postgres e mysql

API - submeter comandos SQL

Classe Statement

- Envia comando SQL para o banco sem otimiza-lo

Classe PreparedStatement


- Envia comando SQL otimizado antes de sua execução, sendo mais eficiente nos casos onde o mesmo comando é utilizado várias vezes.

Exemplo: criar Tabelas

```
PreparedStatement st = con.prepareStatement(  
    " create table Pessoa(id SERIAL, nome varchar(30),  
    dtcadastro timestamp, primary key (id)) "  
);  
  
st.executeUpdate();
```

Exemplo: inserir dados

```
PreparedStatement st = con.prepareStatement(  
    "insert into Pessoa (nome, dtcadastro) values ('joao',  
    '2020-12-01 14:10:00') "  
);  
  
int i = st.executeUpdate();
```



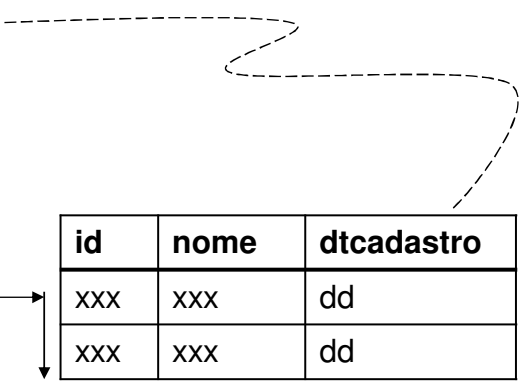
*Retorna a quantidade de
linhas inseridas/atualizadas*

Exemplo: consultar

- Classe **ResultSet** encapsula um **cursor** sobre a tabela resultado.
- O método **next()** avança o cursor para próxima linha

```
try{ ...  
    PreparedStatement st = con.prepareStatement(  
        "select * from pessoa");  
  
    ResultSet rs = st.executeQuery();  
    while( rs.next() ) {  
        System.out.println(  
            rs.getInt("id") +  
            rs.getString("nome") +  
            rs.getDate("dtcadastro"));  
    }  
    rs.close();  
    st.close();  
}  
catch(SQLException e){}
```

rs.next() →



id	nome	dtcadastro
xxx	xxx	dd
xxx	xxx	dd

ResultSet

Métodos para acessar campos da linha resultado

```
getInt (coluna)  
getDouble (coluna)  
getString (coluna)  
getBoolean (coluna)  
getDate (coluna)  
...
```

Ex:

```
rs.getInt("id")  
rs.getString("nome")
```

```
rs.getInt(1)  
rs.getString(2)
```

Uso de parâmetros dentro do SQL

- Pode ser usado em qualquer comando SQL

```
PreparedStatement st;  
st = con.prepareStatement(  
    "select * from pessoa where id = ? and nome = ?"  
);  
st.setInt (1, 5);  
st.setString(2, "joao");    // converte para 'joao'  
  
st = con.prepareStatement(  
    "select * from pessoa where id = :x and nome= :y"  
);  
st.setInt ("x", 5);  
st.setString("y", "joao"); // converte para 'joao'
```

Transação com vários comandos SQL

- Por default, cada comando SQL é comitado automaticamente
- É necessário commit manual para uma transação em grupo.

Ex: uma transferência entre duas contas

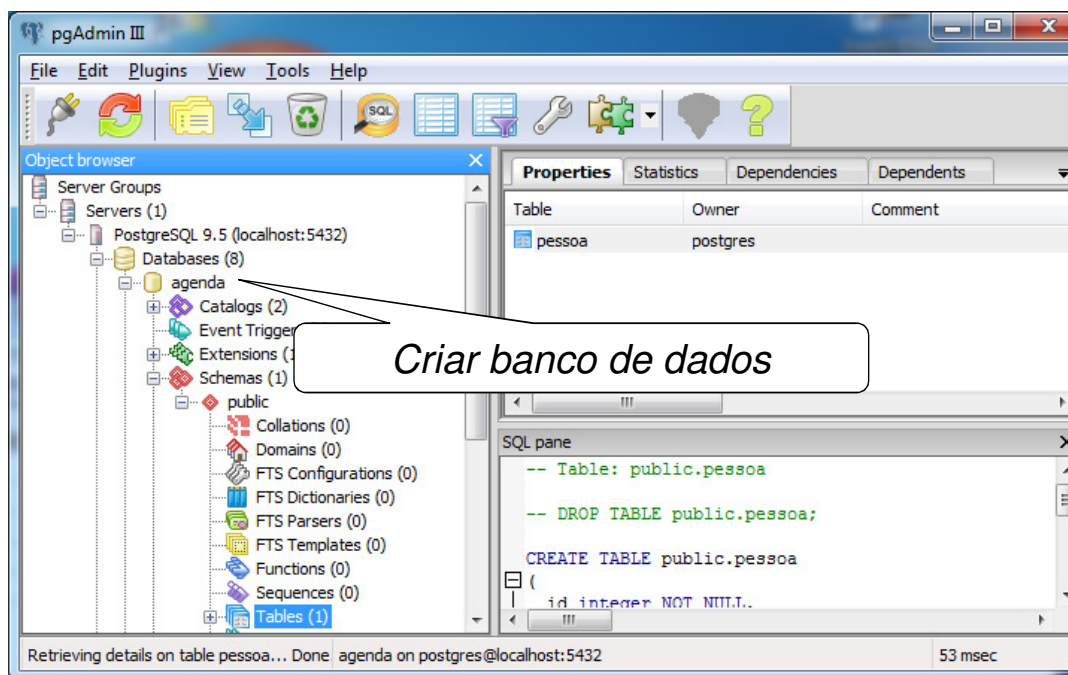
```
...  
con.setAutoCommit(false);  
PreparedStatement st1 = con.prepareStatement(  
    "update Conta set saldo=saldo-100 where id=1" );  
PreparedStatement st2 = con.prepareStatement(  
    "update Conta set saldo=saldo+100 where id=2" );  
a = st1.executeUpdate();  
b = st2.executeUpdate();  
if(a>0 && b>0)  
    con.commit( );    //efetiva os dois updates
```

BANCO

Fausto Maranhão Ayres - IFPB

13

PostgreSQL



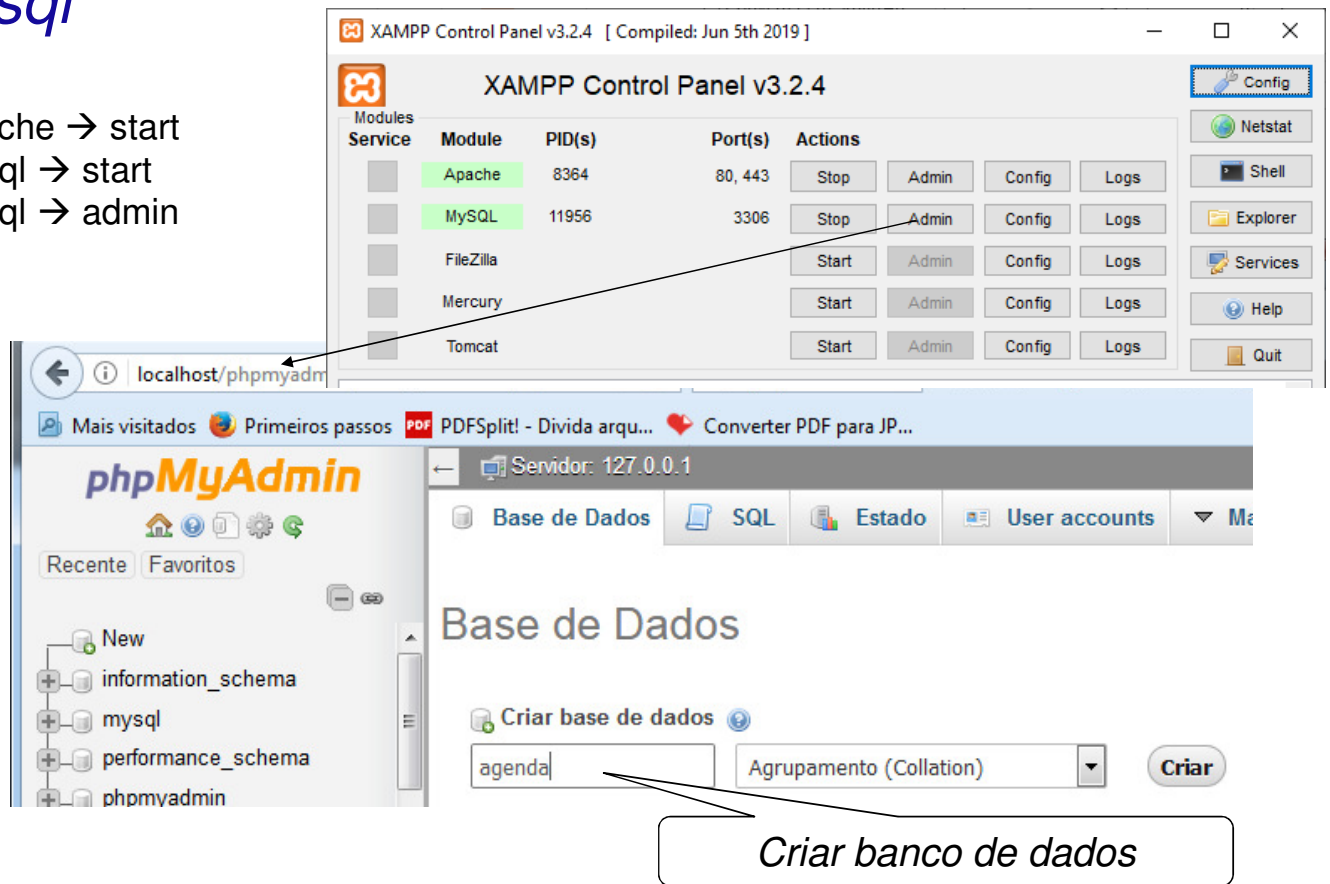
Usuário="postgres" , Senha="ifpb" (ifpb)

Fausto Maranhão Ayres - IFPB

14

Mysql

Apache → start
Mysql → start
Mysql → admin



Usuário="root" , Senha=""

SQL específico de cada Banco

- Verificar se tabela já existe (postgres)

```
PreparedStatement st = con.prepareStatement(  
    "select * from pg_tables where tableowner = 'postgres'  
    and tablename = 'pessoa'");
```

```
ResultSet rs = st.executeQuery();  
if (rs.next()) return; //tabela ja existe
```

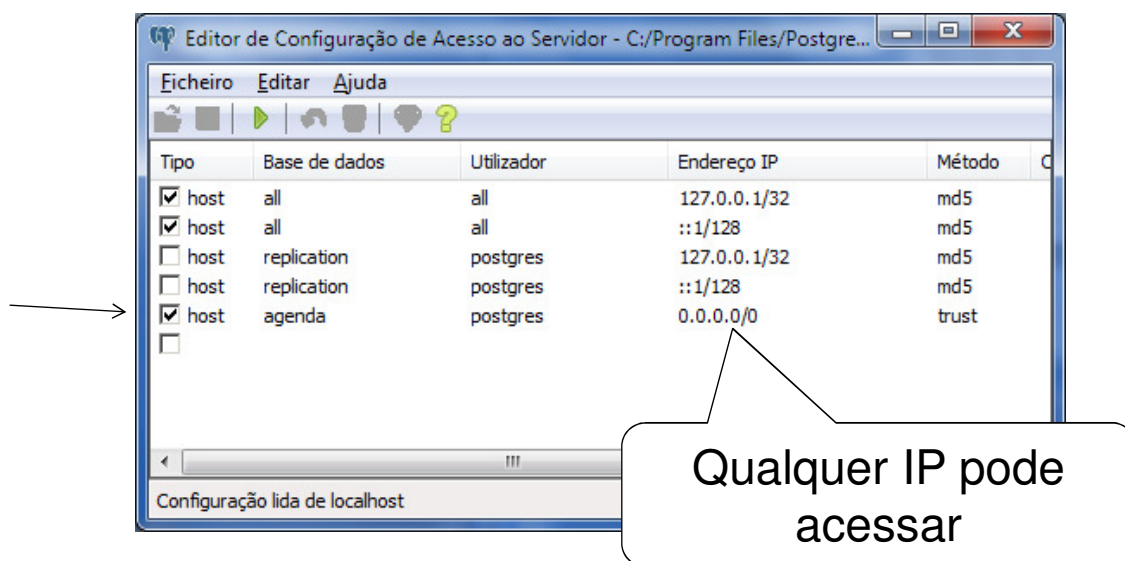

SQL específico de cada Banco

- Verificar se tabela já existe (mysql)

```
PreparedStatement st = con.prepareStatement(  
    "create table IF NOT EXISTS Pessoa(id ....)" );
```

Acesso Cliente-Servidor no postgres

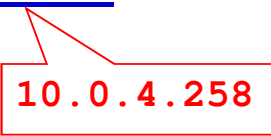
- No computador Servidor faça:
Menu Ferramentas → Config. Servidor → pg_hba.conf



Desativar firewall do servidor

Configurar a aplicação para acessar o servidor

```
try {  
    Connection con = DriverManager.getConnection(  
        "jdbc:postgresql:localhost:5432/agenda",  
        "postgres", "ifpb");  
}  
catch (SQLException e) {  
    System.out.println("erro na conexao");  
}
```



A red box containing the IP address **10.0.4.258** is positioned to the right of the code. A red line points from this box to the underlined word **localhost** in the JDBC URL, indicating that the IP address should be used instead of localhost.

10.0.4.258 IP do servidor