Programação Orientada e Objetos II

Anhanguera

AVALIE SUA PROFISSÃO

QUANDO APARECER EM SEU PORTAL UMA AVALIAÇÃO SOBRE SEU CURSO, RESPONDA:

NOTAS **9 OU 10** SIGNIFICA QUE VOCÊ INDICA

NOTAS **7 OU 8** SIGNIFICA QUE VOCÊ NÃO INDICA

Anhanguera

Anhanguera

Para que seja possível o desenvolvimento de aplicações que se conectem a um sistema de gerenciamento de banco de dados (SGBD), é preciso conectar-se diretamente ao banco utilizando uma Application Programming Interface (API) que o fabricante do SGDB provê. Todavia, a operação torna-se mais dinâmica quando a linguagem de programação fornece alguma maneira de conexão universal, sem a necessidade de mudar de API para troca de cada SGDB.

Introdução ao uso do MySQL em programas Java Os SGBD são os elementos mais comuns para persistência de dados utilizados em aplicações comerciais, pois propiciam formas padronizadas para inserção, alteração, remoção e busca de dados. Portanto, é necessário verificar como as interfaces gráficas, quando acionadas pelo usuário, fazem o uso dos SGDBs para gravar seus dados.

Como existem diversos SGDBs, seria necessário utilizar bibliotecas específicas para cada sistema, o que causaria uma dependência ao tipo de persistência. Para utilizar os SGDBs em Java, especialmente em interfaces gráficas em Java Swing, é indicado utilizar o Java Database Connectivity (JDBC). O JDBC consiste em um conjunto de classes que são incorporadas ao Java Development Kit (JDK), para possibilitar o acesso a diversos SGDBs de forma padronizada sem a necessidade de se utilizar formas específicas para cada sistema de banco de dados (FURGERI, 2015). O JDBC é compatível com diversos sistemas de banco de dados, tais como:

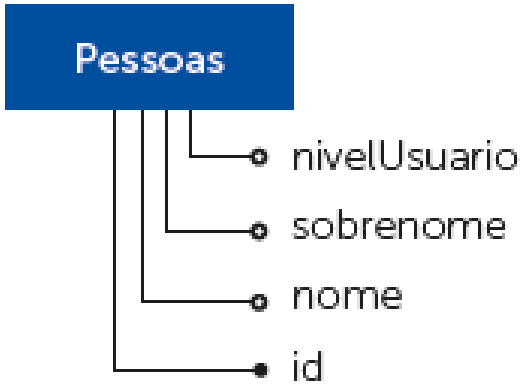
- MySQL: <https://www.mysql.com/>. Acesso em: 31 out. 2018.
- Oracle: <https://www.oracle.com/database/index.html>. Acesso em: 31 out. 2018.
- Microsoft SQL Server: <https://www.microsoft.com/en-us/sql-server/sql-server-2016>. Acesso em: 31 out. 2018.
- PostgreSQL: <https://www.postgresql.org/>. Acesso em: 31 out. 2018.

Dentre estas opções podemos destacar o MySQL, que consiste em uma solução open source, dispondo de versão comercial com suporte técnico e mais funcionalidades. Conta com elementos para controle de transação, triggers, suporte à conexão criptografada e muitos outros elementos de um sistema de banco de dados moderno. Além de suas características, apresenta diversas ferramentas que auxiliam no gerenciamento e desenvolvimento tais como o MySQL Workbench. Com essa ferramenta é possível criar, alterar e remover banco de dados, tabelas, procedures e outros elementos. Para obter o MySQL, no caso do Microsoft Windows, é necessário acessar https://www.mysql.com/downloads/ (ORACLE CORPORATION, 2018b), escolher a opção MySQL Community Edition e fazer o download de MySQL Community Server na opção MySQL Installer for Windows. O processo de instalação das últimas versões consiste em utilizar o assistente para tornar disponível o MySQL, sendo este um processo bem intuitivo.

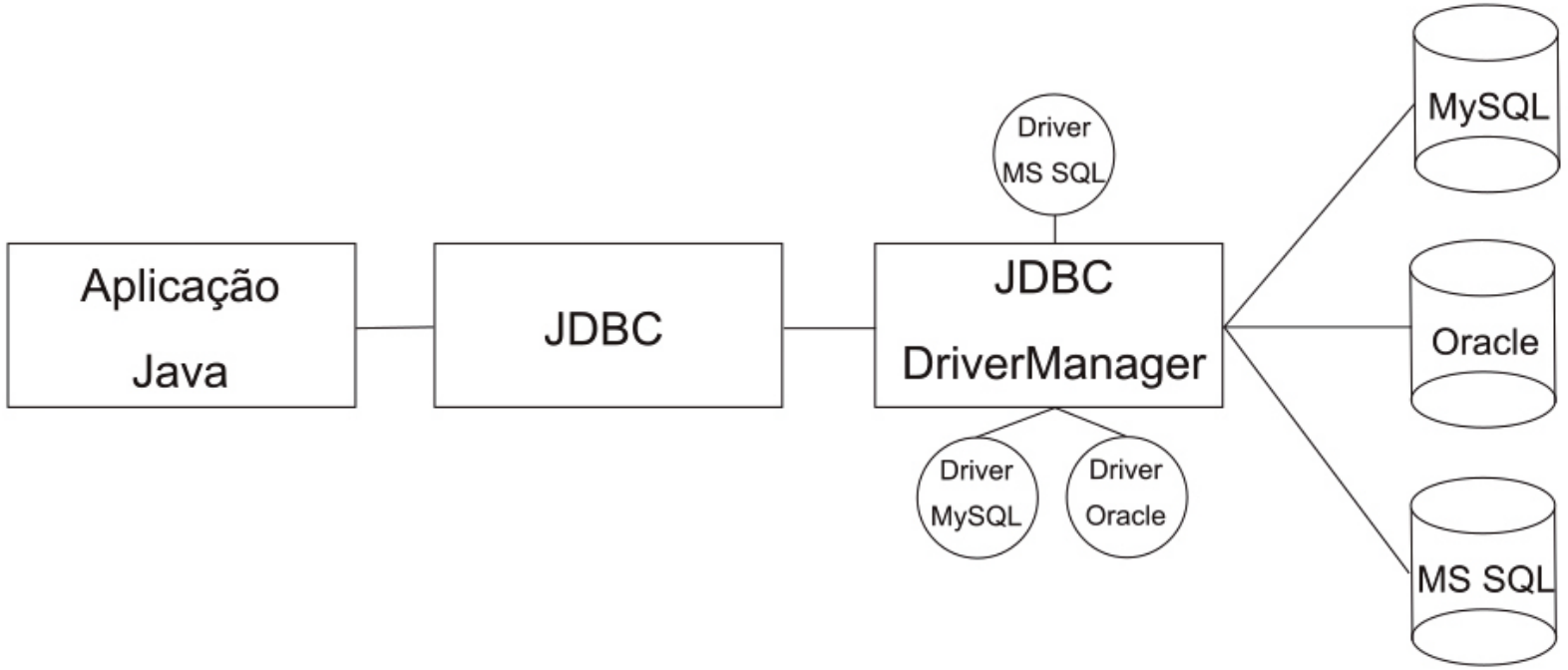
Para utilizar o JDBC com qualquer SGBD é necessário executar cinco passos, e então será possível executar comandos para buscardados ou enviar informações para o SGDB (DEITEL e DEITEL, 2016):

1. Estabelecer a conexão.
2. Criar um objeto da classe statement vinda da conexão para possibilitar a execução das consultas.
3. Executar as consultas.
4. Processar os resultados, sendo os dados enviados ou recebidos.
5. Fechar a conexão

A Figura 1.7 apresenta um exemplo de campos a serem armazenados para o cadastro de uma pessoa. Nosso objetivo é implementar esse diagrama na linguagem Java, que pertence ao paradigma da programação orientada a objetos, e para isso precisamos entender como funciona o JDBC.



A Figura 1.8 apresenta uma visão de como o JDBC atua nas aplicações Java utilizando a API do JDBC, que garante uma interface única para acesso ao SGBD. O JDBC utiliza as classes que controlam o *driver* que será utilizado para se conectar no banco de dados indicado. Para executar o primeiro passo, que consiste em estabelecer a conexão, é necessário garantir que o JDBC conheça o SGBD (HORSTMANN, 2016).



Lembrando: o JDBC é uma interface genérica para diversos sistemas de bancos de dados, e para que o JDBC possa fazer a conexão é necessário que ele conheça quais as peculiaridades do SGBC em que a conexão será estabelecida. Para isso, é preciso que o JDBC utilize um driver para o banco de dados que se deseja conectar, sendo preciso informar diretamente qual driver será utilizado (MANZANO,2014). A maneira de instalar depende do sistema operacional e do driver; nesse livro aprenderemos a usar o driver do MySQL.

A API JDBC é implementada pelos pacotes java.sql e javax.sql. Dentro desses pacotes estão as classes disponíveis para manipulação dos SGBD. Podemos destacar as classes:

- java.sql.DriverManager: essa classe é utilizada para criar a conexão com SGBD.
- java.sql.Conection: essa classe é utilizada para representar a conexão com o SGBD e fornecer acesso às consultas.
- java.sql.Statement: essa classe é utilizada para executar as consultas e comandos no SGDB.
- java.sql.ResultSet: essa classe é utilizada para recuperar os dados que foram buscados, por exemplo, um comando de select.
- javax.sql.DataSource: essa classe é utilizada para agrupar conexões com o SGBD.

Para se conectar a um banco de dados, esteja ele implementado em qualquer SGBD, é necessário criar uma string de conexão, ou URL JDBC (Uniform Resource Locator JDBC). Essa string informará o "caminho" do banco e apresenta a seguinte sintaxe:

jdbc:<driver>:<detalhes da conexão>

No item <driver> especifica-se qual SGBD será utilizado para conexão. Alguns exemplos são ilustrados no Quadro 1.15.

Banco de dados	URL JDBC
MySQL	jdbc:mysql://localhost:3306/nomeBancoDeDados
SQL Server	jdbc:sqlserver://localhost;databaseName=nomeBancoDeDados
Oracle	jdbc:oracle:thin@myserver:1521:nomeBancoDeDados

Conexão com banco de dados

O código no Quadro 1.16 apresenta como fazer a conexão ao MySQL utilizando o JDBC do Java. Entre as linhas 1 a 3 são incluídas algumas classes do JDBC que são necessárias para criar a conexão com o banco de dados. Na linha 6, a variável URLDB armazena a URL para o MySQL; cada SGBD apresenta formas diferentes de montar as informações de endereço e porta, nesse caso, especificamos o usuário local (localhost) com a porta padrão do MySQL 3306 e nome do banco de dados “nomebd”.

As linhas 7 e 8 representam o usuário e senha para a conexão, a linha 12 informa qual é o driver a ser carregado e a linha 13, o método getConnection(URLDB, usuario, senha), da classe DriverManager, faz a conexão utilizando todas as informações.

O comando Class.forName carrega uma classe específica, que fica à disposição para o ambiente que está sendo utilizado. No caso do JDBC, quando é feita a conexão com uma URL específica, ele busca a classe que acha mais indicada (WINDER, 2009). Para que o comando Class.forName tenha sucesso é necessário, em primeiro lugar, fazer o donwload do driver que se deseja usar. Por exemplo, para o MySQL você deverá acessar o endereço <https://www.mysql.com/products/connector/> (ORACLE CORPORATION, 2018a) e selecionar a opção Connector/J. No momento em que esse livro está sendo escrito, o conector está na versão 8.0.11, e essa informação é relevante pois pode impactar os parâmetros de conexão da URL. Esse arquivo poderá ser descompactado em qualquer local no computador. Com o arquivo já no computador, o próximo passo é fazer a devida configuração no Eclipse, e para isso você deve adicionar o caminho no sistema de arquivos na variável de ambiente CLASSPATH. No caso do Eclipse, é possível adicionar esse JAR como item do projeto: na aba Package Explorer, clique com o botão direto do mouse e selecione Properties, na opção Java Build Path selecione Libraries e depois clique em Add External JARS, e então adicione do local onde descompactou o conector no seu computador.

Anhanguera

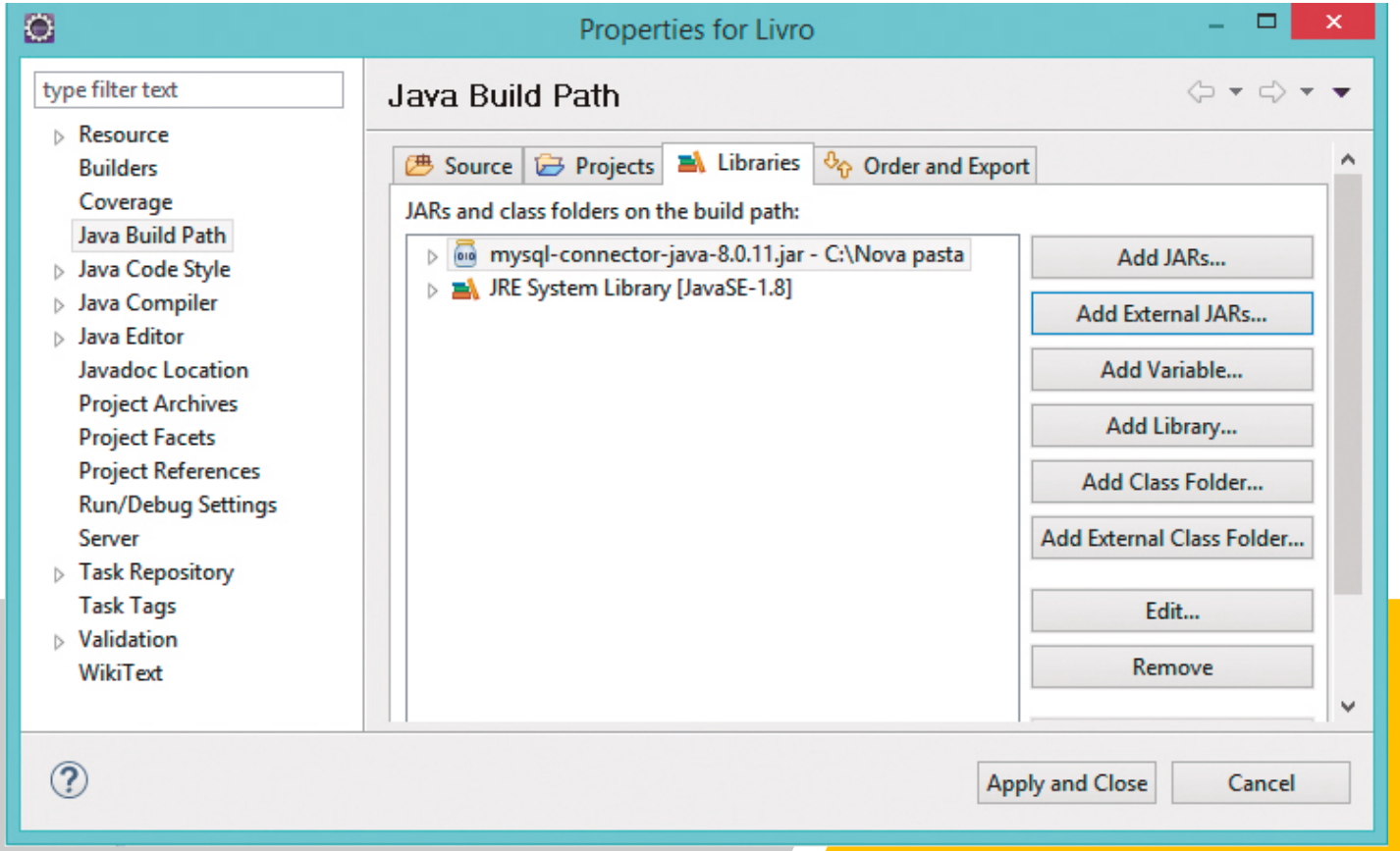
```
1.import java.sql.Connection;
2.import java.sql.DriverManager;
3.import java.sql.SQLException;

4.public class ConexaoBancoDeDados {

5.      private Connection conexao;
6.      private final String URLDB =
“jdbc:mysql://
7.      localhost:3306/nomebd”;
8.      private final String usuario = “user”;
9.      private final String senha = “senha”;

10.     public ConexaoBancoDeDados()
11.     {
12.           try {
13.                 Class.forName(“com.mysql.cj.
jdbc.Driver”);
14.                 conexao = DriverManager.get-
Connection(URLDB, usuario, senha);
15.           } catch (Exception e) {
16.                       e.printStackTrace();
17.           }
18.     }
19.     public static void main(String[] args) {
20.           ConexaoBancoDeDados conexao = new
ConexaoBancoDeDados();
21.     }
22.}
```

A Figura 1.9 apresenta as propriedades do projeto no Eclipse, no qual é possível verificar que elas já foram adicionadas ao JAR para conexão ao MySQL. Com isso, ao executar o método Class.for, a classe “com.mysql.cj.jdbc.Driver” ficará disponível para o JDBC.



Ações no banco de dados: inserção, leitura, atualização e exclusão de registros Com a classe do MySQL disponível será possível fazer a conexão com o banco de dados utilizando a classe Connection. Por exemplo, no código do Quadro 1.16 o objeto conexao da classe Connection garante o acesso a diversas ações no banco de dados. As ações no banco de dados são construídas com a classe Statement. Essas ações são conhecidas como CRUD (CreateRead, Update, Delete), pois representam as quatro operações possíveis em um banco de dados. Para seu uso é preciso adicionar a referência “java.sql.Statement;”.

O código no Quadro 1.17 demonstra como inserir elementos no banco. Na linha 2 é criado um objeto do tipo Statement chamado inserir, e na linha 3 o comando de inserção é executado. Veja que os comandos são em linguagem SQL.

```
import java.sql.Statement;
Statement inserir = conexao.createStatement();
inserir.execute((“INSERT INTO pessoa
(nome,sobrenome,nivelusuario) VALUES
(‘NomePessoa’,’Sobrenome’,1)”);
```

O processo para as demais operações é análogo: cria-se um objeto do tipo Statement e faz-se a execução. Como exemplo, o código no Quadro 1.18 apresenta as demais opções. Na linha 1 é criado o objeto do tipo Statement. Na linha 2, usamos o comando SQL para criar uma instrução de atualização na coluna “nome” da tabela “pessoa” que apresenta o id 1. Na linha 3, o comando exclui o registro da tabela “pessoa” cujo id é 1. Na linha 4, foi criado um novo objeto chamado res do tipo ResultSet, para armazenar o resultado da seleção dos campos nome e sobrenome da tabela “pessoa”. Para utilizar essa classe, é necessário incluir referência a java.sql.ResultSet. Veja que foi necessário usar o método executeQuery para fazer a seleção. Das linhas 5 a 8 criamos uma estrutura de repetição para percorrer o objeto que armazenou o resultado e imprimir os campos na tela. Para que seja possível buscar todos os dados é necessário executar o método “next()”.

Anhanguera

```
Statement comando = conexao.createStatement();
//comando para atualizar
comando.execute(“UPDATE pessoa SET nome = ‘Nome’
WHERE id = 1”);
//comando para deletar
comando.execute(“DELETE FROM pessoa WHERE id =
1”);
//comando para selecionar dados
ResultSet res = comando.executeQuery(“SELECT *
FROM pessoa”);
//comando para exibir os dados lidos
while(res.next())
{
        System.out.println(res.getString(“nome”));
}
```

Portanto, todo o processo de manipulação de dados utilizando o JDBC consiste na criação de strings utilizando os comandos SQL. Para facilitar esse processo, é possível utilizar a classe PreparedStatement, e com ela define-se qual comando SQL será utilizado e marca-se com “?” os locais na string em que os valores serão inseridos no comando. Ainda na classe PreparedStatement pode-se utilizar os comandos setString, setInt e outros para preencher os campos do insert, o que facilita a criação de instruções de forma mais organizada do que com o Statement. Veja um exemplo dessa classe no Quadro 1.19: na linha 1 é criadoum objeto chamado “psInsert” do tipo PreparedStatement, que apresenta um comando SQL (insert). Nas linhas 2 a 4 são definidos os valores que serão substituídos no objeto psInsert nos locais com “?”, ou seja, o compilador executará o seguinte insert: “INSERT INTO (nome, sobrenome,nivelusuario) VALUES (‘Nome’,’Sobrenome’,1)”, e por fim, na linha 5, o comando será executado.

Anhanguera

```
PreparedStatement psInsert = con.
prepareStatement(“INSERT INTO (nome,
sobrenome,nivelusuario) VALUES (?,?,?)”);
psInsert.setString(1, “Nome”);
psInsert.setString(2, “Sobrenome”);
psInsert.setInt(3, 1);
psInsert.execute();
```

Portanto, com os comandos Statement, PreparedStatement e ResultSet é possível enviar e receber dados do banco de dados. É importante lembrar-se da coesão das classes no modelo orientado a objetos, portanto as classes que utilizam o banco de dados não devem fazer outras tarefas. . Isso garante a coesão do sistema e aumenta as possibilidades de sua manutenção.aumenta as possibilidades de sua manutenção.

Para criar o acesso ao banco de dados é possível utilizar bibliotecas específicas, com isso se garante a forma correta de acesso para cada banco. Todavia, é possível utilizar o Java Database Connectivity (JDBC), que dispõe de um conjunto de elementos para garantir que a aplicação possa acessar diversos sistemas de banco de dados de maneira padronizada.

Quais dos elementos a seguir, integrantes do JDBC, garantem a conexão a diversos bancos de dados com a mesma interface?

a) Driver e DriverManager.
b) Connection e Statement.
c) PreparedStatement e Statement.
d) URL e Password.
e) com.mysql.jdbc.Driver e jdbc:mysq.l

Para conectar a um sistema de gerenciamento de banco de dados (SGBD) utilizando o JDBC, é necessário indicar qual driver utilizar. No momento em que o JDBC configura o objeto Connection, é preciso comunicar um conjunto de informações mínimas para garantir a utilização da conexão que está sendo criada.

Dos itens a seguir, quais deles representam as informações mínimas para conectar em uma instância de um banco de dados MySQL?

a) IP, usuário e senha.
b) DNS e usuário.
c) URL contendo jdbc:mysql://localhost:3306/, usuário e senha.
d) URL contendo jdbc:mysql://localhost:3306/bd, usuário e senha.
e) Usuário e senha.