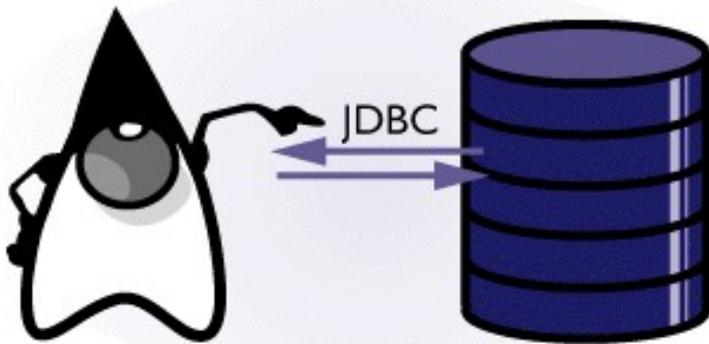
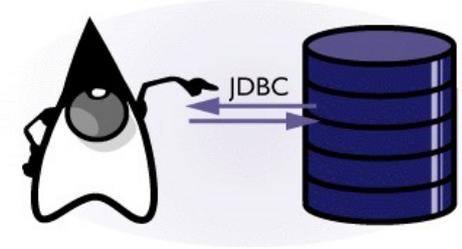


Java & Bancos de Dados

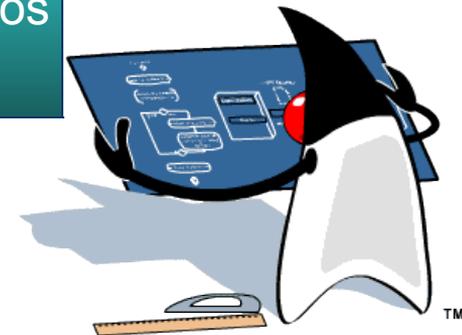
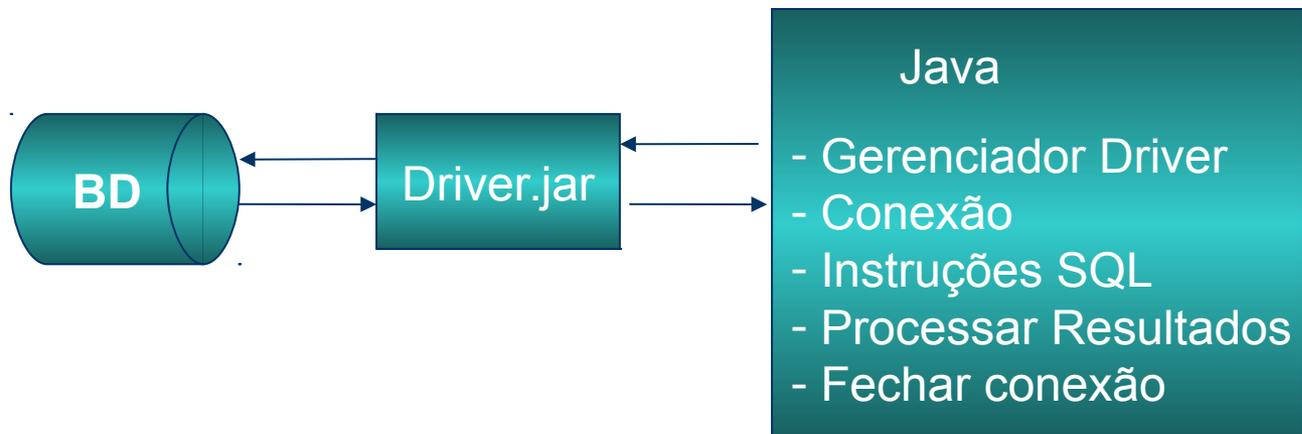
Adaptado de Slides da
Universidade Salgado de Oliveira
Goiânia
Sistemas de Informação – Ling.Prog.V
Prof. Roberto Couto Lima
Fonte: Curso Java
Prof. Fabricio Leonard Leopoldino (Universo)

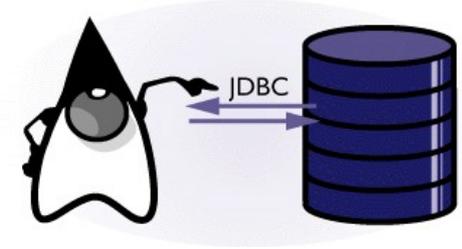




Acesso a banco de dados

Estrutura típica para que uma aplicação Java tenha acesso a um banco de dados relacional:

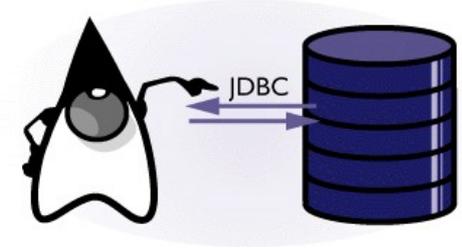




Acesso ao banco de dados

Para que uma aplicação Java tenha acesso a um banco de dados relacional os seguintes passos devem ser atendidos:

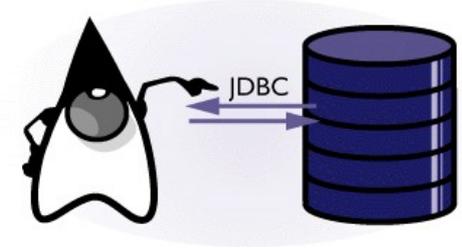
- Habilitar o **driver JDBC** a partir da aplicação cliente;
- Estabelecer uma **conexão** entre a aplicação cliente e servidor do banco de dados;
- Montar e executar a instrução SQL desejada; e
- Processar no cliente o **resultado** da consulta.



Acesso ao banco de dados

Principais Falhas:

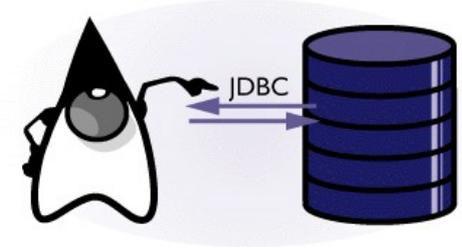
- Banco de Dados não iniciado;
- Driver não Encontrado
- Driver incompatível;
- URL – Configurada de forma incorreta (nome banco, usuário, senha, servidor)
- Classe de conexão (inexistente, não instanciada, com erros)
- Usuário sem direito de acesso
- Falta de tratamento de Exceções try catch
- Classe java.sql não instanciada



Arquitetura da JDBC

As principais classes e interfaces do pacote `java.sql` são:

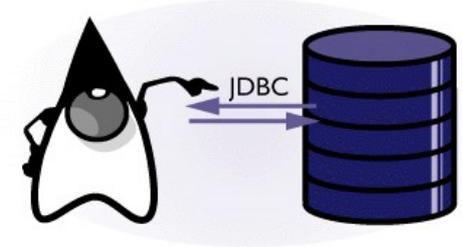
- **DriverManager** - gerencia o driver e cria uma conexão com o banco.
- **Connection** - é a classe que representa a conexão com o bando de dados.



Arquitetura da JDBC

As principais classes e interfaces do pacote `java.sql` são:

- **Statement** - controla e executa uma instrução SQL .
- **PreparedStatement** - controla e executa uma instrução SQL preparada.
- **ResultSet** - contém o conjunto de dados retornado por uma consulta SQL.
- **ResultSetMetaData** - é a classe que trata dos metadados do banco.



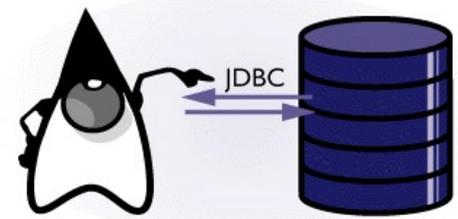
Utilizando a JDBC

Para a aplicação Java se comunicar com um banco de dados e acessar os seus dados, uma conexão com o BD deve ser estabelecida.

A conexão é estabelecida de seguinte forma:

- Carregamento do driver JDBC específico
- Criação da conexão com o BD

A partir da conexão é possível interagir com o BD, fazendo consultas, atualização e busca às meta informações da base e das tabelas.

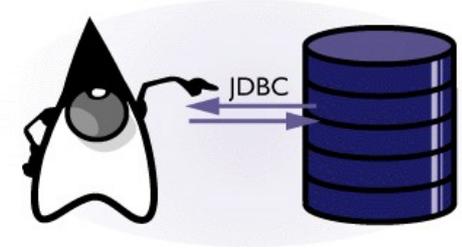


Carregando o driver JDBC

Carregamento do driver JDBC específico:

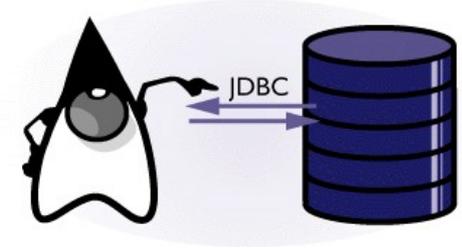
```
Class.forName( "StringNomeDriveJDBC" );
```

A String (**StringNomeDriveJDBC**) passada ao método `forName()` é o nome completo qualificado da classe que implementa o Driver JDBC de cada banco de dados. Lembrando que cada driver possui um nome diferente, consulte a documentação do fabricante.



Utilizando a JDBC

```
9 public class ConexaoBancoDeDados {
10
11     static String driverJDBC = "org.gjt.mm.mysql.Driver";
12
13     public static void main(String[] args) {
14
15         try{
16             System.out.println("Carregando o driver JDBC...");
17             Class.forName(driverJDBC);
18             System.out.println("Driver carregado com sucesso!!");
19         } catch (Exception e){
20             System.out.println("Falha no carregamento!!");
21         }
22     }
23 }
```



Exemplo de drivers e conexões

JDBC:ODBC:

- sun.jdbc.odbc.JdbcOdbcDriver
- jdbc:odbc:<alias>

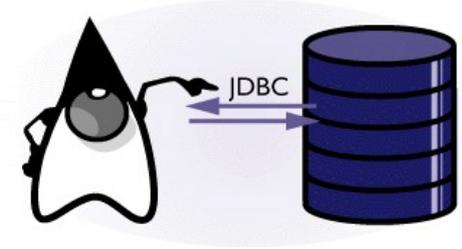
1. Driver
2. url

Microsoft MSSQL Server JDBC Driver:

- com.microsoft.jdbc.sqlserver.SQLServerDriver
- jdbc:microsoft:sqlserver://<server_name>:<1433>

Firebird:

- org.firebirdsql.jdbc.FBDriver
- jdbc:firebirdsql:[//host[:port]/]<database>



Utilizando a JDBC

Oracle:

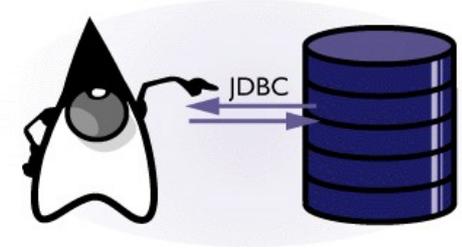
- oracle.jdbc.driver.OracleDriver
- jdbc:oracle:thin:@<server>[:<1521>]:<database_name>

Postgresql:

- org.postgresql.Driver
- jdbc:postgresql:[<///host>[:<5432>/]]<database>

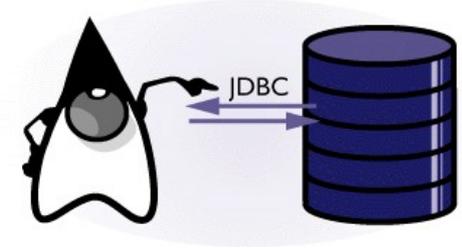
MySql:

- org.gjt.mm.mysql.Driver
- jdbc:mysql://<hostname>[:<3306>]/<dbname>



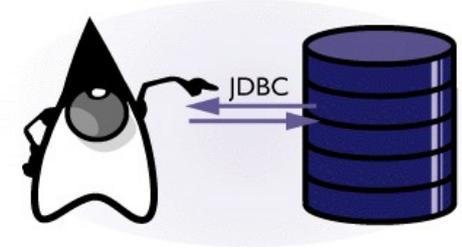
Carregando o driver do MySQL

```
9 public class ConexaoBancoDeDados {
10
11     static String driverJDBC = "org.gjt.mm.mysql.Driver";
12
13     public static void main(String[] args) {
14
15         try{
16             System.out.println("Carregando o driver JDBC...");
17             Class.forName(driverJDBC);
18             System.out.println("Driver carregado com sucesso!!");
19         } catch (Exception e){
20             System.out.println("Falha no carregamento!!");
21         }
22     }
23 }
```



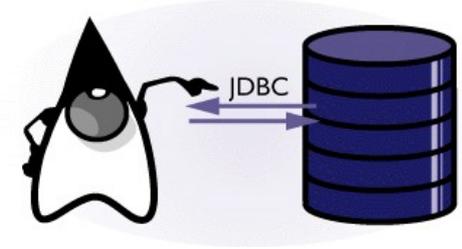
Carregando o driver do Oracle

```
9 public class ConexaoBancoDeDados {
10
11     static String driverJDBC = "oracle.jdbc.driver.OracleDriver";
12
13     public static void main(String[] args) {
14
15         try{
16             System.out.println("Carregando o driver JDBC...");
17             Class.forName(driverJDBC);
18             System.out.println("Driver carregado com sucesso!!");
19         } catch (Exception e){
20             System.out.println("Falha no carregamento!!");
21         }
22     }
23 }
```



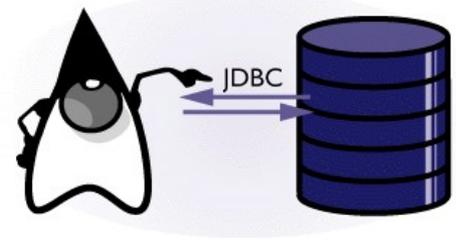
Carregando o driver do PostgreSQL

```
9 public class ConexaoBancoDeDados {
10
11     static String driverJDBC = "org.postgresql.Driver";
12
13     public static void main(String[] args) {
14
15         try{
16             System.out.println("Carregando o driver JDBC...");
17             Class.forName(driverJDBC);
18             System.out.println("Driver carregado com sucesso!!");
19         } catch (Exception e){
20             System.out.println("Falha no carregamento!!");
21         }
22     }
23 }
```



Carregando o driver da JDBC:ODBC

```
9 public class ConexaoBancoDeDados {
10
11     static String driverJDBC = "sun.jdbc.odbc.JdbcOdbcDriver";
12
13     public static void main(String[] args) {
14
15         try{
16             System.out.println("Carregando o driver JDBC...");
17             Class.forName(driverJDBC);
18             System.out.println("Driver carregado com sucesso!!");
19         } catch (Exception e){
20             System.out.println("Falha no carregamento!!");
21         }
22     }
23 }
```



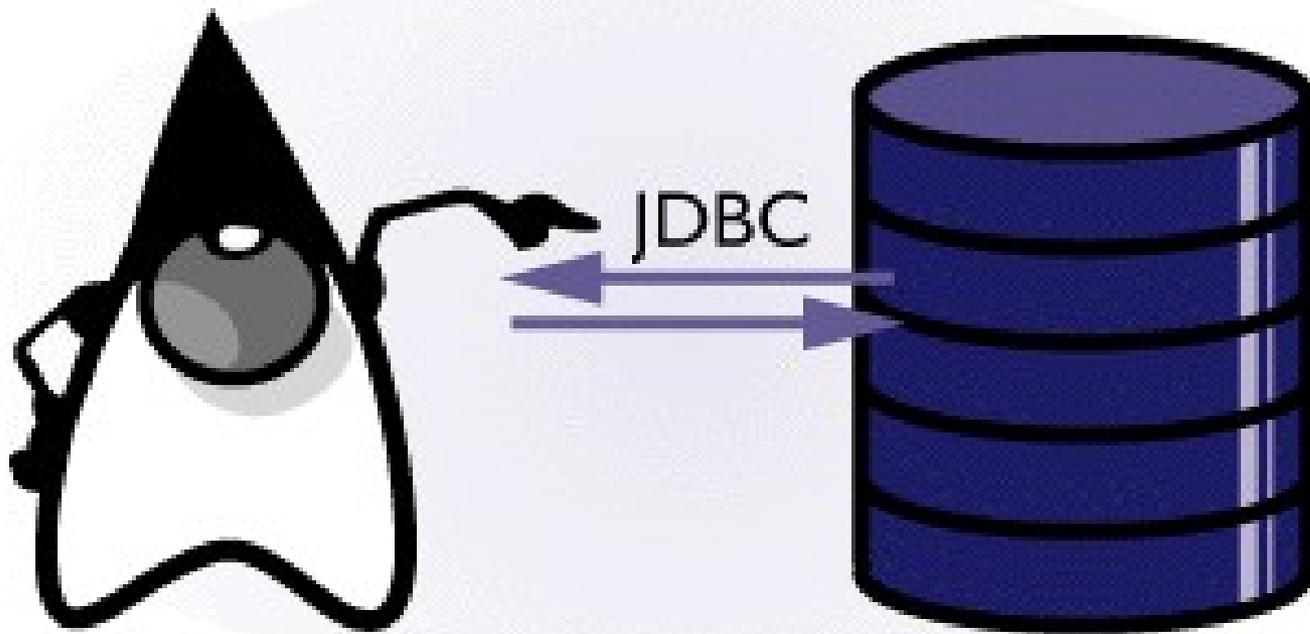
Conectando ao Banco de Dados

Criação da conexão com o BD:

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

Após o carregamento do driver, a classe `DriverManager` é a responsável por se conectar ao banco de dados e devolver um objeto **Connection**, que representa a conexão com o BD.

Exemplo de conexão ao banco de dados MySQL



```
12 public class ConexaoBancoDeDados {
13
14     static String driverJDBC = "org.gjt.mm.mysql.Driver";
15     static String url = "jdbc:mysql://localhost:3306/reservas";
16     static String user = "root";
17     static String senha = "root";
18
19     public static void main(String[] args) {
20
21         try{
22             System.out.println("Carregando o driver JDBC...");
23             Class.forName(driverJDBC);
24             System.out.println("Driver carregado com sucesso!!");
25         } catch (Exception e){
26             System.out.println("Falha no carregamento!!");
27         }
28
29         try{
30             System.out.println("Conectando ao banco...");
31             Connection conexao = DriverManager.getConnection(url, user, senha);
32             System.out.println("Conexão efetuada com sucesso!!!");
33         } catch (Exception e){
34             System.out.println("Falha na conexão!!");
35         }
36     }
37 }
```

Statement

- Implementação de uma Interface que fornece métodos para executar uma instrução SQL;
- Não aceita a passagem de parâmetros;
- principais métodos da Interface *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')");
```

PreparedStatement

- A interface PreparedStatement possui todos os recursos da interface Statement;
- acrescentando a utilização de parâmetros em uma instrução SQL;
- métodos da interface PreparedStatement são (SUN, 2007):
 - *execute()*, consolida a instrução SQL informada;
 - *setDate()*, método utilizado para atribuir um valor do tipo Data;
 - *setInt()*, utilizado para atribuir valores do tipo inteiro;
 - *setString()*, método utilizado para atribuir valores do tipo Alfa Numéricos.

PreparedStatement

```
// Instanciando o objeto preparedStatement (pstmt)
PreparedStatement pstmt =
    conn.prepareStatement("UPDATE ALUNO SET NOME = ?");
// Setando o valor ao parâmetro
pstmt.setString(1, "MARIA RITA");
```

ResultSet

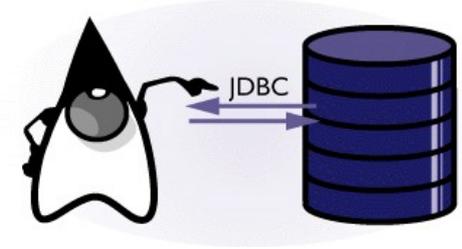
- Esta interface permite o recebimento e gerenciamento do conjunto de dados resultante de uma consulta SQL;
- métodos capazes de acessar os dados;
- métodos desta interface freqüentemente utilizados (SUN, 2007):
 - *next()*, move o cursor para a próxima linha de dados, já que o conjunto de dados retornados pela consulta SQL é armazenado como em uma lista.
 - *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");
}
```

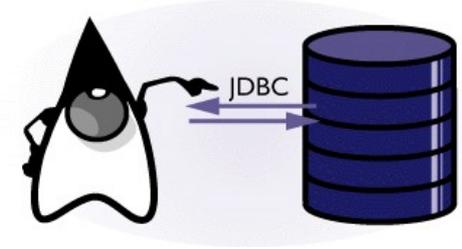
- métodos como o *getInt()*, *getString()* para recuperar os valores;



Utilizando a JDBC

Com a conexão estabelecida já é possível interagir com o BD de várias formas:

- Criar tabelas e outros elementos
- Inserir, Alterar e Remover registros
- Buscar registros
- Buscar as meta informações do banco



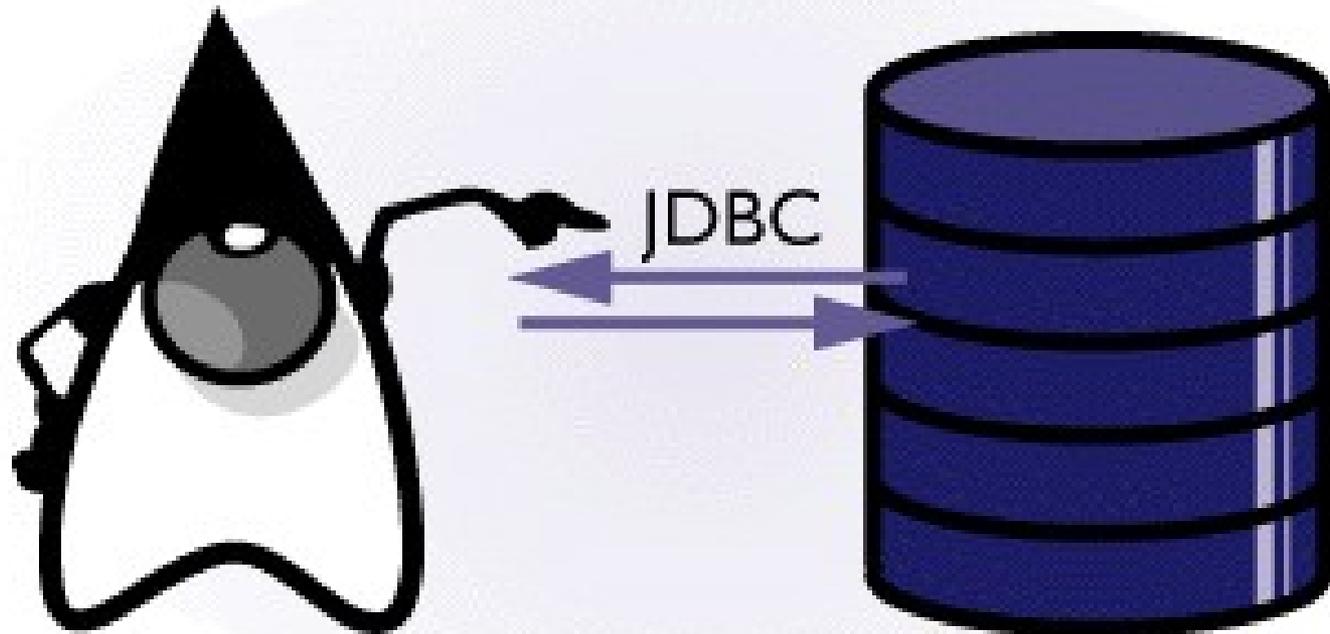
Criando tabelas

Uma nova tabela pode ser criada especificando o seu nome juntamente com os nomes das colunas e seus tipos de dado:

Exemplo do comando SQL

```
CREATE TABLE aluno (  
    matricula int,  
    nome varchar(80),  
    email varchar(80),  
    idade int  
);
```

Exemplo de criação de tabela com a JDBC

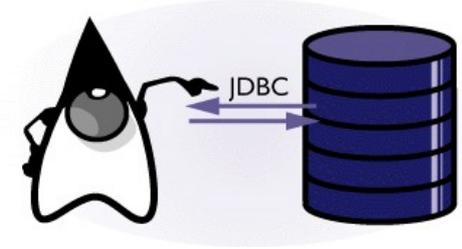


```
13 public class ConexaoBancoDeDados {
14
15     static String driverJDBC = "org.gjt.mm.mysql.Driver";
16     static String url = "jdbc:mysql://localhost:3306/reservas";
17     static String usuario = "root";
18     static String senha = "root";
19 public static void main(String[] args) {
20     Connection conexao = null;
21     Connection con = conexao;
22     Statement st = null;
23     String instrucaoSQL = "CREATE TABLE aluno ( "
24         + "matricula int, nome varchar(80), "
25         + "email varchar(80), idade int)";
26     try{
27         Class.forName(driverJDBC);
28         conexao = DriverManager.getConnection(url, usuario, senha);
29
30         System.out.println("Criando a tabela, aguarde...");
31         st = conexao.createStatement();
32         st.executeUpdate(instrucaoSQL);
33         System.out.println("Tabela criada com sucesso!");
34         st.close();
35         conexao.close();
36     } catch (Exception e){
37         System.out.println("Erro");
38         e.printStackTrace();
39     }
40 }
```

Explicação



1. Através do método `createStatement()` de `Connection` é retornado um objeto `Statement`, que é usado para executar um comando SQL no BD.
2. O método `executeUpdate()` de `Statement` recebe o SQL que será executado. Este método deve ser usado para DDLs e DML
3. Depois os métodos `close()` de `Statement` e `Connection` são invocados para liberar os recursos.



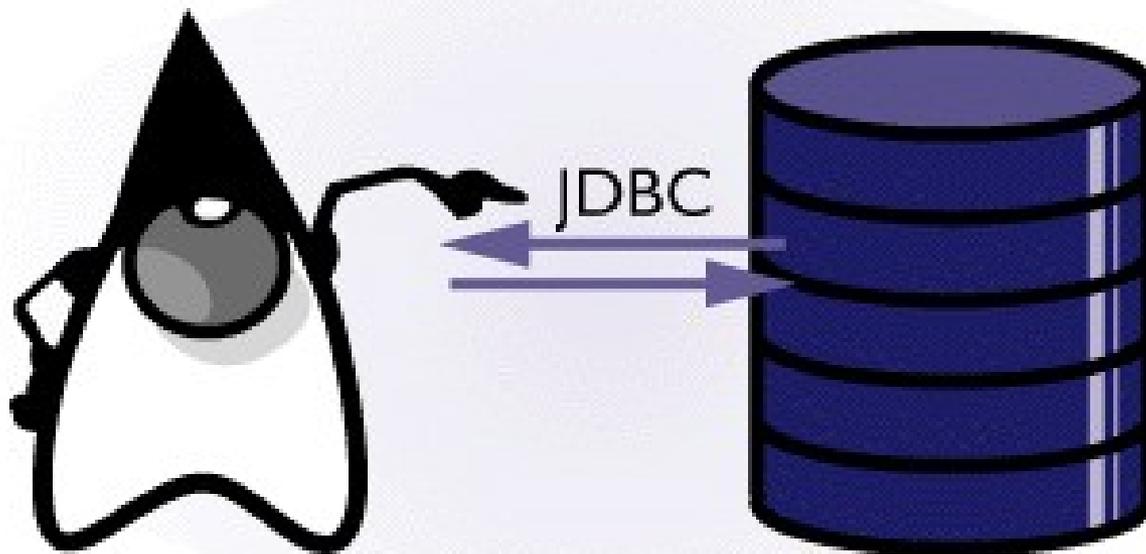
Inserindo dados

O comando INSERT é utilizado para incluir linhas em uma tabelada tabela:

Exemplo do comando SQL

```
INSERT INTO aluno (matricula, nome, email, idade)  
VALUES (1, 'Fabricio', 'fabricio@universo.br', 29);
```

Exemplo de inserção de dados com JDBC



```
13 public class ConexaoBancoDeDados {
14
15     static String driverJDBC = "org.gjt.mm.mysql.Driver";
16     static String url = "jdbc:mysql://localhost:3306/reservas";
17     static String usuario = "root";
18     static String senha = "root";
19 public static void main(String[] args) {
20     Connection conexao = null;
21     Connection con = conexao;
22     Statement st = null;
23     String instrucaoSQL = "INSERT INTO aluno (matricula, nome, email, idade) "
24         + "VALUES (1, 'Fabricio', 'fabricio@universo.br', 29)";
25
26     try{
27         Class.forName(driverJDBC);
28         conexao = DriverManager.getConnection(url, usuario, senha);
29
30         System.out.println("Inserindo dados na tabela...");
31         st = conexao.createStatement();
32         st.executeUpdate(instrucaoSQL);
33         System.out.println("Dados inseridos com sucesso!");
34         st.close();
35         conexao.close();
36
37     } catch (Exception e){
38         System.out.println("Erro");
39         e.printStackTrace();
40     }
41 }
42 }
```

Instrução Preparada

```
public boolean incluir(Object o) {
    String sInstrucaoSQL;
    sMensagemErro = "";
    Cliente cliente = (Cliente) o;
    try {
        objConexaoBD.conecta();
        sInstrucaoSQL = "insert into " + NOME_TABELA + campo+ "values( ?,?,?,?,?,?,?,?,?,?)";
        PreparedStatement stmt = objConexaoBD.con.prepareStatement(sInstrucaoSQL);
        stmt.setInt(1, cliente.getCodCliente());
        stmt.setString(2, cliente.getNome());
        stmt.setString(3, cliente.getEndereco());
        stmt.setString(4, cliente.getFone());
        stmt.setInt(5, cliente.getIdade());
        stmt.setString(6, cliente.getSexo());
        stmt.setFloat(7, (float) cliente.getRendaMensal());
        stmt.execute();
        stmt.close();
        objConexaoBD.desconecta();
    }
    catch(SQLException ex) { sMensagemErro = "SQLException: " + ex.getMessage();
    }
    if (sMensagemErro.equals("")) return true;
    else return false;
}
```

Apoio da Ferramenta Case



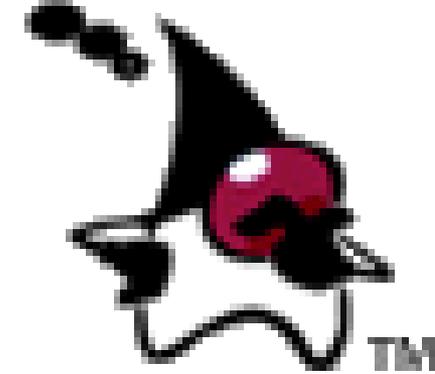
- Criar uma tabela com a seguinte estrutura

Alunos		
<u>alunoMatricula</u>	<u>varchar(5)</u>	<u><pk></u>
alunoNome	varchar(80)	
alunoEmail	varchar(80)	
alunoIdade	int	

← Tabela

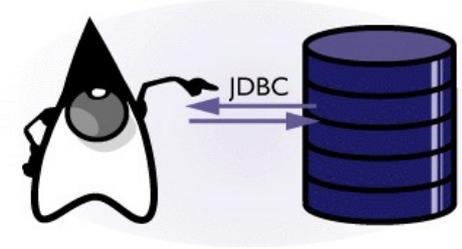
Instrução SQL →

```
1 create table alunos
2 (
3     alunoMatricula varchar(5) not null,
4     alunoNome      varchar(80),
5     alunoEmail     varchar(80),
6     alunoIdade     int,
7     primary key (alunomatricula)
8 )
```



- Exemplo Dados

alunomatrícula	alunonome	alunoemail	alunoidade
A0001	Josefina Fininha	jf@universo.com	20
A0002	Bertolldo Grosso	bt@universo.com	25
A0003	Fulano de tal	jft@universo.com	25



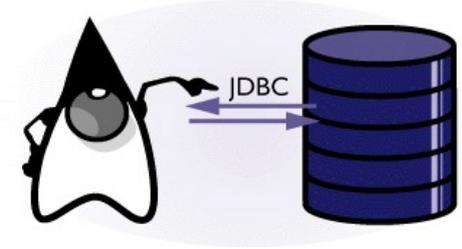
Recuperando dados do BD

A recuperação de dados do BD é trivial e é feita através da execução de uma *query* SQL (consulta sql), e os dados podem ser recuperados através da interface ResultSet, retornada na execução da query.

Explicação

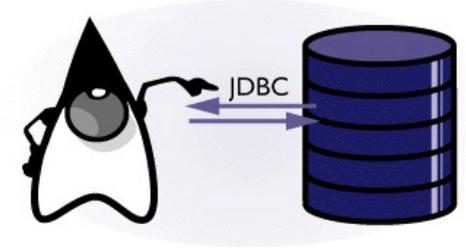


1. O método `executeQuery()` de `Statement` executa uma consulta (query) SQL e retorna um objeto `ResultSet`, que contém os dados recuperados.
2. Através do método `next()` de `ResultSet`, o cursor interno é movido para o próximo registro. O método retorna `false` caso não existam mais registros ou `true` caso contrário.
3. Os valores dos registros podem ser recuperados como o método `getString()`, que recebe o nome do campo ou seu *alias*.



Consultando dados

Para ver os dados de uma tabela, a tabela deve ser consultada. O comando `SELECT` do SQL é utilizado para esta função. O comando é dividido em lista de seleção (a parte que especifica as colunas a ser retornadas), lista de tabelas (a parte que especifica as tabelas de onde os dados vão ser extraídos), e uma qualificação opcional (a parte onde são especificadas as restrições).

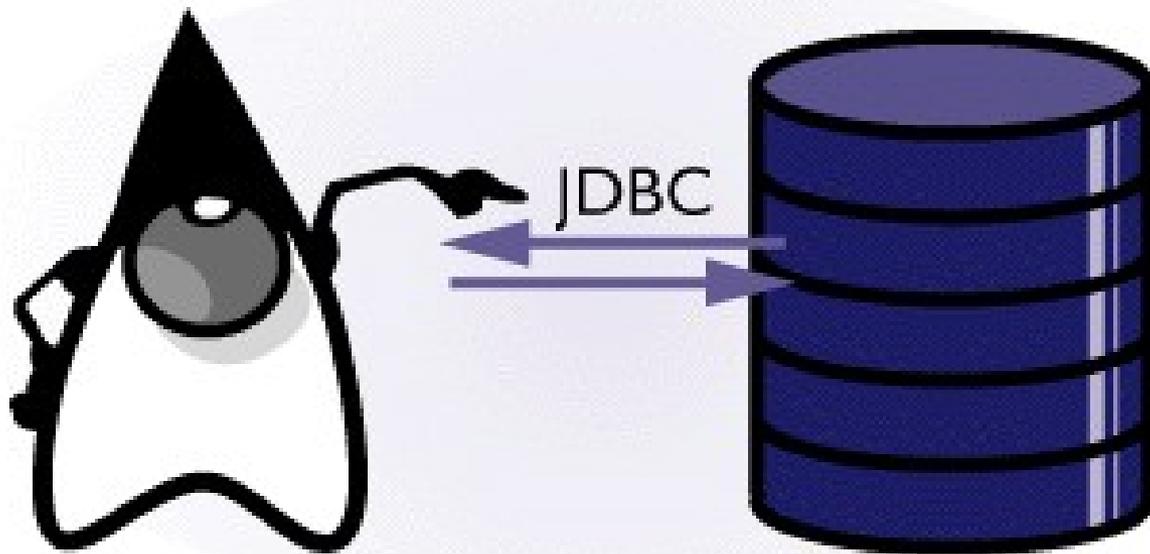


Consultando dados

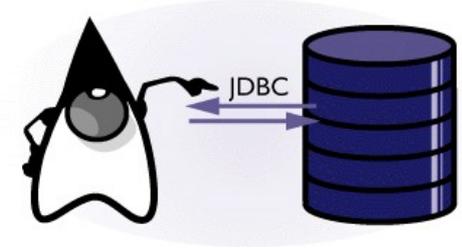
Exemplo do comando SQL

```
Select * from alunos
```

Exemplo de consulta de dados com JDBC



```
14 public class ConexaoBancoDeDados {
15
16     static String driverJDBC = "org.gjt.mm.mysql.Driver";
17     static String url = "jdbc:mysql://localhost:3306/reservas";
18     static String usuario = "root";
19     static String senha = "root";
20 public static void main(String[] args) {
21     Connection conexao = null;
22     Connection con = conexao;
23     Statement st = null;
24     ResultSet result = null;
25     String instrucaoSQL = "Select * from alunos";
26     try{
27         Class.forName(driverJDBC);
28         conexao = DriverManager.getConnection(url, usuario, senha);
29         System.out.println("Consultando os dados na tabela...");
30         st = conexao.createStatement();
31         result = st.executeQuery(instrucaoSQL);
32         while (result.next()){
33             System.out.println("+-----+");
34             System.out.println("Matricula.....: " + result.getString(1));
35             System.out.println("Nome.....: " + result.getString(2));
36             System.out.println("Email.....: " + result.getString(3));
37             System.out.println("Idade.....: " + result.getInt(4));
38         }
39         result.close();
40         st.close();
41         conexao.close();
42     }catch (Exception e){
43         System.out.println("Erro");
44         e.printStackTrace();
45     }
46 }
47 }
```



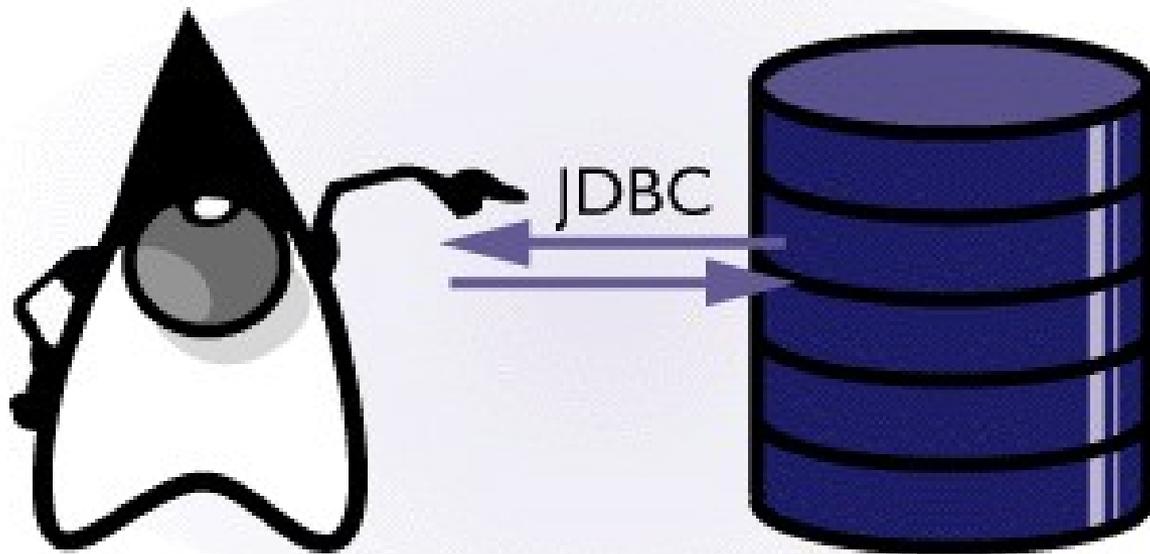
Alterando dados

As linhas existentes podem ser atualizadas utilizando o comando UPDATE.

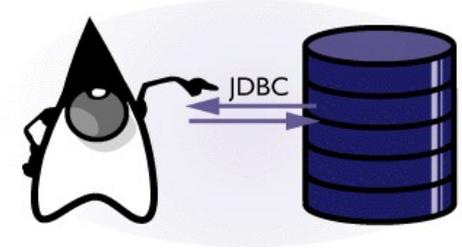
Exemplo do comando SQL

```
UPDATE alunos SET alunoidade = 30;
```

Exemplo de alteração de dados com JDBC



```
14 public class ConexaoBancoDeDados {
15
16     static String driverJDBC = "org.gjt.mm.mysql.Driver";
17     static String url = "jdbc:mysql://localhost:3306/reservas";
18     static String usuario = "root";
19     static String senha = "root";
20 public static void main(String[] args) {
21     Connection conexao = null;
22     Connection con = conexao;
23     Statement st = null;
24     ResultSet result = null;
25     String instrucaoSQL = "update alunos set alunoidade = 25";
26     try{
27         Class.forName(driverJDBC);
28         conexao = DriverManager.getConnection(url, usuario, senha);
29         System.out.println("Atualizando os dados na tabela...");
30         st = conexao.createStatement();
31         st.executeUpdate(instrucaoSQL);
32         System.out.println("Dados atualizados na tabela.");
33         st.close();
34         conexao.close();
35     } catch (Exception e){
36         System.out.println("Erro");
37         e.printStackTrace();
38     }
39 }
40 }
```



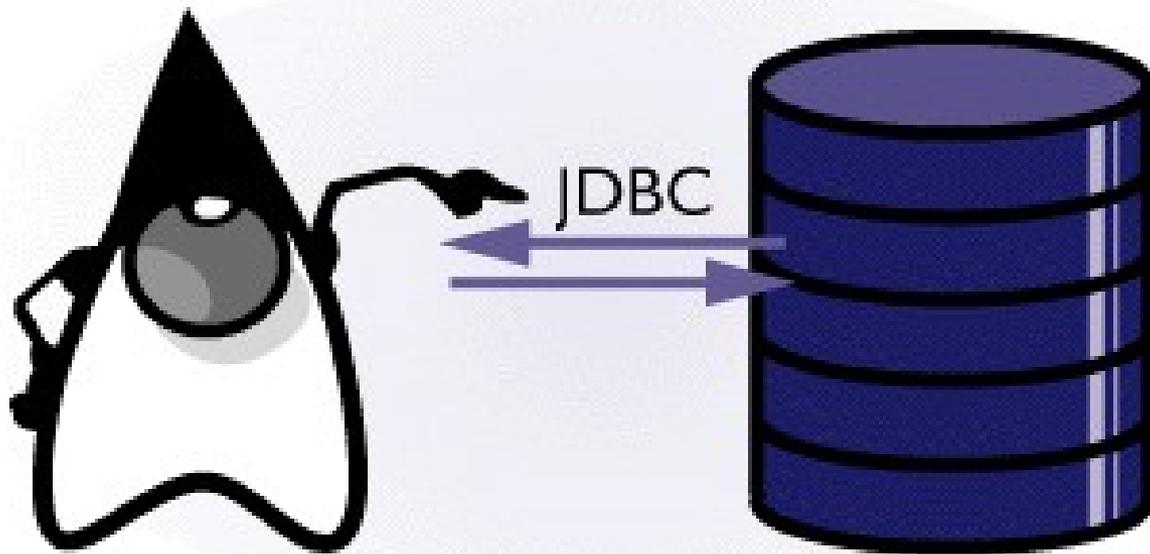
Excluindo dados

As exclusões são realizadas utilizando o comando DELETE.

Exemplo do comando SQL

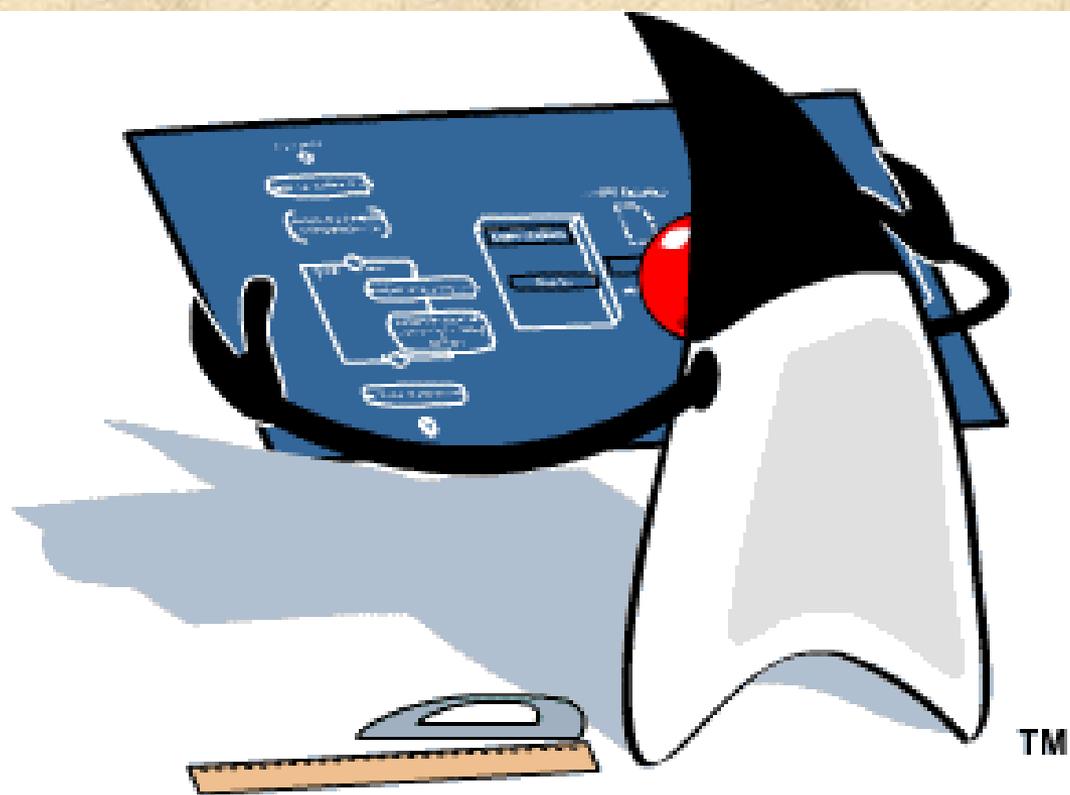
```
DELETE FROM alunos
```

Exemplo de exclusão de dados com JDBC



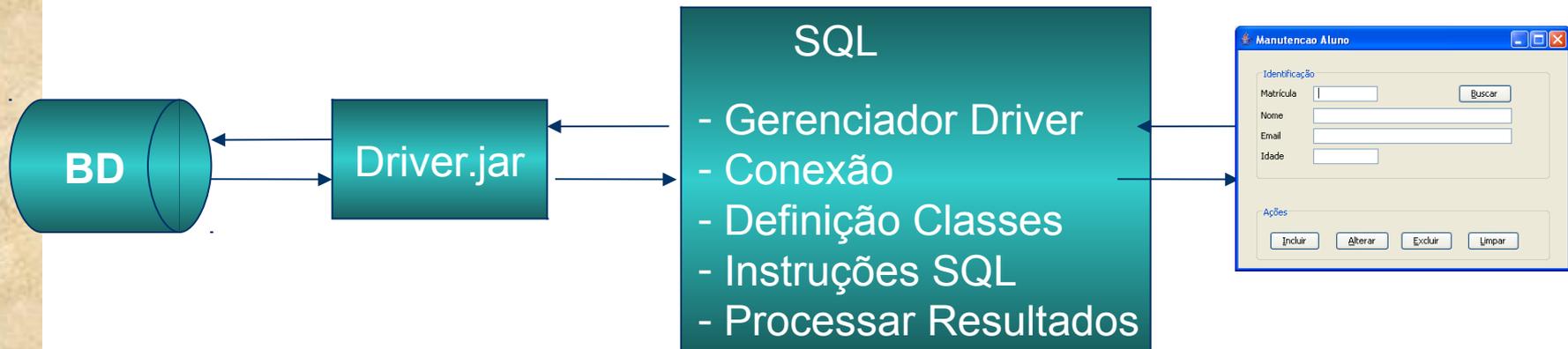
```
14 public class ConexaoBancoDeDados {
15
16     static String driverJDBC = "org.gjt.mm.mysql.Driver";
17     static String url = "jdbc:mysql://localhost:3306/reservas";
18     static String usuario = "root";
19     static String senha = "root";
20 public static void main(String[] args) {
21     Connection conexao = null;
22     Connection con = conexao;
23     Statement st = null;
24     ResultSet result = null;
25     String instrucaoSQL = "DELETE FROM alunos";
26     try{
27         Class.forName(driverJDBC);
28         conexao = DriverManager.getConnection(url, usuario, senha);
29         System.out.println("Excluindo os dados na tabela...");
30         st = conexao.createStatement();
31         st.executeUpdate(instrucaoSQL);
32         System.out.println("Dados excluidos na tabela.");
33         st.close();
34         conexao.close();
35     }catch (Exception e){
36         System.out.println("Erro");
37         e.printStackTrace();
38     }
39 }
40 }
```

Formulários



TM

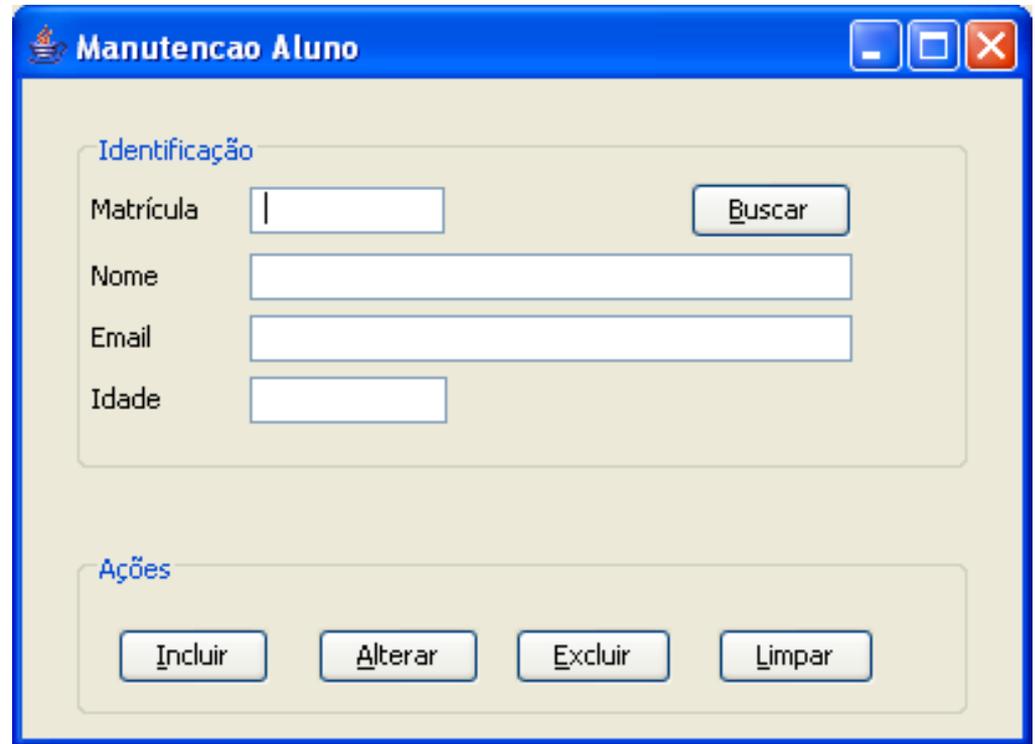
Formulários



Formulários

SQL

- Gerenciador Driver
- Conexão
- Definição Classes
- Instruções SQL
- Processar Resultados



The image shows a screenshot of a Windows application window titled "Manutencao Aluno". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The main content area is light gray and contains two sections:

- Identificação:** This section contains four input fields and one button:
 - Matricula:** A text input field with a vertical cursor, followed by a "Buscar" button.
 - Nome:** A wide text input field.
 - Email:** A wide text input field.
 - Idade:** A text input field.
- Ações:** This section contains four buttons arranged horizontally:
 - Incluir:** A button with the text "Incluir".
 - Alterar:** A button with the text "Alterar".
 - Excluir:** A button with the text "Excluir".
 - Limpar:** A button with the text "Limpar".

Bibliografia

Site Oficial do Java

<http://java.sun.com>

Tutorial Oficial da JDBC

<http://java.sun.com/docs/books/tutorial/jdbc/index.html>

Documentação da API da JDBC

<http://java.sun.com/j2se/1.4.2/docs/api/java/sql/package-summary.html>

Portal Unicamp

<http://www.dca.fee.unicamp.br/cursos/PooJava/javadb/jdbc.html>