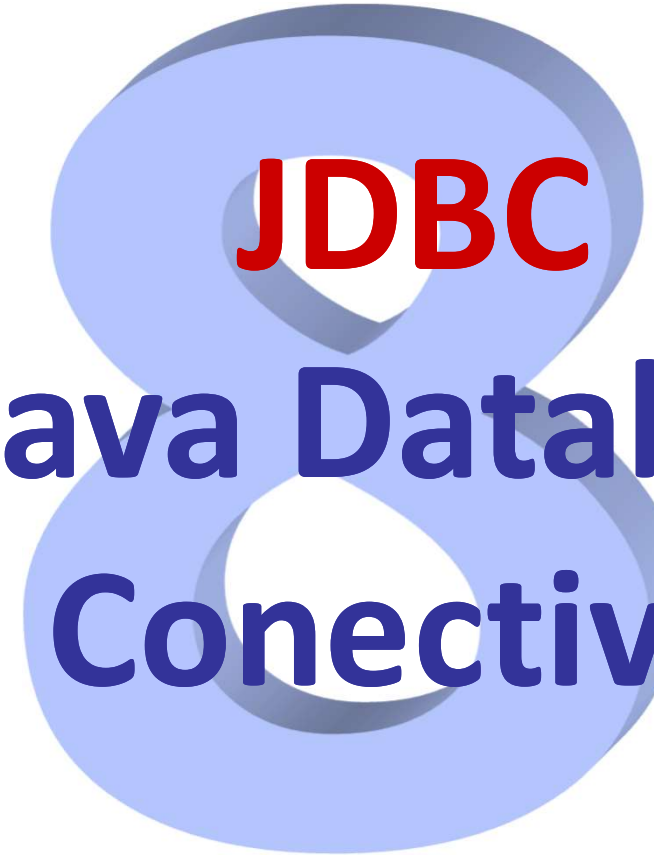


JDBC
Java Database Conectivity

Prof. Alex Sandro: alex@ifpb.edu.br

JDBC

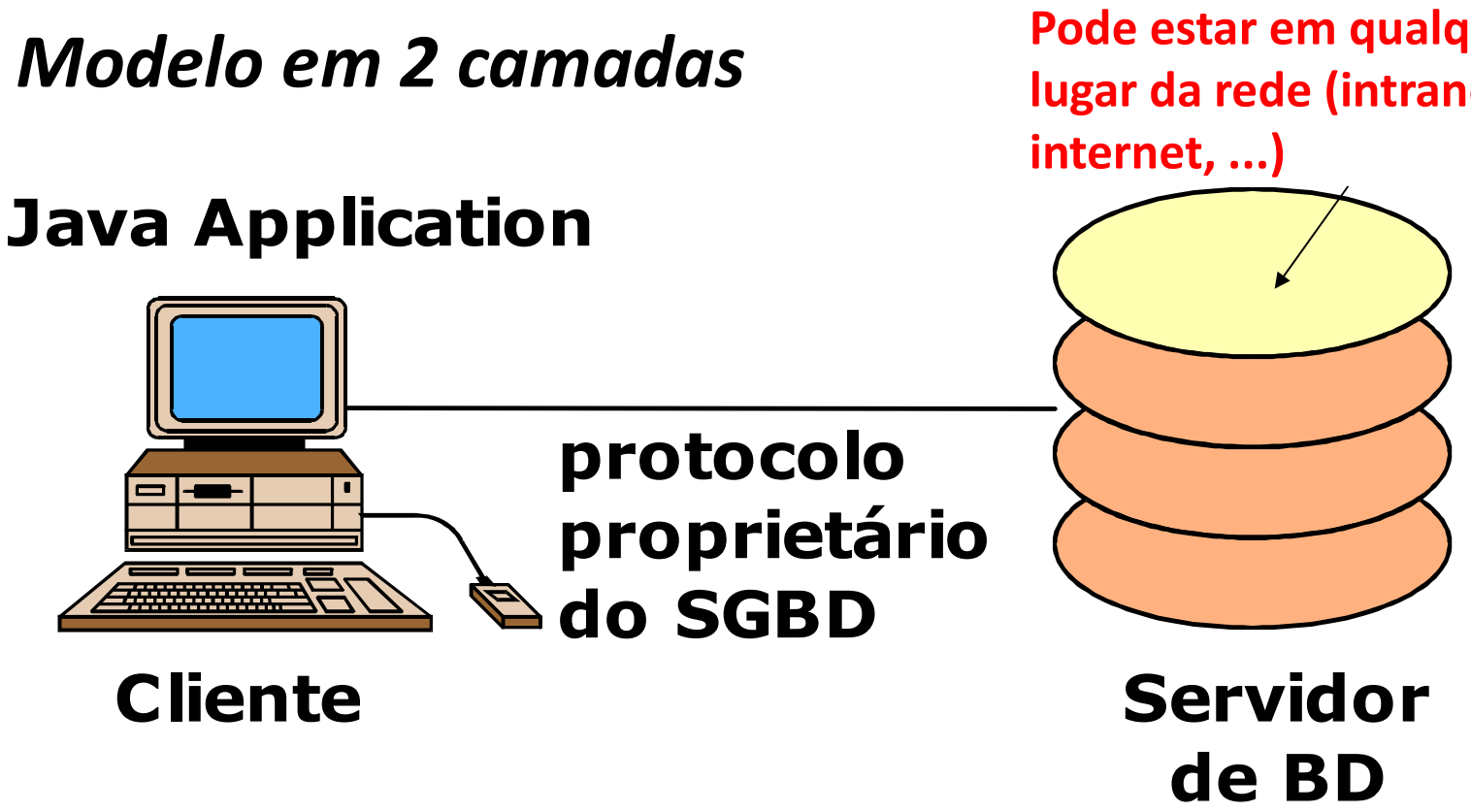
- ***Introdução***
  - *O que é JDBC?*
  - *Amplia o que você pode fazer com java*
- ***Java x JDBC***
  - ✓ *Java: Não é mais necessário escrever um programa para cada plataforma*
  - ✓ *JDBC: Não é mais necessário escrever um programa para cada tipo de banco de dados*
    - *Possibilita o uso de BD's já instalados.*
  - ✓ *Proporcionam computação distribuída*
  - ✓ *Independência de banco de dados*

JDBC

- ***Tarefas***
  - ✓ *Estabelecer uma conexão com o banco de dados;*
  - ✓ *Executar comandos DDL, SQL e DML*
  - ✓ *Recebe um conjunto de resultados*
  - ✓ *Executa storeds procedures*
  - ✓ *Obtém informações sobre o banco de dados (metadados)*
  - ✓ *Executar transações.*

Arquitetura do JDBC

- *Modelo em 2 camadas*



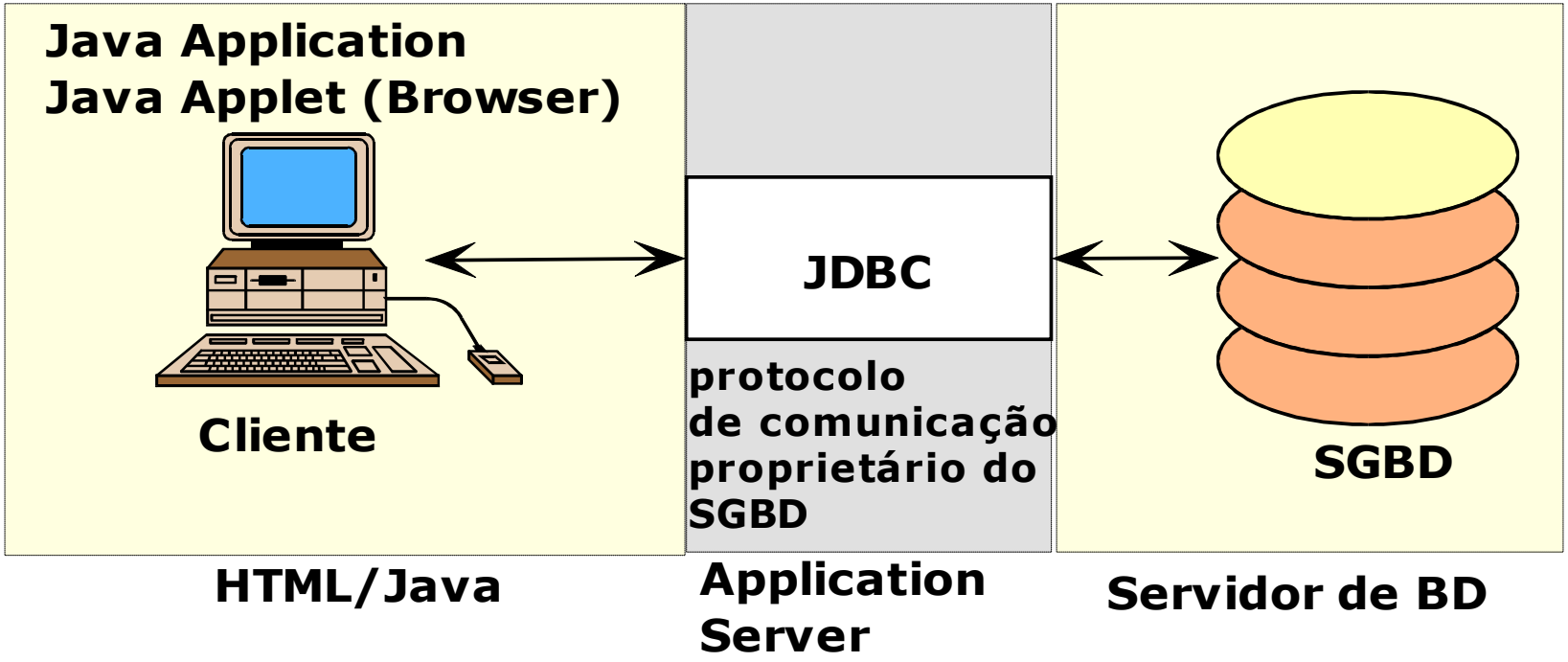
- *Conhecido como config. Cliente/Servidor*
- *Comunicação direta da aplicação ->SGBD*

Arquitetura do JDBC

- ***Modelo em 2 camadas***
  - ***Vantagem***
    - *Menor custo*
    - *Performance*
    - *Gerenciamento da aplicação*
  - ***Desvantagens***
    - *Aplicação ou applet fica muito grande*
    - *Gargalo no número de conexões que o SGBD deve suportar (baixa escalabilidade)*
    - *Tem que instalar nos clientes (Aplicações Java)*

Arquitetura do JDBC

- *Modelo em n camadas*



  - *Presença de uma camada intermediária*

Arquitetura do JDBC

- ***Modelo em n camadas***
  - ***Vantagem***
    - *Utilização de Banco de Dados sem servidor (Dbase, Paradox, Access, etc.)*
    - *Pode adicionar um pool de conexões ao BD no middleware*
    - *Possibilidade de uma camada de segurança;*
    - *Applet "leve"*
  - ***Desvantagens***
    - *Maior complexidade no gerenciamento*
    - *Maior custo*

Transações no JDBC

- ***Modelo em n camadas***
  - *Cada nova conexão é iniciada no modo **autocommit***
  - *O **autocommit** pode ser desabilitado;*
  - *Após um commit ou rollback, automaticamente é iniciada uma nova transação.*

```
conection.setAutoCommit( false );
```

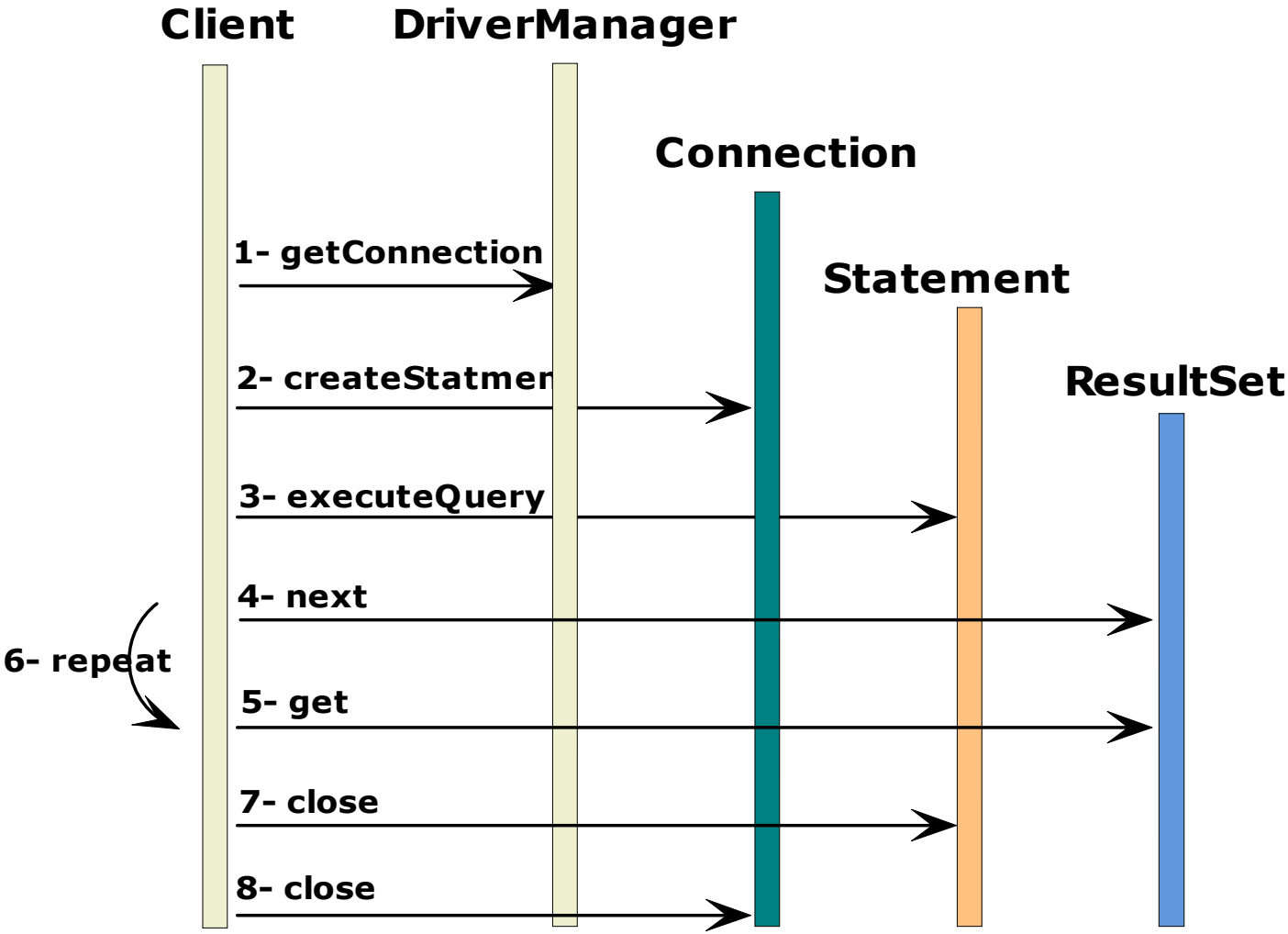
Interface JDBC

- ***O necessário para a fase inicial…***

Interface	*Descrição*
Connection	Representa a conexão com o banco de dados especificado. Dentro do contexto da conexão, cláusulas SQL são executadas e os resultados são devolvidos
DatabaseMetaData	Usado para obter informações sobre as bases, tabelas, índices, tipos de campos, etc.
Statement	Usado para executar cláusulas SQL que se repetem várias vezes de maneira eficiente e executar consultas parametizadas. Instâncias de PreparedStatement contém cláusulas SQL já compiladas.
CallableStatement	Usado para executar SQL Stored Procedure
ResultSet	Usado para acessar os dados retornados pelas cláusulas SQL
ResultSetMetaData	Usado para obter informações sobre o conjunto dos resultados de uma cláusula SQL

Interface JDBC

- *Cenário de aplicação*



SQLServer JDBC driver

- ***Considerações***
  - ✓ *Qualquer driver que você venha a utilizar possui uma documentação com relação ao modo de uso.*
  - ✓ *A versão utilizada aqui acessa SGBDs na versão 6.5, 7.0, 2000 e 2005 para uso com protocolo TCP/IP*
    - ***Também acessa o SBGD Sybase***
  - ✓ *API utilizada nesta aula:* ***jtds-1.3.0.jar*** *JDBC 3.0(na página do curso)*
    - *Lembre-se que o código JDBC para acesso a dados é praticamente o mesmo, independente do SGBD utilizado*

SQLServer JDBC – Passo-a-passo

1. **Importação do pacote jdbc**

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

2. **Registro do driver**

   ✓ Método **forName()**

   `Class.forName(net.sourceforge.jtds.jdbc.Driver);`

3. **Sintaxe da Url de conexão**

```
"jdbc:jtds:<server_type>://<server>:<port>"
```

   server_type: tipo do servidor (SQLServer ou Sybase)

   server: nome do computador ou IP onde se encontra instalado o SGBD

   port: 1433 (default na instalação)

SQLServer JDBC – Passo-a-passo

4. Abrindo uma conexão

```
String url =
"jdbc:jtds:sqlserver://192.168.0.20:1433"

String user = "sa";
String password = "";

Connection con = DriverManager.getConnection
(url, user, password );
/* Essa linha eh exclusiva para SQLServer
   Ela faz a seleção do Database */
String bd = "javamail";
con.setCatalog( bd );
```

SQLServer JDBC – Passo-a-passo

5. Obter um Statement

```
/* É necessário para enviar cláusulas SQL */
Statement st = con.createStatement();
```

6. Preparar e executar a consulta

```
/* É necessário para enviar cláusulas SQL */
String query ="Select nome,matr FROM si_usuario"
ResultSet rs = st.executeQuery( query );
```

7. Agora é só usar!

```
while ( rs.next() ) {
   System.out.print("Nome: " +
                        rs.getString(1));
   System.out.println("Matricula: " +
                           rs.getInt(2));


}
```

SQLServer JDBC – Passo-a-passo

8. **Não esqueça de liberar os objetos e fechar a conexão**

```
st.close(); // Fechando o Statement
rs.close(); // Liberando o conjunto ativo
con.close(); // Fechando a conexão
```

9. **Todo o código**

   Veja aqui!

Outros métodos do JDBC

- **Para atualização (insert, update, delete)**

```
String updateStr = "UPDATE si_usuarios
SET senha='12345' WHERE login = 'alex'";
Statement st;
st.executeUpdate( updateStr );
st.executeUpdate("Create Table teste( id
number, nome varchar)");
st.executeUpdate("INSERT INTO teste
VALUES (10,'Testando')");
st.executeUpdate("DELETE FROM usuario");
```

Outros métodos do JDBC

- **Uso de Statements parametrizados**

```
PreparedStatment upTeste =
con.prepareStatment("SELECT login,senha
FROM si_usuarios WHERE login = ? AND
senha = ?");

upTeste.setString(1,"alex");
upTeste.setInt(2, 12345);
upTeste.executeQuery();
/* Se fosse atualização:
   upTeste.executeUpdate() */
```

Bibliografia

- *http://java.sun.com*
- *Curso de Banco de Dados II – Prof. Alex. Disponível em: www.ffm.com.br/~cunha*