

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação com Acesso a BD

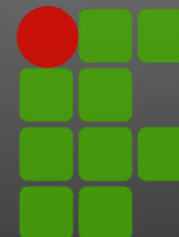
Acesso a Bancos de Dados (JDBC)

Autor: Cleyton Maciel (clayton.maciел@ifrn.edu.br)

Adaptação: Pedro Baesse (pedro.baesse@ifrn.edu.br)

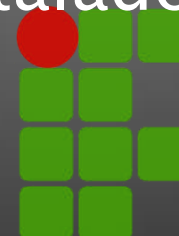
Agenda

- ▶ JDBC
 - Conceito
 - Tipos
- ▶ Transações
- ▶ Tipos SQL e JAVA
- ▶ Conexão JDBC
 - Implementação



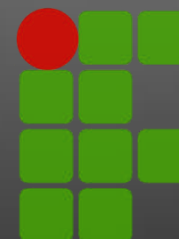
JDBC

- ▶ Java DataBase Connectivity
- ▶ Conjunto de classes e interfaces (API) escritas em Java que faz o envio de cláusulas SQL para qualquer banco de dados relacional
- ▶ Amplia o que você pode fazer com Java
- ▶ Possibilita o uso de BDs já instalados



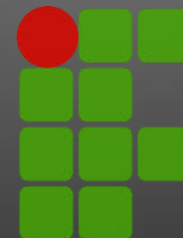
JDBC: Características

- ▶ Fácil mapeamento objeto para relacional
- ▶ Independência de banco de dados
- ▶ Computação distribuída



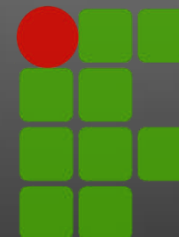
O QUE JDBC FAZ?

1. Estabelece conexão com o banco de dados
2. Executa consultas
3. Recebe o conjunto de resultados das consultas
4. Executa stored procedures
5. Obtém informações sobre o banco de dados, tabelas, índices, visões, e stored procedures
6. Executa transações



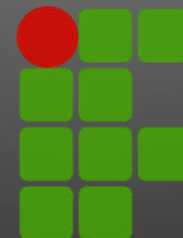
TIPOS DE DRIVERS JDBC

- ▶ Ponte JDBC–ODBC
- ▶ Acesso API–Nativo
- ▶ Acesso por Protocolo de Rede (MiddleWare)
- ▶ Acesso Nativo Java



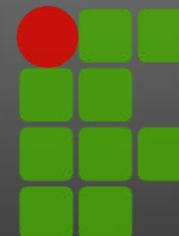
Ponte JDBC-ODBC

- ▶ Convertem chamadas JDBC em chamadas ODBC e as envia ao driver ODBC para acessar o BD causando baixa performance
- ▶ Qualquer BD com driver ODBC pode ser acessada
- ▶ Dependente de plataforma (menos portabilidade)
- ▶ Precisa ser instalado na máquina cliente
- ▶ Deve ser evitado
 - sun jdbc-odbc bridge.



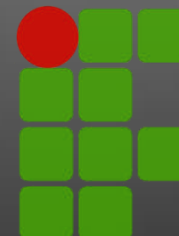
Acesso API-Nativo

- ▶ Convertem chamadas JDBC em chamadas nativas do banco e comunicam-se diretamente com o SGBD
- ▶ Apresentam ganhos de performance comparado JDBC-ODBC
- ▶ Driver precisa ser instalado na máquina cliente e nem todo BD possui essa biblioteca
- ▶ Dependente de plataforma
 - IBM DB2, BEA WebLogic(SQL Server, Oracle e Sybase), InfoZoom(SQL Server e Access), etc.



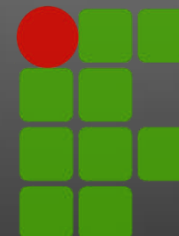
Acesso por Protocolo de Rede

- ▶ Convertem chamadas JDBC em um protocolo de rede independente do SGBD e comunicam-se com um gateway (middleware) que traduz estas requisições para o protocolo específico do SGBD
- ▶ Possibilitam o uso de diferentes SGBDs
- ▶ Esta é a mais flexível alternativa de JDBC
- ▶ Acesso pela internet ou rede liberando o cliente de configuração
- ▶ Precisam de cuidados adicionais com a segurança
 - DataDirect (SQL Server), CONNX (DB2, SQL Server, Access, Oracle, Sysbase, Informix), etc.

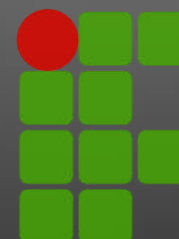
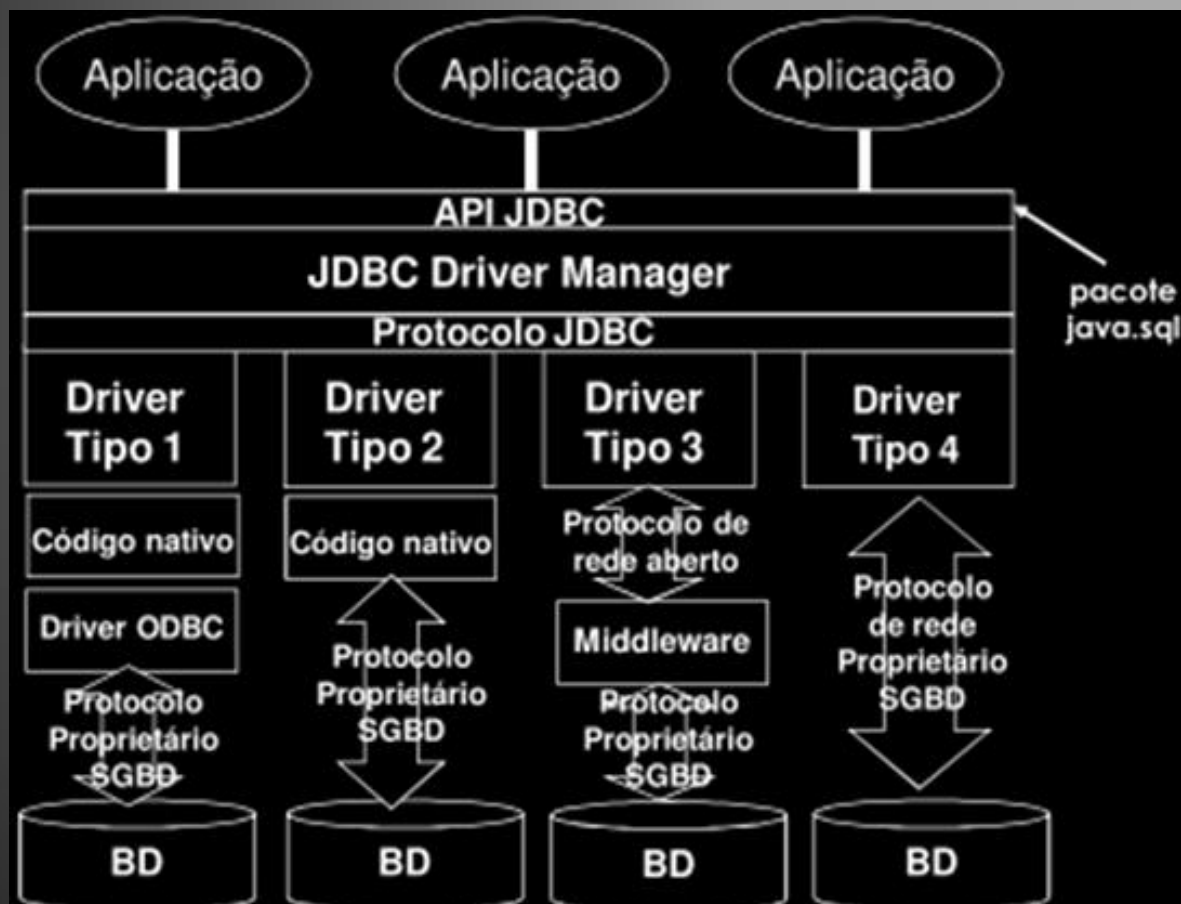


Acesso Nativo em Java

- ▶ Convertem chamadas JDBC para um protocolo usado diretamente pelo SGBD, sem uma camada intermediária
 - Melhor performance
- ▶ Independentes de plataforma (Puro Java)
- ▶ Instalado na JVM do cliente (pacote java.sql)
- ▶ São geralmente específicos para determinada base de dados
 - Atinav (SQL Server), BEA WebLogic(SQL Server e Informix), DataDirect(Oracle e SQL Server), etc.

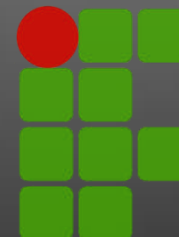


ARQUITETURA JDBC



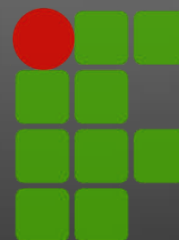
DRIVER MANAGER

- ▶ A classe DriverManager é uma camada de gerenciamento do JDBC;
- ▶ Responsável pela interface entre os usuários e os drivers;



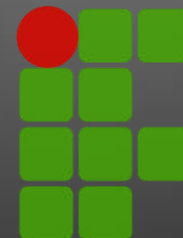
TRANSAÇÕES EM JDBC

- ▶ JDBC provê um modelo de controle de transações
- ▶ Cada nova conexão é iniciada no modo autocommit
- ▶ JDBC implicitamente emite um commit após cada Statement executado
- ▶ Podemos desabilitar o autocommit
- ▶ Após um commit ou um rollback, automaticamente é iniciada uma nova transação
 - Ex: `con.setAutoCommit(false);`
...
`con.commit();`



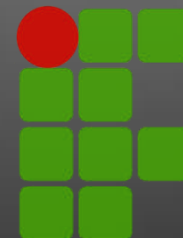
Tipos entre SQL e Java

SQL	JAVA
CHAR	String
VARCHAR	String
LONGVARCHAR	String
NUMERIC	java.lang.BigDecimal
DECIMAL	java.lang.BigDecimal
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int



Tipos entre SQL e Java

SQL	JAVA
BIGINT	long
REAL	float
FLOAT	double
DOUBLE	double
BINARY	byte[]
VARBINARY	byte[]
LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp



Pacote java.sql

► 5 passos básicos

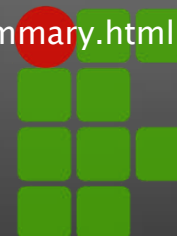
1. Registro do driver na aplicação
2. Abertura da conexão com o SGBD
3. Execução de comandos SQL
4. Recuperação dos resultados
5. Fechamento da conexão

► Principais classes

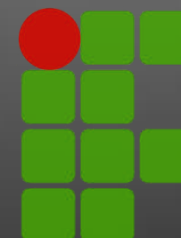
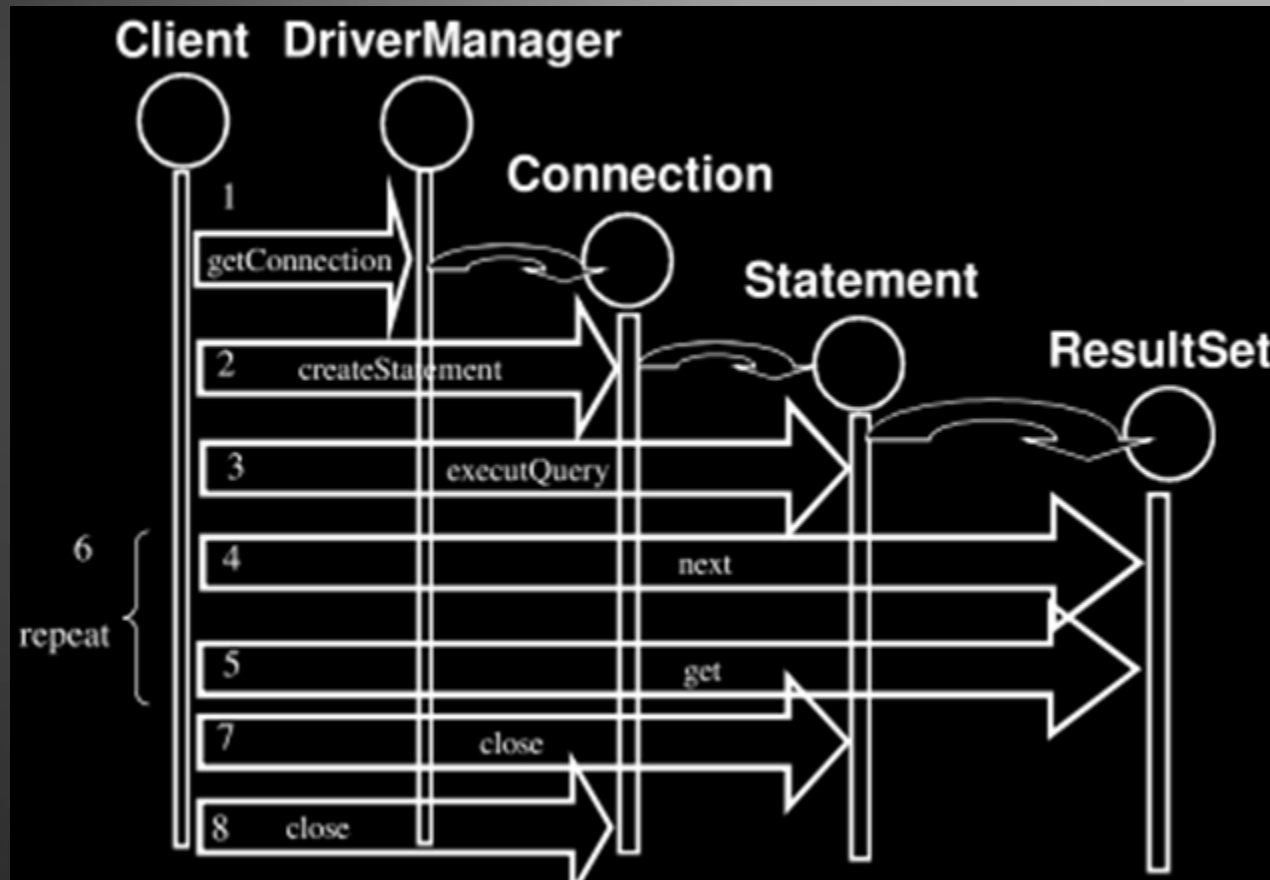
- **DriverManager**: responsável por criar uma conexão com o banco de dados
- **Connection**: classe responsável por manter uma conexão aberta com o banco de dados
- **Statement**: gerencia e executa instruções SQL
- **PreparedStatement**: pré-compila e aceita parâmetros nas instruções SQL
- **ResultSet**: responsável por receber e apresentar os dados obtidos do banco de dados

► Referência

- <http://java.sun.com/javase/6/docs/api/java/sql/package-summary.html>



Uma aplicação JDBC



JDBC – Passos Básicos

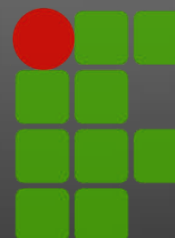
1. Registro do driver

- O driver é registrado automaticamente quando a classe é carregada na aplicação
- Carrega o driver em tempo de execução

```
Class.forName("org.postgresql.Driver")
```

```
Class.forName("com.mysql.jdbc.Driver")
```

```
Class.forName("oracle.jdbc.OracleDriver")
```

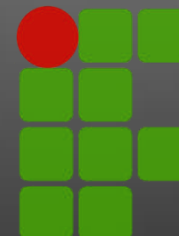


JDBC – Passos Básicos

2. Abertura da conexão

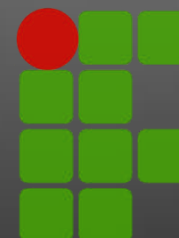
- Depois do registro do driver, é necessário fornecer informações ao DriverManager para realizar a conexão com o Banco de Dados

```
DriverManager.getConnection(url,"usuario","senha");
```



Connection

- ▶ Representa a conexão com o banco de dados
- ▶ Métodos desta classe frequentemente utilizados (SUN, 2007):
 - **commit()**, executa todas as alterações feitas com o banco de dados pela atual transação
 - **rollback()**, desfaz qualquer alteração feita com o banco de dados pela atual transação
 - **close()**, libera o recurso que estava sendo utilizado pelo objeto



JDBC – Passos Básicos

► Conexão com o SGBD:

```
DriverManager.getConnection  
(url,"login","senha");
```

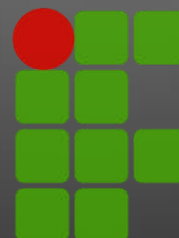
◦ url: URL de conexão JDBC

- jdbc:postgresql://localhost: 5432/cursodb
- jdbc:mysql://localhost:3306/cursodb
- dbc:oracle:thin:@localhost:1521:XE

- login: usuário com direitos de acesso ao banco de dados;
- senha: senha para autenticação.

► Sintaxe geral de urls

- “jdbc:<subprotocolo>://<servidor>:<porta>/<banco_de_dados>”



JDBC – Passos Básicos

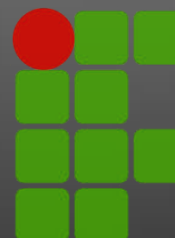
3. Execução de comandos SQL é feita por meio de um Statement

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("Select a, b, c, d from Table1");

while (rs.next()){
    int a = rs.getInt(1); // int a = rs.getInt("a"); //
    BigDecimal b = rs.getBigDecimal(2,0);
    char c[] = rs.getString(3).toCharArray();
    boolean d = rs.getBoolean(4);
}
```

4. Recuperação dos resultados é feita pelo ResultSet

- O cursor de um ResultSet é inicialmente posicionado antes da primeira linha da tabela.



Statement

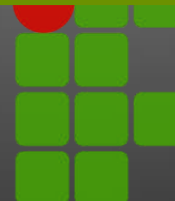
- ▶ Fornece métodos para executar uma instrução SQL
- ▶ Não aceita a passagem de parâmetros
- ▶ Principais métodos da classe Statement são (SUN, 2007):
 - `executeUpdate()`: executa instruções SQL do tipo: INSERT, UPDATE e DELETE;
 - `executeQuery()`: executa instruções SQL de busca de dados, do tipo: SELECT;
 - `close()`: libera o recurso que estava sendo utilizado pelo objeto.

```
//Instanciando o objeto statement (stmt)
```

```
Statement stmt = conn.createStatement();
```

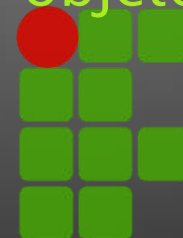
```
//Executando uma instrução SQL
```

```
stmt.executeUpdate("INSERT INTO ALUNO VALUES (1, 'Pedro da Silva')");
```



ResultSet

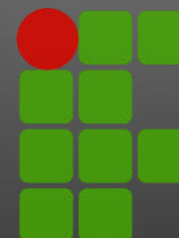
- ▶ Permite o recebimento e gerenciamento do conjunto de dados resultante de uma consulta SQL
- ▶ Métodos freqüentemente utilizados (SUN, 2007):
 - **next()**: move o cursor para a próxima linha de dados, já que o conjunto de dados retornados pela consulta SQL é armazenado como em uma tabela
 - **close()**: libera o recurso que estava sendo utilizado pelo objeto
 - **getString(String columnName)**: recupera o valor da coluna informada como parâmetro, da linha atual do conjunto de dados recebidos pelo objeto ResultSet



ResultSet

```
//Recebendo o conjunto de dados da consulta SQL
ResultSet rs = stmt.executeQuery("SELECT id, nome FROM
ALUNO");

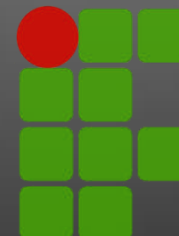
// Se houver resultados, posiciona-se o cursor na próxima
linha de dados
while (rs.next()) {
    // Recuperando os dados retornados pela consulta SQL
    int id = rs.getInt("id");
    String nome = rs.getString("nome");
}
```



Acessando DB em duas camadas

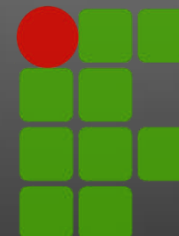
5. Fechamento da conexão

```
stmt.close();  
con.close();
```



PreparedStatement

- ▶ Statement pré-compilado
 - Apresenta ganhos de eficiência quando várias consultas similares são enviadas com parâmetros diferentes
- ▶ Uma String com a instrução SQL é preparada previamente, deixando-se “?” no lugar dos parâmetros
- ▶ Parâmetros são inseridos em ordem, com setXXX() onde XXX é um tipo igual aos retornados pelos métodos de ResultSet

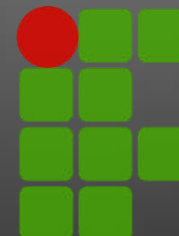


PreparedStatement

...

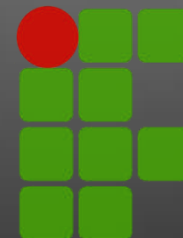
```
String sql = "INSERT INTO Conta VALUES(?, ?, ?)";  
PreparedStatement cstmt =  
con.prepareStatement(sql);  
cstmt.setString(1, "123");  
cstmt.setDouble(2, 0);  
cstmt.setString(3, "C");  
cstmt.executeUpdate();
```

...



STORED PROCEDURES COM JDBC

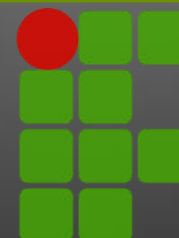
- ▶ Permite usar as particularidades do SGBD;
- ▶ Diminui a portabilidade através dos SGBD's;
- ▶ Portabilidade – SQL 92;



Stored Procedures

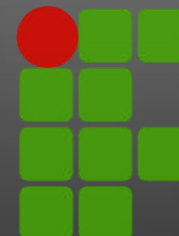
- ▶ Stored Procedures (Proc. Armazenados)
 - Procedimentos desenvolvidos em linguagem proprietária do SGBD (PL-SQL, Transact-SQL, etc)
 - Podem ser chamados através de objetos CallableStatement
 - Parâmetros são passados da mesma forma que em instruções PreparedStatement

```
con.prepareCall("{call proc_update}");  
con.prepareCall("{call proc_update(?)}");  
con.prepareCall("{? = call proc_update(?)}");
```



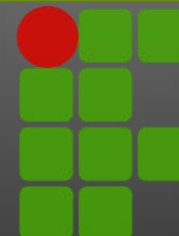
Metadados

- ▶ Classe DatabaseMetaData
 - Permite obter informações relacionadas ao banco de dados
- ▶ Classe ResultSetMetaData
 - Permite obter informações sobre o ResultSet, como por exemplo quantas colunas e linhas existem na tabela de resultados, o nome das colunas, etc.

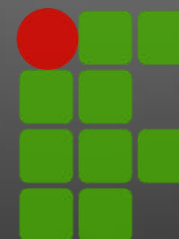


Metadados

```
...  
Connection con;  
DatabaseMetaData dbdata = con.getMetaData();  
String dbname = dbdata.getDatabaseProductName();  
...  
ResultSet rs;  
ResultSetMetaData meta = rs.getMetaData();  
int col = meta.getColumnCount();  
String [] nomeCols = new String[col];  
for (int i=0; i<col; i++)  
    nomeCols[i] = meta.getColumnName(i);  
...
```



Exemplo



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Dúvidas

