Unidade 12: Programação de Banco de Dados com Java

Prof. Daniel Caetano

Objetivo: Construir uma aplicação Java que interaja com Banco de Dados

INTRODUÇÃO

Nas aulas anteriores vimos um aspecto fundamental em quase todas as aplicações comerciais: a interface gráfica. Agora veremos o outro aspecto presente em 99 entre 100 aplicações comerciais: o banco de dados.

Não é objetivo deste curso apresentar detalhes sobre banco de dados, visto que há disciplinas específicas para isto. O objetivo é apresentar como integrar o Java com um Banco de Dados como o MS SQL e o MySQL, e apresentar estes resultados em um componente visual.

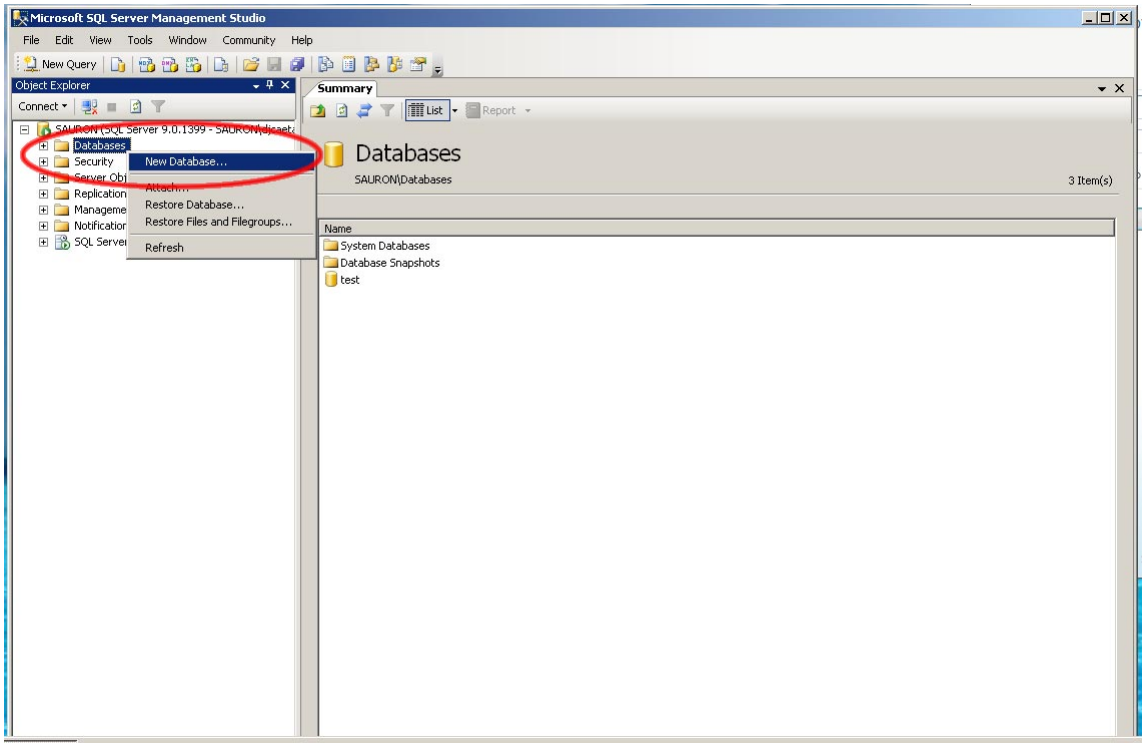
Esta aula será apresentada no formato de tutorial, apresentando primeiramente os passos para a criação de um banco de dados simples no MS SQL Server e, posteriormente, como ler estes dados usando um programa em Java.

1. CONSTRUINDO UM BANCO DE DADOS COM O MS SQL SERVER

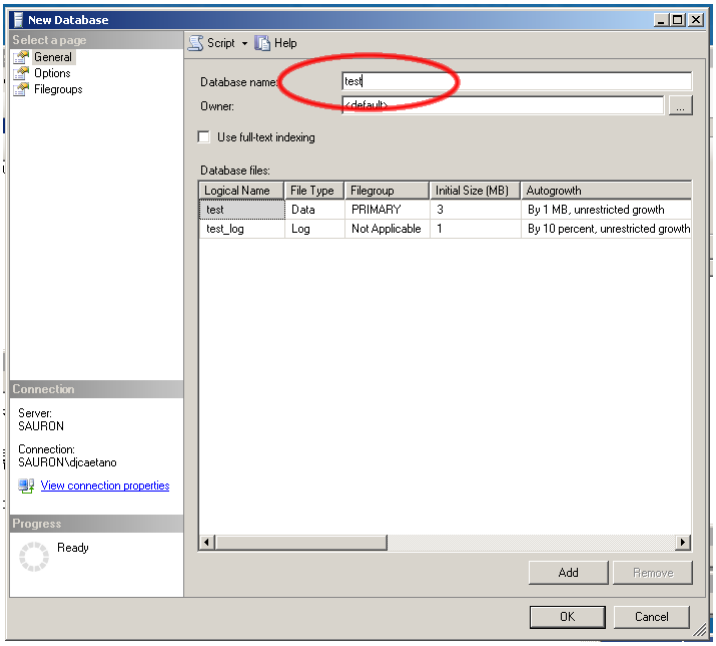
PASSO 1: Abra o aplicativo **SQL Server Management Studio**, disponível em ***Iniciar > Todos os Programas > Microsoft SQL Server > SQL Server Management Studio***. Se tudo estiver corretamente configurado, uma janela similar a apresentada abaixo surgirá. Clique no botão "Conectar".

Caso o sistema exija um password, consulte o administrador do sistema ou o professor.

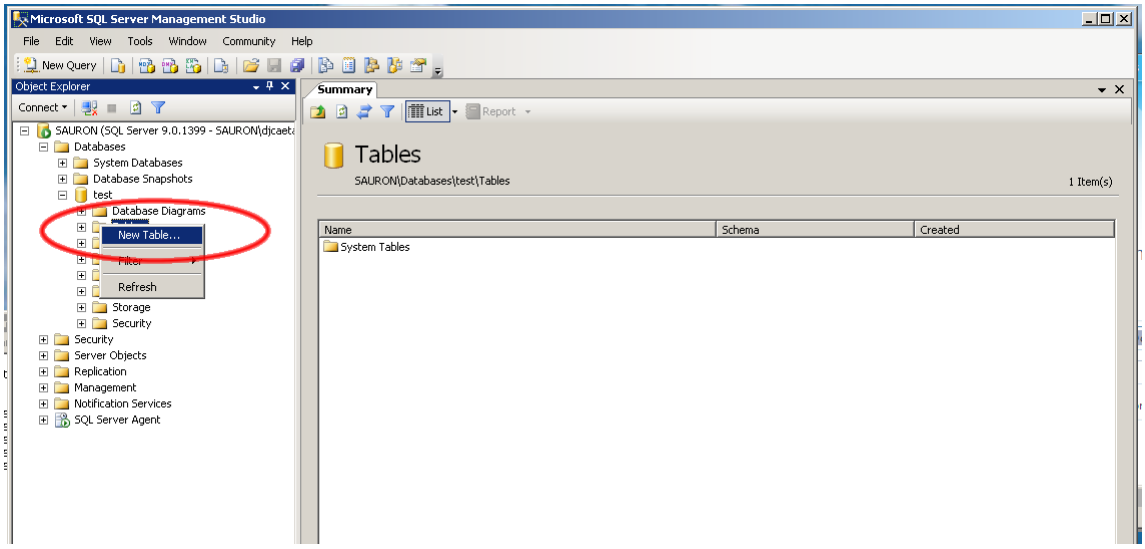
PASSO 2: Uma vez conectado, o primeiro passo é criar um banco de dados. Para isso, clique com o botão direito na opção **Databases** e selecione a opção **New Database...** , conforme indicado abaixo.



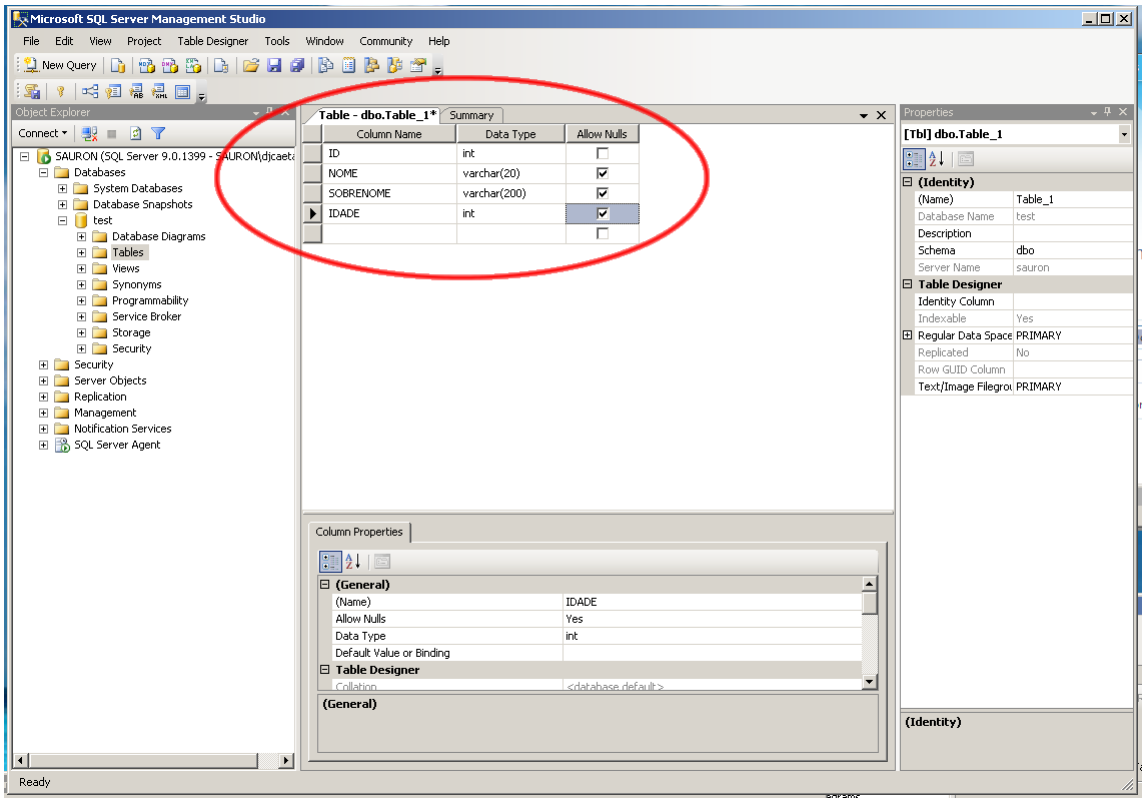
PASSO 3: Na janela que aparece em seguida, dê o nome de **test** para o banco de dados, conforme indicado na figura a seguir. Depois disso, clique no botão **OK**.



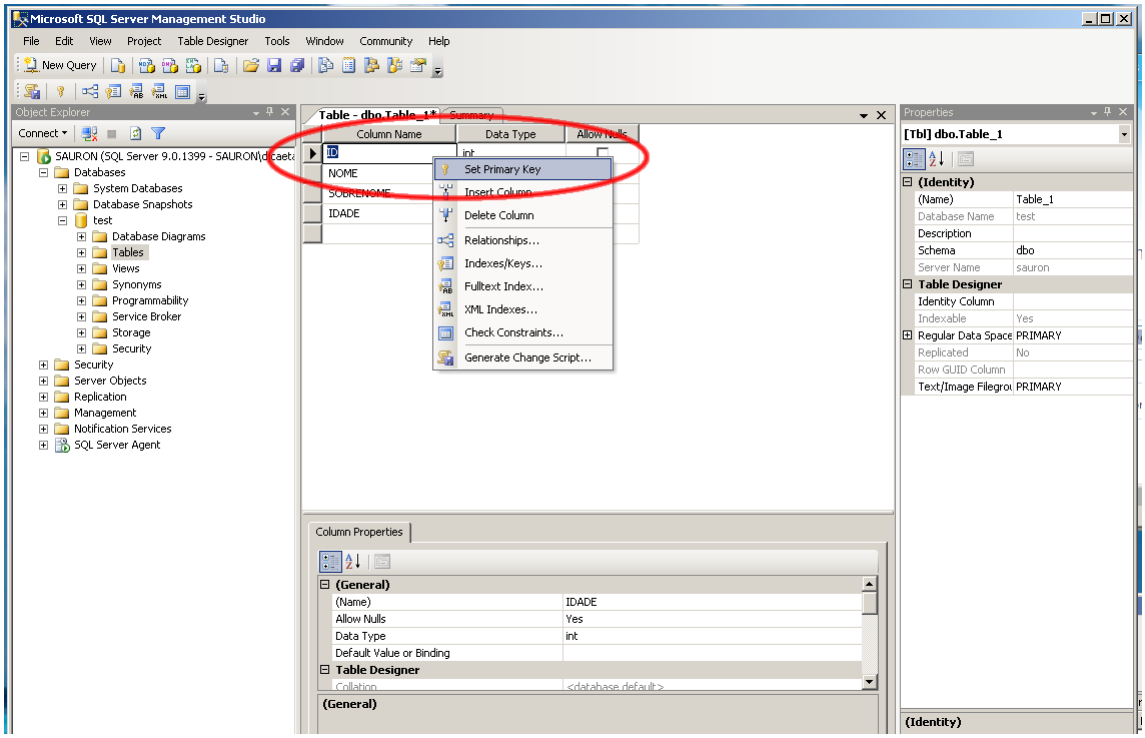
PASSO 4: O próximo passo é a criação de uma tabela (ou relação) neste banco de dados. São as tabelas que armazenam os dados em um banco de dados. Para criar uma nova tabela, clique com o botão direito na opção **Databases > test > Tables** e selecione a opção **New Table...** , como indicado abaixo.



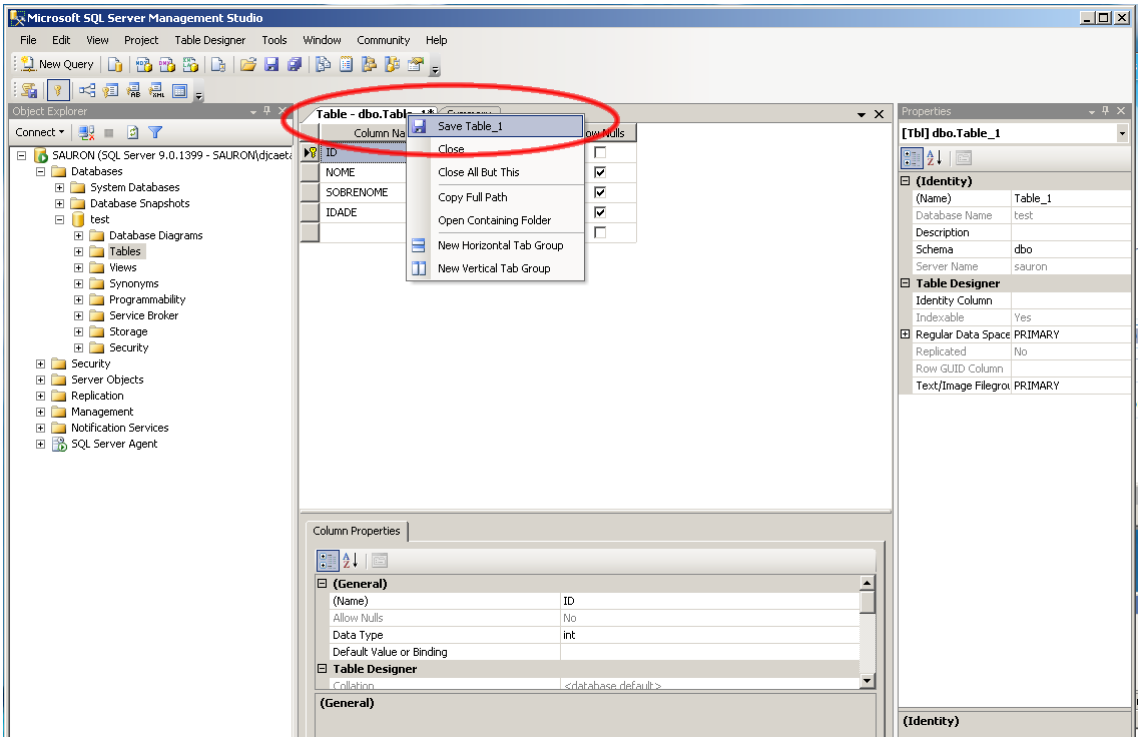
PASSO 5: Vamos agora definir quais dados irão existir em cada uma das colunas da tabela. Configure como indicado na figura abaixo. Serão QUATRO colunas: ID (int, sem Allow Nulls), NOME (varchar(20) com Allow Nulls), SOBRENOME (varchar(200) com Allow Nulls) e IDADE (int com Allow Nulls).



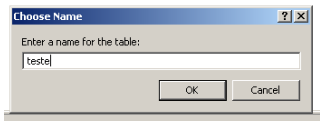
PASSO 6: Vamos agora definir a chave primária de nossa tabela. Clique com o botão direito na linha da coluna chamada ID, e selecione a opção **Set Primary Key**, como indicado abaixo. Uma "chavinha" deverá aparecer ao lado do nome da coluna ID.



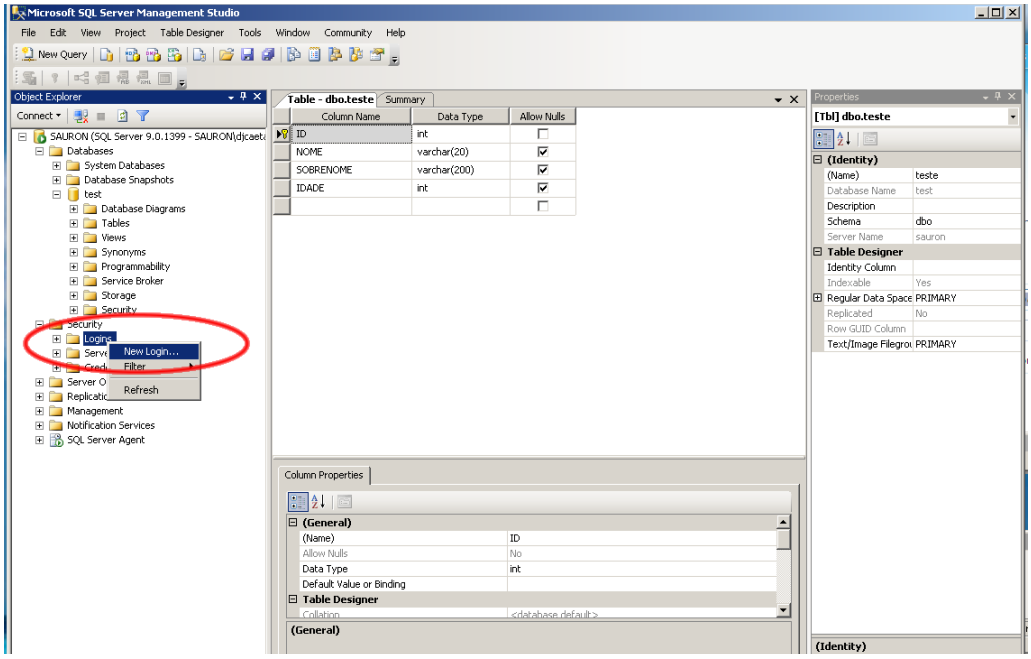
PASSO 7: Agora é o momento de gravarmos a nossa tabela. Clique com o botão direiro do mouse na aba **Table - dbo.Table_1*** e selecione a opção **Save Table_1** conforme indicado na figura abaixo.



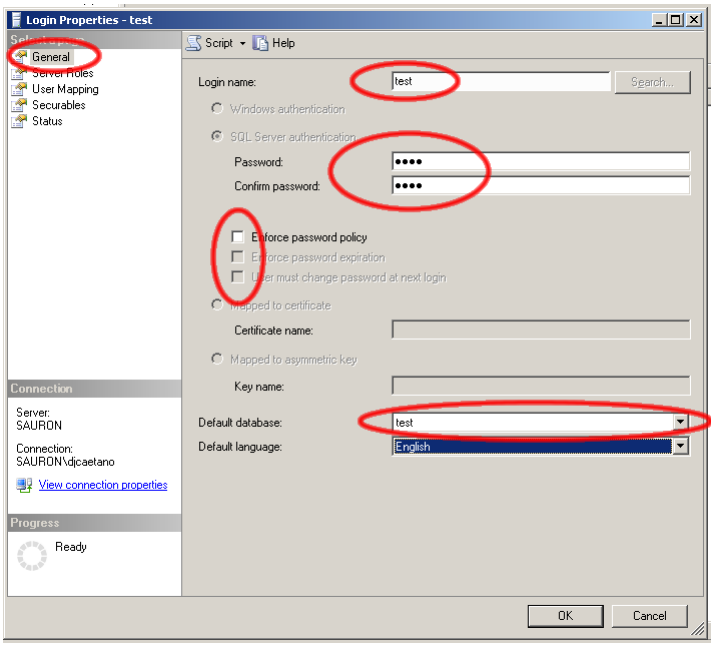
PASSO 8: Na janela que irá aparecer, dê o nome de **teste** para a tabela, conforme indicado na figura a seguir.



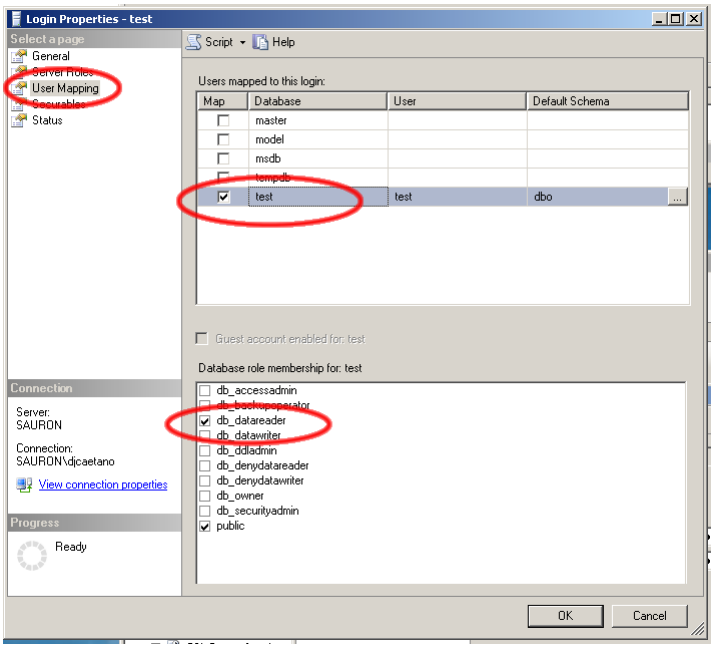
PASSO 9: Agora chegou o momento de criar um usuário para a nossa aplicação, ou seja, um usuário que seja capaz de ler a tabela que já criamos. Clique com o botão direito na opção **Security > Logins** e selecione a opção **New Login...** , como indicado abaixo.



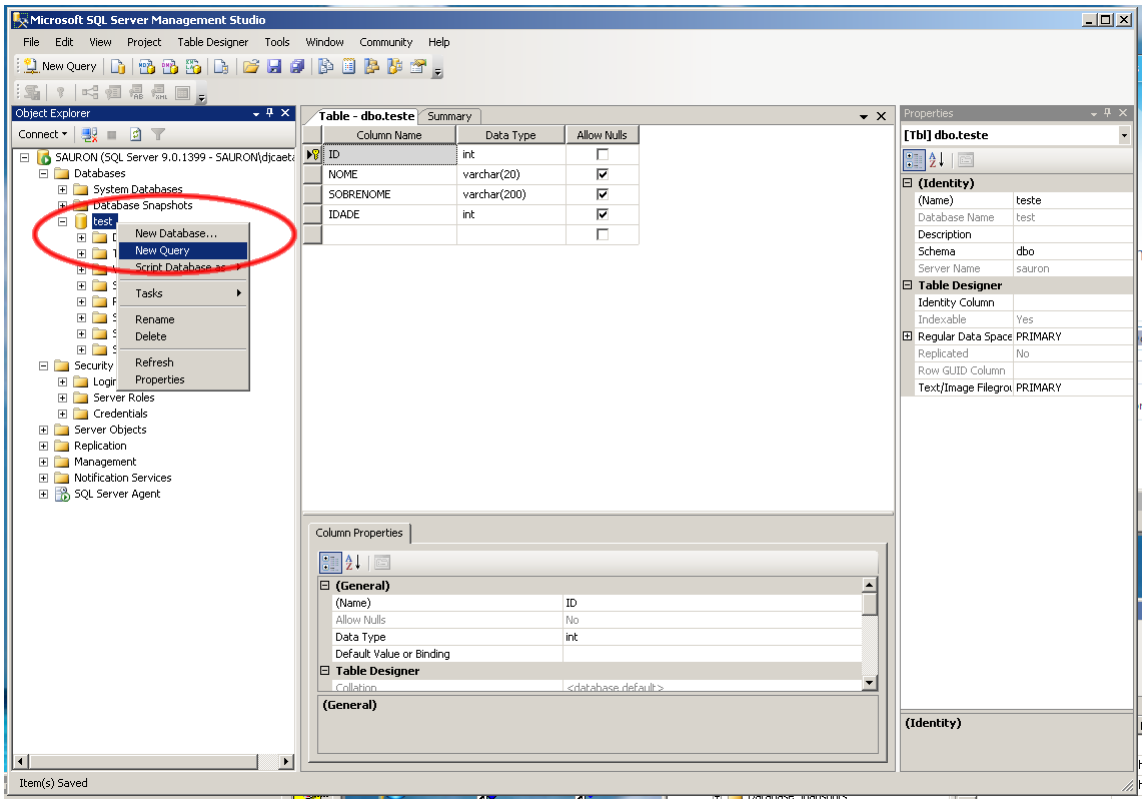
PASSO 10: Na primeira janela que irá aparecer, configuraremos o login do usuário. Dê o nome **test**, configure as duas senhas como **test**, desligue a opção "**Enforce password policy**" e selecione como Default Database o banco de dados **test** que criamos.



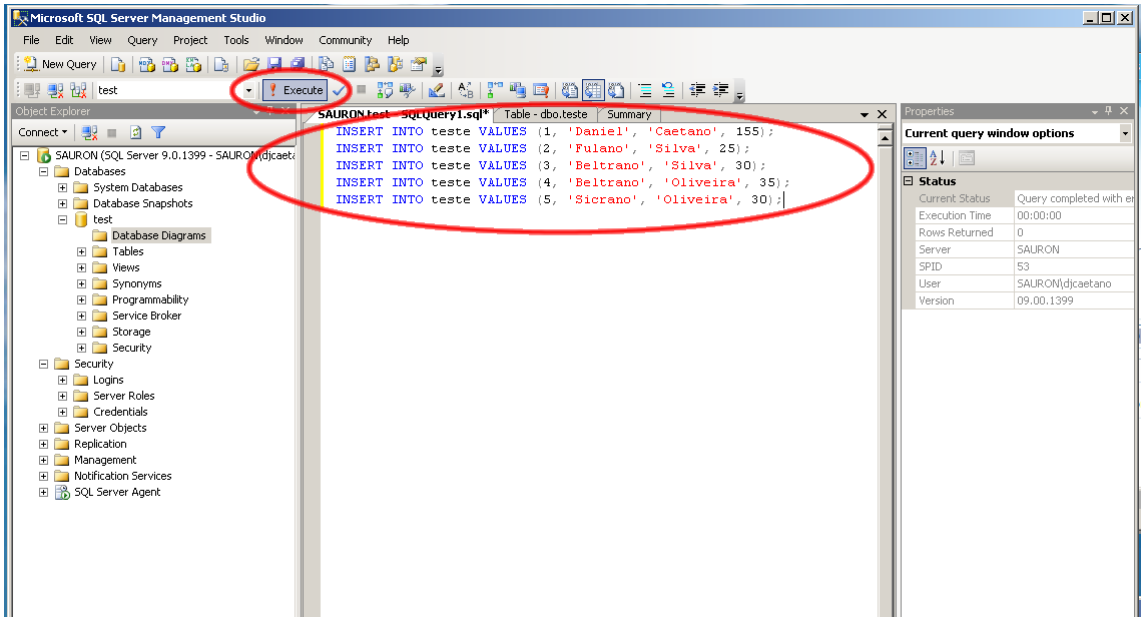
PASSO 11: Selecione agora a opção **User Mapping**, marque o database **test** e, como Database role membership, marque **db_datareader** (e deixe marcado o public). Observe a indicação na janela abaixo.



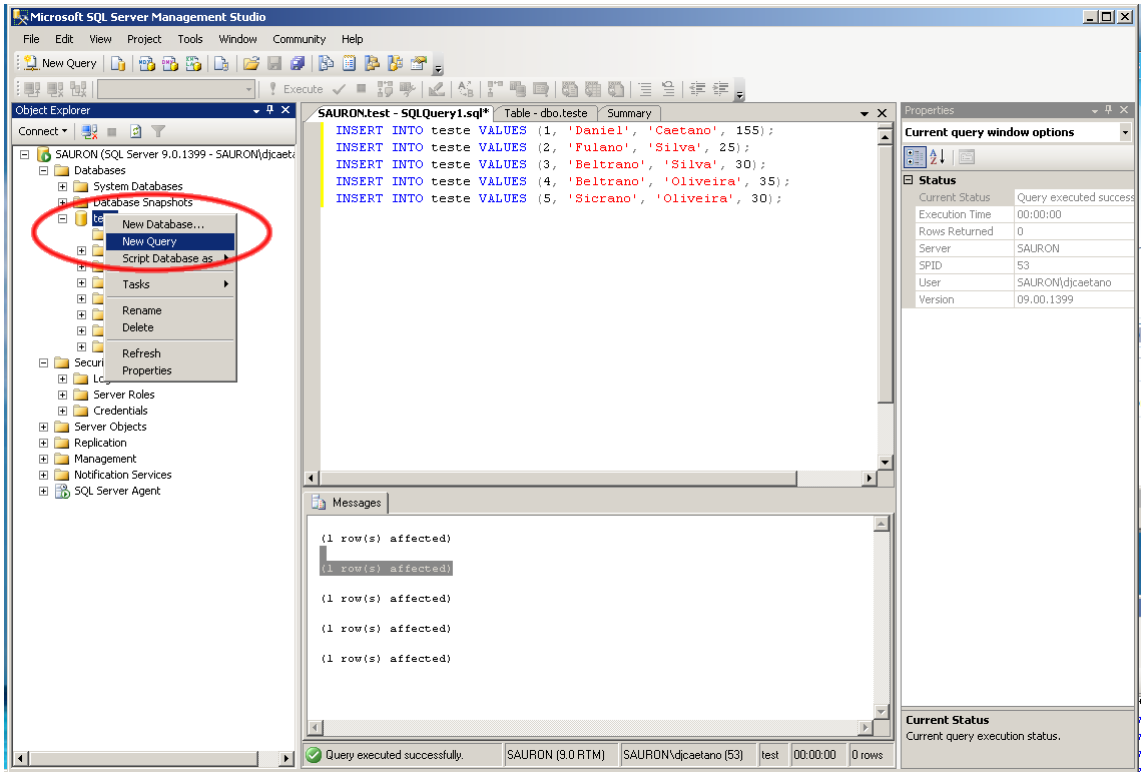
PASSO 12: Agora iremos inserir alguns dados em nossa tabela, para que possamos consultá-los futuramente, pelo programa em Java. Para isso, precisamos criar uma query (busca). Clique com o botão direito no item **Databases > test** e selecione **New Query**.



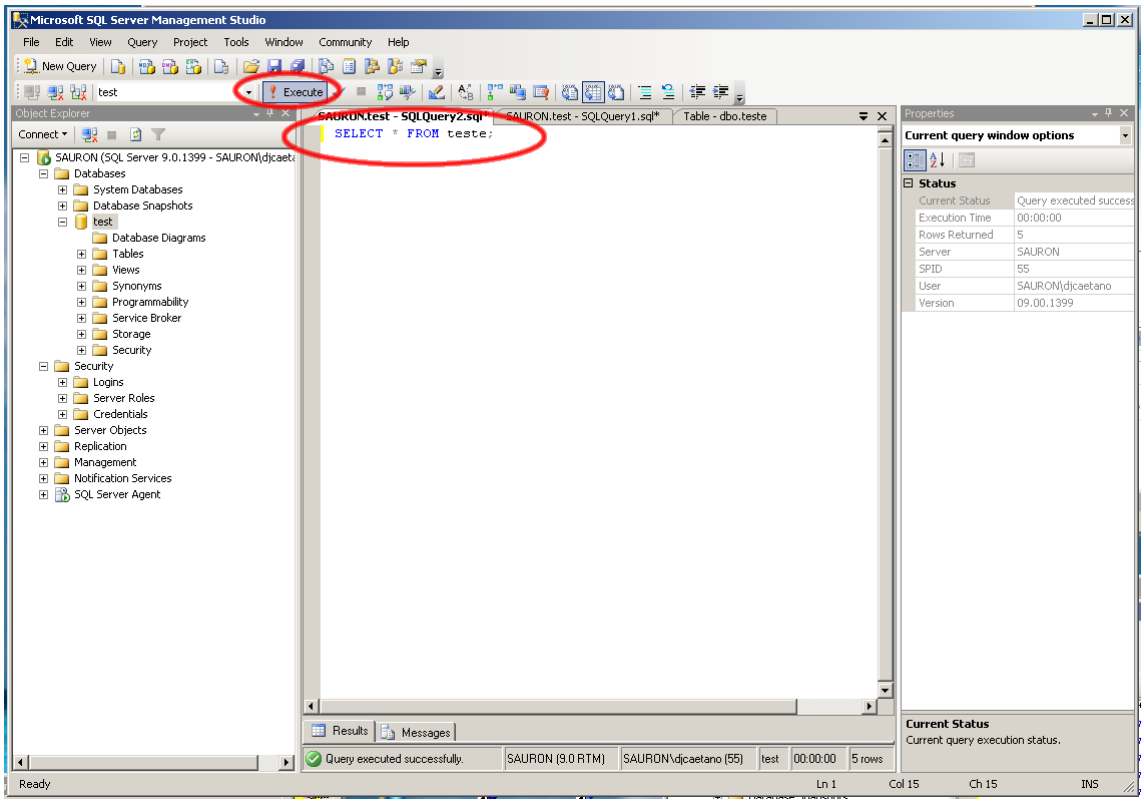
PASSO 13: Na nova tela, iremos inserir 5 linhas em nossa tabela, usando a instrução INSERT. Digite-as como indicado na figura abaixo e, depois, clique na opção **Execute**, conforme indicado.



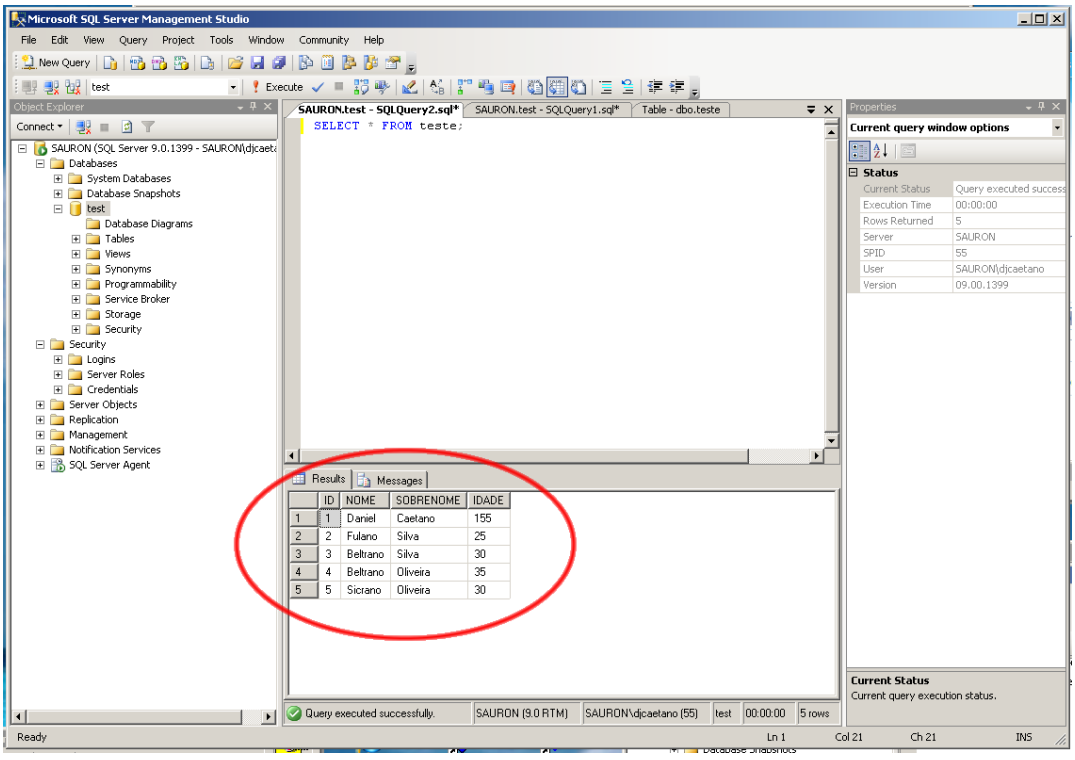
PASSO 14: Se tudo foi feito corretamente e nenhum erro ocorreu, podemos testar a verificar se os dados foram inseridos com sucesso. Para isso, criaremos uma nova query. Clique com o botão direito no item **Databases > test** e selecione **New Query**.



PASSO 15: Na nova tela, iremos realizar uma busca, usando a instrução SELECT. Digite-as como indicado na figura abaixo e clique na opção **Execute**, conforme indicado.



PASSO 16: Como resultado, a janela do SQL Server deve mostrar os dados conforme a figura abaixo.



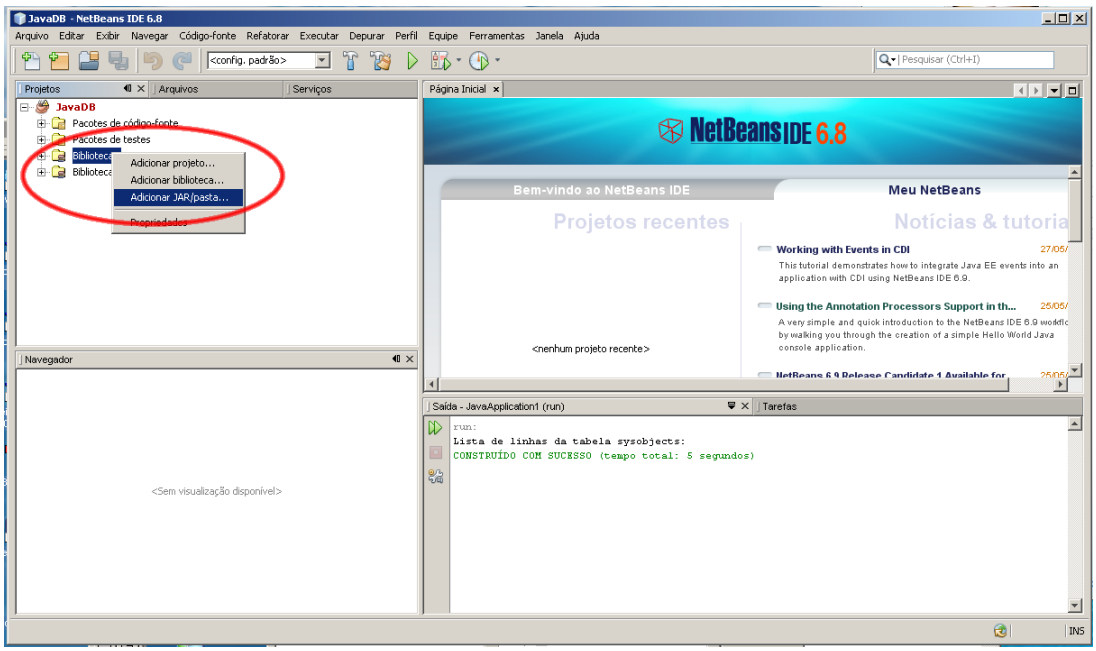
2. CONSTRUINDO UMA APLICAÇÃO JAVA PARA BANCO DE DADOS

PASSO 1: Crie um projeto ***Java*** normal chamado **JanelaDB** no NetBeans.

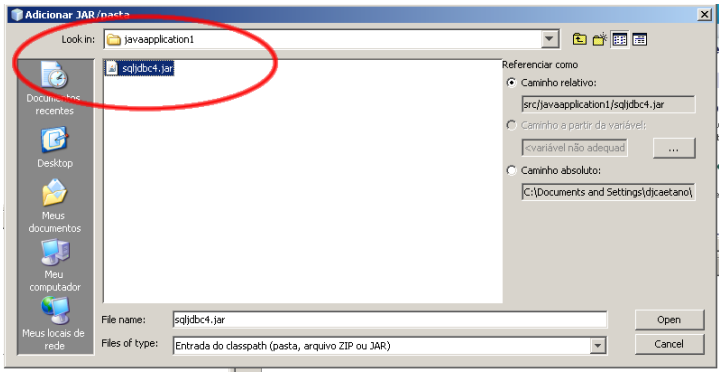
PASSO 2: Como iremos conectar no banco de dados MS SQL Server, precisamos acrescentar em nosso projeto a biblioteca de acesso ao MS SQL Server em nosso ambiente de desenvolvimento. Para isso, primeiramente copie o arquivo **sqljdbcb4.jar**, fornecido pelo professor, para o diretório do seu projeto (provavelmente em **Meus Documentos > My NetBeans Projects > JavaDB**).

> Nota: caso o Java que você esteja usando seja anterior ao Java 1.6, você deverá usar o arquivo **sqljdbc.jar**, ao invés do sqljdbc4.jar.

PASSO 3: Agora iremos associar este arquivo ao projeto. Para isso, na área de projeto, clique com o botão direito no item **JavaDB > Bibliotecas**, e selecione a opção **Adicionar JAR/pasta...** no menu. Observe na figura abaixo.



PASSO 4: E selecione o arquivo **sqljdbcb4.jar**, fornecido pelo professor e colocado no diretório adequado, conforme indicado na figura.



PASSO 5: Agora, clique com o botão direito no pacote **janeladb** e selecione **Novo > Classe Java**. Dê o nome de **JMain** para esta classe.

PASSO 6: Comecemos criando o código para a classe Main. Na área de projeto, dê um duplo clique no arquivo **Main.java**, que deve estar no pacote janeladb. Isso irá apresentar o código fonte atual na janela da direita. Além dos comentários, o código deve ter os seguintes elementos:

Main.java

```
package janeladb;
public class Main {
   public static void main(String[] args) {
      }
   }
```

PASSO 7: O código que iremos acrescentar na classe Main simplesmente irá criar uma janela do tipo JMain, que ainda iremos programar. Modifique o arquivo **Main.java** para que fique como indicado a seguir.

Main.java

```
package janeladb;
public class Main {
   public static void main(String[] args) {
      Jmain janela = new JMain();
      }
   }
```

PASSO 8: O próximo passo será simplesmente a criação da janela JFrame, com um elemento JTable interno. Para isso, vamos editar o arquivo JMain, clicando duas vezes no nome de arquivo **JMain.java**, na área de projeto. O código dele deve ser o indicado abaixo:

JMain.java

```
package janeladb;
public class JMain {
   }
```

PASSO 9: Vamos começar inserindo os *imports* básicos, conforme indicado abaixo.

JMain.java

```
package janeladb;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class JMain {
   }
```

PASSO 10: A próxima coisa que precisa ser feita é a definição que a nossa nova classe estende a classe JFrame, para que possamos configurá-la como a janela de nossa aplicação e, assim, definir seus elementos visuais. Para dizer que **JMain estende JFrame**, faça a indicação abaixo.

JMain.java

```
package janeladb;
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class JMain extends JFrame {
   }
```

PASSO 11: O NetBeans provavelmente vai reclamar, pois agora precisaremos fazer um construtor (método com o mesmo nome da classe) para a nossa janela. O construtor pode ser programado como indicado abaixo: ele cria apenas configura a janela e cria um elemento JTable dentro.

JMain.java

```
package janeladb;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class JMain extends JFrame {
   public JMain() {
      super("Consultando SQL!"); // Cria a janela com este título
      Conteiner painel = getContentPane(); // Pega a área de cliente da janela
      painel.setLayout (new FlowLayout()); // Configura area de cliente como flow
      Jtable result = new JTable(6,4); // Cria tabela result com 6 linhas e 4 colunas
      painel.add(result); // Acrescenta tabela na janela

      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // Fecha progr. no botão fechar
      setSize(400,150); // Define tamanho da janela
      setVisible(true); // torna a janela visível
      }
   }
```

PASSO 12: Vamos criar um método para consultar o banco de dados e preencher a tabela. Esse método será chamado no construtor e terá a seguinte assinatura:

public void readData(JTable tabela)

Note que ele deve receber a tabela a ser preenchida como um parâmetro. Observe a implementação básica a seguir.

JMain.java

```
package janeladb;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class JMain extends JFrame {
   public JMain() {
      super("Consultando SQL!"); // Cria a janela com este título
      Conteiner painel = getContentPane(); // Pega a área de cliente da janela
      painel.setLayout (new FlowLayout()); // Configura area de cliente como flow
      Jtable result = new JTable(6,4); // Cria tabela result com 6 linhas e 4 colunas
      painel.add(result); // Acrescenta tabela na janela
      readData(result);
      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // Fecha progr. no botão fechar
      setSize(400,150); // Define tamanho da janela
      setVisible(true); // torna a janela visível
      }
   public void readData(JTable tabela) {
      }
   }
```

PASSO 13: Vamos agora acrescentar o código que conecta ao banco de dados.

JMain.java

```
package janeladb;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class JMain extends JFrame {
   public JMain() {
      super("Consultando SQL!"); // Cria a janela com este título
      Conteiner painel = getContentPane(); // Pega a área de cliente da janela
      painel.setLayout (new FlowLayout()); // Configura area de cliente como flow
      Jtable result = new JTable(6,4); // Cria tabela result com 6 linhas e 4 colunas
      painel.add(result); // Acrescenta tabela na janela
      readData(result);
      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // Fecha progr. no botão fechar
      setSize(400,150); // Define tamanho da janela
      setVisible(true); // torna a janela visível
      }
   public void readData(JTable tabela) {
      try {
          // Driver SQL instalado?
          Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        }
      catch(java.lang.ClassNotFoundException e) {
         // Se não achou, mostra erro
         System.err.println("Classe não encontrada: " + e.getMessage());
         }
      try {
         // Define endereço da conexão
         String endereco = "jdbc:sqlserver://localhost:1433;" +
               "databaseName=test;user=test;password=test;";
         // Cria conexação e um objeto de transacao
         Connection conexao = DriverManager.getConnection(endereco);
         Statement transacao = conexao.createStatement();

         // Aqui virá o código de consulta

         // Fim da Conexão
         transacao.close();
         conexao.close();
         }
      catch(SQLException ex) {
         System.err.println("Excessão no SQL: " + ex.getMessage());
         }
      }
   }
```

Neste caso, há dois blocos *try-catch*: o primeiro testa se o driver necessário para a conexão com o banco de dados está instalado; o segundo faz a conexão propriamente dita. Dentro do bloco da conexão, antes de fechá-la, é que serão realizadas as consultas e as atualizações da tabela.

PASSO 14: O próximo passo é realizar uma consulta no banco de dados. Iremos coletar todas as colunas de todas as linhas da tabela, usando a sintaxe

SELECT id, nome, sobrenome, idade FROM teste

O código que implementa esta consulta pode ser visto a seguir.

JMain.java

```
package janeladb;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class JMain extends JFrame {
   public JMain() {
      super("Consultando SQL!"); // Cria a janela com este título
      Conteiner painel = getContentPane(); // Pega a área de cliente da janela
      painel.setLayout (new FlowLayout()); // Configura area de cliente como flow
      Jtable result = new JTable(6,4); // Cria tabela result com 6 linhas e 4 colunas
      painel.add(result); // Acrescenta tabela na janela
      readData(result);
      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // Fecha progr. no botão fechar
      setSize(400,150); // Define tamanho da janela
      setVisible(true); // torna a janela visível
      }
   public void readData(JTable tabela) {
      try {
         // Driver SQL instalado?
         Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
         }
      catch(java.lang.ClassNotFoundException e) {
         // Se não achou, mostra erro
         System.err.println("Classe não encontrada: " + e.getMessage());
         }
      try {
         // Define endereço da conexão
         String endereco = "jdbc:sqlserver://localhost:1433;" +
               "databaseName=test;user=test;password=test;";
         // Cria conexação e um objeto de transacao
         Connection conexao = DriverManager.getConnection(endereco);
         Statement transacao = conexao.createStatement();

         // Define a busca
         String query = "SELECT id, nome, sobrenome, idade FROM teste";
         // Executa busca e pega resultado
         ResultSet resultado = transacao.executeQuery(query);
         // Colhe informações sobre resultado (nome das colunas, número de linhas etc)
         ResultSetMetaData infoResultado = resultado.getMetaData();

         // Fim da Conexão
         transacao.close();
         conexao.close();
         }
      catch(SQLException ex) {
         System.err.println("Excessão no SQL: " + ex.getMessage());
         }
      }
   }
```

PASSO 15: Agora iremos colocar a informação dos nomes das colunas na primeira linha da tabela, conforme indicado a seguir

JMain.java

```
package janeladb;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class JMain extends JFrame {
   public JMain() {
      super("Consultando SQL!"); // Cria a janela com este título
      Conteiner painel = getContentPane(); // Pega a área de cliente da janela
      painel.setLayout (new FlowLayout()); // Configura area de cliente como flow
      Jtable result = new JTable(6,4); // Cria tabela result com 6 linhas e 4 colunas
      painel.add(result); // Acrescenta tabela na janela
      readData(result);
      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // Fecha progr. no botão fechar
      setSize(400,150); // Define tamanho da janela
```

```
      setVisible(true); // torna a janela visível
      }
   public void readData(JTable tabela) {
      try {
         // Driver SQL instalado?
         Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
         }
      catch(java.lang.ClassNotFoundException e) {
         // Se não achou, mostra erro
         System.err.println("Classe não encontrada: " + e.getMessage());
         }
      try {
         // Define endereço da conexão
         String endereco = "jdbc:sqlserver://localhost:1433;" +
               "databaseName=test;user=test;password=test;";
         // Cria conexação e um objeto de transacao
         Connection conexao = DriverManager.getConnection(endereco);
         Statement transacao = conexao.createStatement();

         // Define a busca
         String query = "SELECT id, nome, sobrenome, idade FROM teste";
         // Executa busca e pega resultado
         ResultSet resultado = transacao.executeQuery(query);
         // Colhe informações sobre resultado (nome das colunas, número de linhas etc)
         ResultSetMetaData infoResultado = resultado.getMetaData();

         /* Imprimindo nomes das colunas em cada coluna */
         // Imprime nome da coluna 1 do resultado na linha 0 / coluna 0 da tabela
         tabela.setValueAt(infoResultado(1),0,0);
         // Imprime nome da coluna 2 do resultado na linha 0 / coluna 1 da tabela
         tabela.setValueAt(infoResultado(2),0,1);
         // Imprime nome da coluna 3 do resultado na linha 0 / coluna 2 da tabela
         tabela.setValueAt(infoResultado(3),0,2);
         // Imprime nome da coluna 4 do resultado na linha 0 / coluna 3 da tabela
         tabela.setValueAt(infoResultado(4),0,3);

         // Fim da Conexão
         transacao.close();
         conexao.close();
         }
      catch(SQLException ex) {
         System.err.println("Excessão no SQL: " + ex.getMessage());
         }
      }
   }
```

PASSO 16: Falta apenas o último passo: acrescentar as linhas de informação, colhidas no banco de dados, na tabela, conforme indicado no código a seguir.

JMain.java

```
package janeladb;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class JMain extends JFrame {
   public JMain() {
      super("Consultando SQL!"); // Cria a janela com este título
      Conteiner painel = getContentPane(); // Pega a área de cliente da janela
      painel.setLayout (new FlowLayout()); // Configura area de cliente como flow
      Jtable result = new JTable(6,4); // Cria tabela result com 6 linhas e 4 colunas
      painel.add(result); // Acrescenta tabela na janela
      readData(result);
      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // Fecha progr. no botão fechar
      setSize(400,150); // Define tamanho da janela
      setVisible(true); // torna a janela visível
      }
   public void readData(JTable tabela) {
      try {
         // Driver SQL instalado?
         Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
         }
      catch(java.lang.ClassNotFoundException e) {
```

```
         // Se não achou, mostra erro
         System.err.println("Classe não encontrada: " + e.getMessage());
         }
      try {
         // Define endereço da conexão
         String endereco = "jdbc:sqlserver://localhost:1433;" +
               "databaseName=test;user=test;password=test;";
         // Cria conexação e um objeto de transacao
         Connection conexao = DriverManager.getConnection(endereco);
         Statement transacao = conexao.createStatement();

         // Define a busca
         String query = "SELECT id, nome, sobrenome, idade FROM teste";
         // Executa busca e pega resultado
         ResultSet resultado = transacao.executeQuery(query);
         // Colhe informações sobre resultado (nome das colunas, número de linhas etc)
         ResultSetMetaData infoResultado = resultado.getMetaData();

         /* Imprimindo nomes das colunas em cada coluna */
         // Imprime nome da coluna 1 do resultado na linha 0 / coluna 0 da tabela
         tabela.setValueAt(infoResultado(1),0,0);
         // Imprime nome da coluna 2 do resultado na linha 0 / coluna 1 da tabela
         tabela.setValueAt(infoResultado(2),0,1);
         // Imprime nome da coluna 3 do resultado na linha 0 / coluna 2 da tabela
         tabela.setValueAt(infoResultado(3),0,2);
         // Imprime nome da coluna 4 do resultado na linha 0 / coluna 3 da tabela
         tabela.setValueAt(infoResultado(4),0,3);

         /* Loop que imprime dados na tabela */
         int linecount=1; // contador de linha sendo impressa
         while (resultado.next()) { // enquanto houver mais linhas...
            int i1 = resultado.getInt("id"); // pega coluna "id" como inteiro
            String s1 = resultado.getString("nome"); // pega coluna "nome" como string
            String s2 = resultado.getString("sobrenome"); // pega "sobrenome" como string
            int i2 = resultado.getInt("idade"); // pega coluna "idade" como inteiro

            // Coloca dados nas colunas certas da linha
            tabela.setValueAt(i1, linecount, 0);
            tabela.setValueAt(s1, linecount, 1);
            tabela.setValueAt(s2, linecount, 2);
            tabela.setValueAt(i2, linecount, 3);

            // Atualiza contador de linhas
            linecount = linecount + 1;
            }

         // Fim da Conexão
         transacao.close();
         conexao.close();
         }
      catch(SQLException ex) {
         System.err.println("Excessão no SQL: " + ex.getMessage());
         }
      }
   }
```

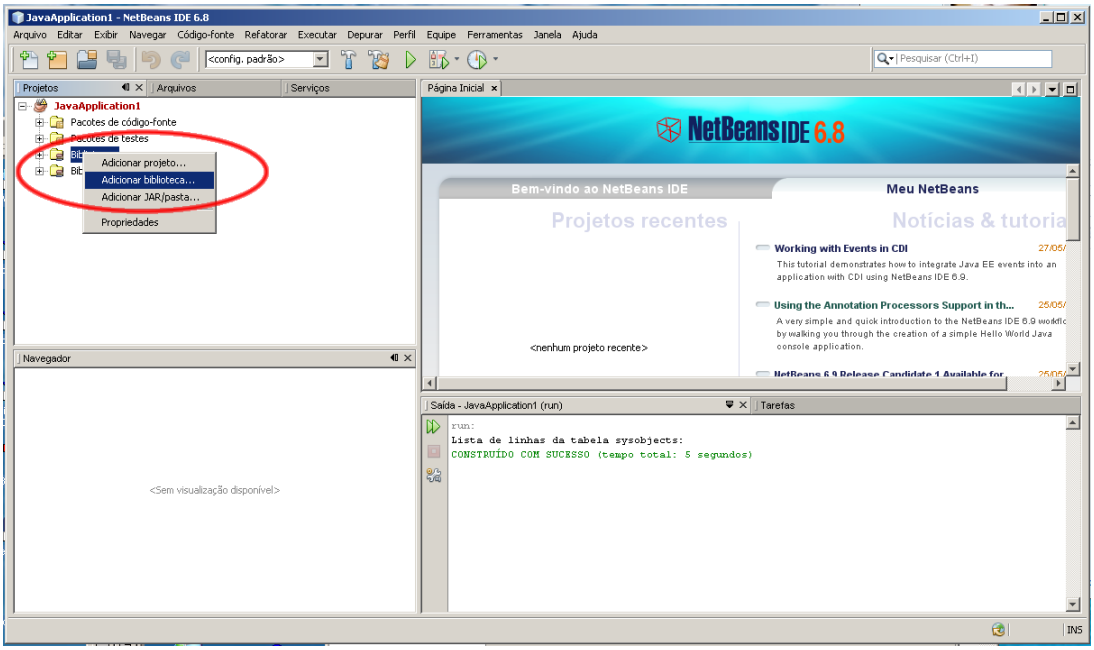
E, com isso, nosso programa está finalizado.

3. CONEXÃO COM O MYSQL

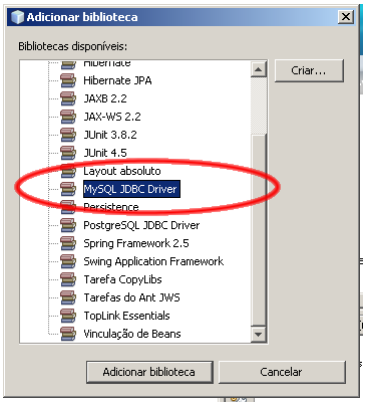
O MySQL é o banco de dados mais acessível, uma vez que é gratuito e seu download é de um tamanho bastante reduzido. Adicionalmente, o driver do MySQL já vem instalado no pacote Java SE + NetBeans, bastando para isso selecioná-lo no momento da criação do projeto. Os primeiros passos para a associação do MySQL ao projeto desenvolvido nesta aula são indicados a seguir.

PASSO 1: Crie um projeto ***Java*** normal chamado **JanelaDB** no NetBeans.

PASSO 2: Como iremos conectar no banco de dados MySQL, precisamos acrescentar em nosso projeto a biblioteca de acesso ao MySQL. Para isso, na área de projeto, clique com o botão direito no item **JavaDB > Bibliotecas**, e selecione a opção **Adicionar Biblioteca...** no menu. Observe na figura abaixo.



PASSO 3: Selecione, agora, o item **MySQL JDBC Driver**, conforme indicado na figura, clicando no botão "Adicionar Biblioteca" em seguida.



PASSO 4: Continue com os passos 5 em diante da seção 2. Ao final do último passo da seção 2, realize as mudanças identificadas no código que está indicado a seguir.

JMain.java

```
package janeladb;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.sql.*;

public class JMain extends JFrame {
   public JMain() {
      super("Consultando SQL!"); // Cria a janela com este título
      Conteiner painel = getContentPane(); // Pega a área de cliente da janela
      painel.setLayout (new FlowLayout()); // Configura area de cliente como flow
```

```
      Jtable result = new JTable(6,4); // Cria tabela result com 6 linhas e 4 colunas
      painel.add(result); // Acrescenta tabela na janela
      readData(result);
      setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // Fecha progr. no botão fechar
      setSize(400,150); // Define tamanho da janela
      setVisible(true); // torna a janela visível
      }
   public void readData(JTable tabela) {
      try {
         // Driver SQL instalado?
         Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
         Class.forName("com.mysql.jdbc.Driver");
         }
      catch(java.lang.ClassNotFoundException e) {
         // Se não achou, mostra erro
         System.err.println("Classe não encontrada: " + e.getMessage());
         }
      try {
         // Define endereço da conexão
         String endereco = "jdbc:sqlserver://localhost:1433;" +
               "databaseName=test;user=test;password=test;";
         String endereco = "jdbc:mysql://mysql.caetano.eng.br/estcioalunos";
         // Cria conexação e um objeto de transacao
         Connection conexao = DriverManager.getConnection(endereco);
         Connection conexao = DriverManager.getConnection(endereco,"estacioalunos",
               "estacioalunos");
         Statement transacao = conexao.createStatement();

         // Define a busca
         String query = "SELECT id, nome, sobrenome, idade FROM teste";
         // Executa busca e pega resultado
         ResultSet resultado = transacao.executeQuery(query);
         // Colhe informações sobre resultado (nome das colunas, número de linhas etc)
         ResultSetMetaData infoResultado = resultado.getMetaData();

         /* Imprimindo nomes das colunas em cada coluna */
         // Imprime nome da coluna 1 do resultado na linha 0 / coluna 0 da tabela
         tabela.setValueAt(infoResultado(1),0,0);
         // Imprime nome da coluna 2 do resultado na linha 0 / coluna 1 da tabela
         tabela.setValueAt(infoResultado(2),0,1);
         // Imprime nome da coluna 3 do resultado na linha 0 / coluna 2 da tabela
         tabela.setValueAt(infoResultado(3),0,2);
         // Imprime nome da coluna 4 do resultado na linha 0 / coluna 3 da tabela
         tabela.setValueAt(infoResultado(4),0,3);

         /* Loop que imprime dados na tabela */
         int linecount=1; // contador de linha sendo impressa
         while (resultado.next()) { // enquanto houver mais linhas...
            int i1 = resultado.getInt("id"); // pega coluna "id" como inteiro
            String s1 = resultado.getString("nome"); // pega coluna "nome" como string
            String s2 = resultado.getString("sobrenome"); // pega "sobrenome" como string
            int i2 = resultado.getInt("idade"); // pega coluna "idade" como inteiro

            // Coloca dados nas colunas certas da linha
            tabela.setValueAt(i1, linecount, 0);
            tabela.setValueAt(s1, linecount, 1);
            tabela.setValueAt(s2, linecount, 2);
            tabela.setValueAt(i2, linecount, 3);

            // Atualiza contador de linhas
            linecount = linecount + 1;
            }

         // Fim da Conexão
         transacao.close();
         conexao.close();
         }
      catch(SQLException ex) {
         System.err.println("Excessão no SQL: " + ex.getMessage());
         }
      }
   }
```

BIBLIOGRAFIA

DEITEL, H.M; DEITEL, P.J. **Java: como programar** - Sexta edição. São Paulo: Pearson-Prentice Hall, 2005.