

# Introdução ao JDBC



Programação Orientada a Objetos  
Dra. Alana Moraes

# Objetivo da Aula

- Compreender o funcionamento das ferramentas de mapeamento de acesso aos bancos de dados
- Utilizar componentes para acesso a dados em banco de dados.

# Por que usar Bancos de Dados?

- Muitos sistemas precisam manter as informações com as quais eles trabalham para permitir consultas futuras, geração de relatórios ou possíveis alterações nas informações.
- Para que esses dados sejam mantidos para sempre, esses sistemas geralmente guardam essas informações em um banco de dados, que as mantém de forma organizada e prontas para consultas.
- A maioria dos bancos de dados comerciais são os chamados relacionais (conceito de Entidade-Relacionamento), que é uma forma de trabalhar e pensar diferente ao paradigma orientado a objetos.

# MySQL

- Banco de dados que usaremos na presente aula.
- É um dos mais importantes bancos de dados relacionais, e é gratuito, além de ter uma instalação fácil para todos os sistemas operacionais.
- É importante entender alguns conceitos básicos sobre Banco de dados antes de começar.
- Geralmente roda na porta 3306
  - Cuidado para não esquecer a senha de root na instalação

# Banco de Dados

- O processo de armazenamento de dados é também chamado de persistência.
- A biblioteca de persistência em banco de dados relacionais do Java é chamada JDBC (Java DataBase Connectivity), e também existem diversas ferramentas do tipo ORM (Object Relational Mapping) que facilitam bastante o uso do JDBC.
- Neste momento, focaremos nos conceitos e no uso do JDBC.

# O que é JDBC?

- Java Database Connectivity define um conjunto de classes e interfaces para escrever programas ou sistemas em Java que utilizem Banco de Dados.
- Estas classes e interfaces efetuam as conexões entre a aplicação e o Sistema Gerenciador de Banco de Dados (SGBD).
- Ao utilizar o JDBC, podemos enviar comandos SQL ou PL/SQL para quase todos os bancos de dados relacionais.

# Principais Características

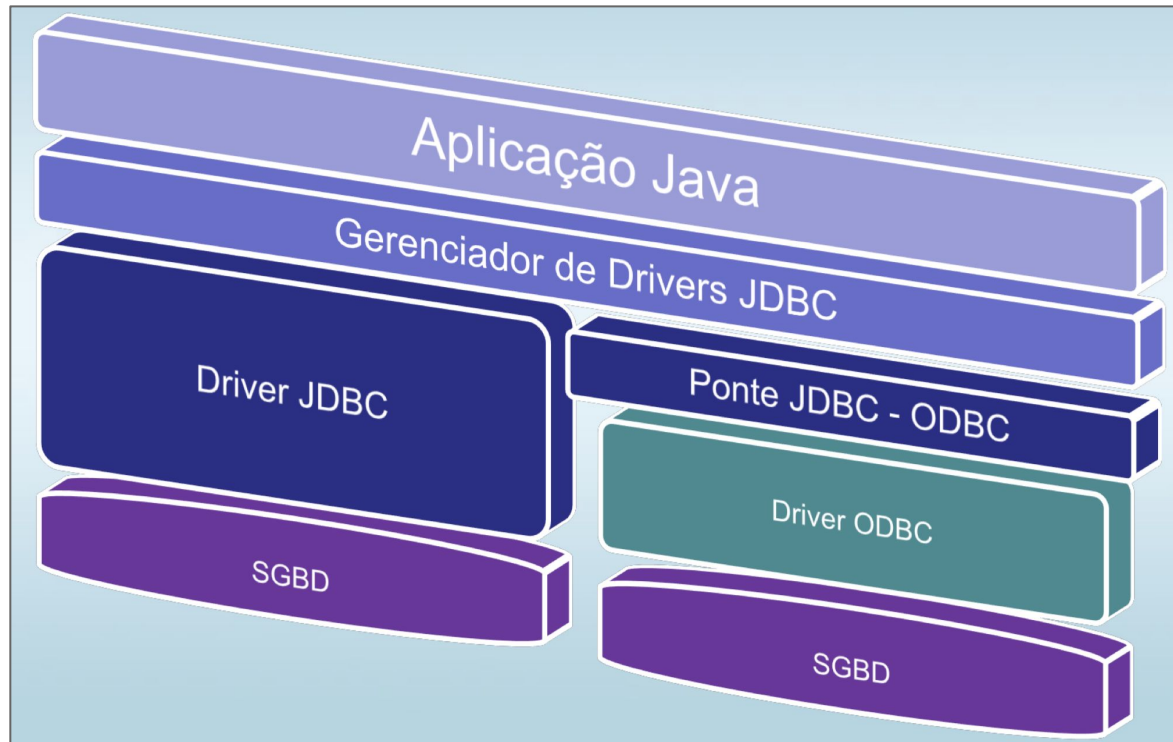
- Disponibiliza ao programador de aplicações Java abrir conexões em um SGBD, consultando ou modificando algum dado de um determinado BD, utilizando comandos SQL.
- Características:
  - Portabilidade
  - API independente do Banco de Dados utilizado
  - Permite o desenvolvimento de aplicações independente do banco de dados escolhido
    - Implementa a Estrutura em Camadas

# Estrutura de Acesso ao Banco de Dados

- Fornece um conjunto de API's de acesso para executar comandos SQL
- É implementado no pacote padrão java.sql, que é fornecido no SDK
- Funciona para qualquer tipo de Banco de Dados relacional que seu fabricante disponibilizou o Driver específico
- É independente de API ou da Linguagem proprietária dos fabricantes de SGBD



# Arquitetura JDBC



# Responsabilidade do JDBC

- Estabelecer conexão com o SGBD
- Enviar comandos SQL
- Receber os resultados
- Processar os resultados

# Tipos de Drivers JDBC

- Tipo1: ponte ODBC-JDBC
  - Usam uma ponte para ter acesso a um banco de dados. Este tipo de solução requer a instalação de software do lado do cliente.
- Tipo 2: solução com código nativo
  - Usam uma API nativa. Esses drivers contém métodos Java implementados em C ou C++. Requer software no cliente.
- Tipo 3: solução 100% Java no cliente
  - Oferecem uma API de rede via middleware que traduz requisições para API do driver desejado. Não requer software no cliente.
- Tipo 4: solução 100% Java
  - Drivers que se comunicam diretamente com o banco de dados usando soquetes de rede. É uma solução puro Java. Não requer código adicional do lado do cliente.

# Aplicação JDBC

API JDBC (pacote java.sql)

JDBC Driver Manager

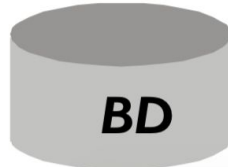
Protocolo JDBC

**Driver tipo 1**  
**Ponte JDBC-ODBC**

**Código nativo**

**Driver ODBC**

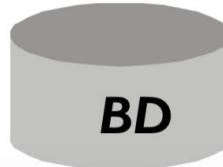
**Protocolo  
proprietário  
SGBD**



**Driver tipo 2**  
**API nativa**

**Código nativo**

**Protocolo  
proprietário  
SGBD**

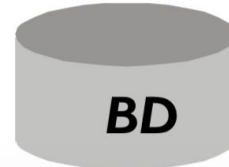


**Driver tipo 3**  
**JDBC-Rede**

**Protocolo de  
rede aberto**

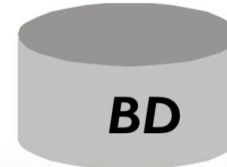
**Middleware**

**Protocolo  
proprietário  
SGBD**



**Driver tipo 4**  
**Pure Java**

**Protocolo  
proprietário  
SGBD**



# DriverManager e Driver

- A interface Driver é utilizada apenas pelas implementações de drivers JDBC
  - É preciso carregar a classe do driver na aplicação que irá utilizá-lo. Isto pode ser feito com `Class.forName()`:
    - `Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`
- A classe DriverManager manipula objetos do tipo Driver.
  - Possui métodos para registrar drivers, removê-los ou listá-los.
  - É usado para retornar Connection, que representa uma conexão a um banco de dados, a partir de uma URL JDBC recebida como parâmetro:
    - `Connection con = DriverManager.getConnection ("jdbc:odbc:dados", "nome","senha");`

# DriverManager e Driver

- Cada um dos bancos nos forneceria uma API que devolve uma instância de Connection diferente. Por exemplo:
  - `Connection c = new MySqlConnection("localhost/banco", "usuario", "senha");`
  - `Connection c = new SqlConnection("localhost/banco", "usuario", "senha");`
- DriverManager quem lida com isso:
  - `conexao = DriverManager.getConnection (stringDeConexao, usuario, senha);`

# DriverManager e Driver

Como eu sei qual o Driver eu estou conversando no projeto?

```
public class JDBCExemplo {  
    public static void main(String[] args) throws SQLException {  
        Connection conexao = DriverManager.getConnection(  
            "jdbc:mysql://localhost/fj21");  
        System.out.println("Conectado!");  
        conexao.close();  
    }  
}
```

# Connection, ResultSet e Statement

- Interfaces que contém métodos implementados em todos os drivers JDBC.
  - Connection
    - Representa uma conexão ao banco de dados, que é retornada pelo DriverManager na forma de um objeto.
  - Statement (PreparedStatement)
    - Oferece meios de passar instruções SQL para o sistema de bancos de dados.
  - ResultSet
    - É um cursor para os dados recebidos.



# Comandos MySQL

- CRUD
  - `INSERT INTO usuarios (nome, login, senha)VALUES ('João Carlos', 'joca', 'abc123')`
  - `SELECT login, senha FROM usuarios`
  - `UPDATE usuarios SET senha = '123456' WHERE id_usuario = 1`
  - `DELETE FROM usuarios WHERE login = 'joca'`

# Exemplo

- Driver do Mysql
  - <https://dev.mysql.com/downloads/connector/j/>
  - Exemplo de projeto no git (não esqueça de criar o banco de dados para rodar o projeto).

# Dúvidas?

[alanamm.prof@gmail.com](mailto:alanamm.prof@gmail.com)