DOMAIN DRIVEN DESIGN

Prof. Rafael Desiderio

JDBC

Introdução

- Um **banco de dados** é uma coleção organizada de dados;
- Um **sistema de gerenciamento de banco de dados (SGBD)** fornece mecanismos para **armazenar, organizar, recuperar e modificar** dados para muitos usuários;
- Os SGBD mais utilizados são os **relacionais (SGBDR);**
- A **SQL** é uma **linguagem de padrão internacional** utilizada com banco de dados relacionais para realizar consultas e manipulação de dados;
- São exemplos de **SGBDR**:
  - MySQL
  - Microsoft SQL Server
  - Oracle
  - Sybase
  - Informix

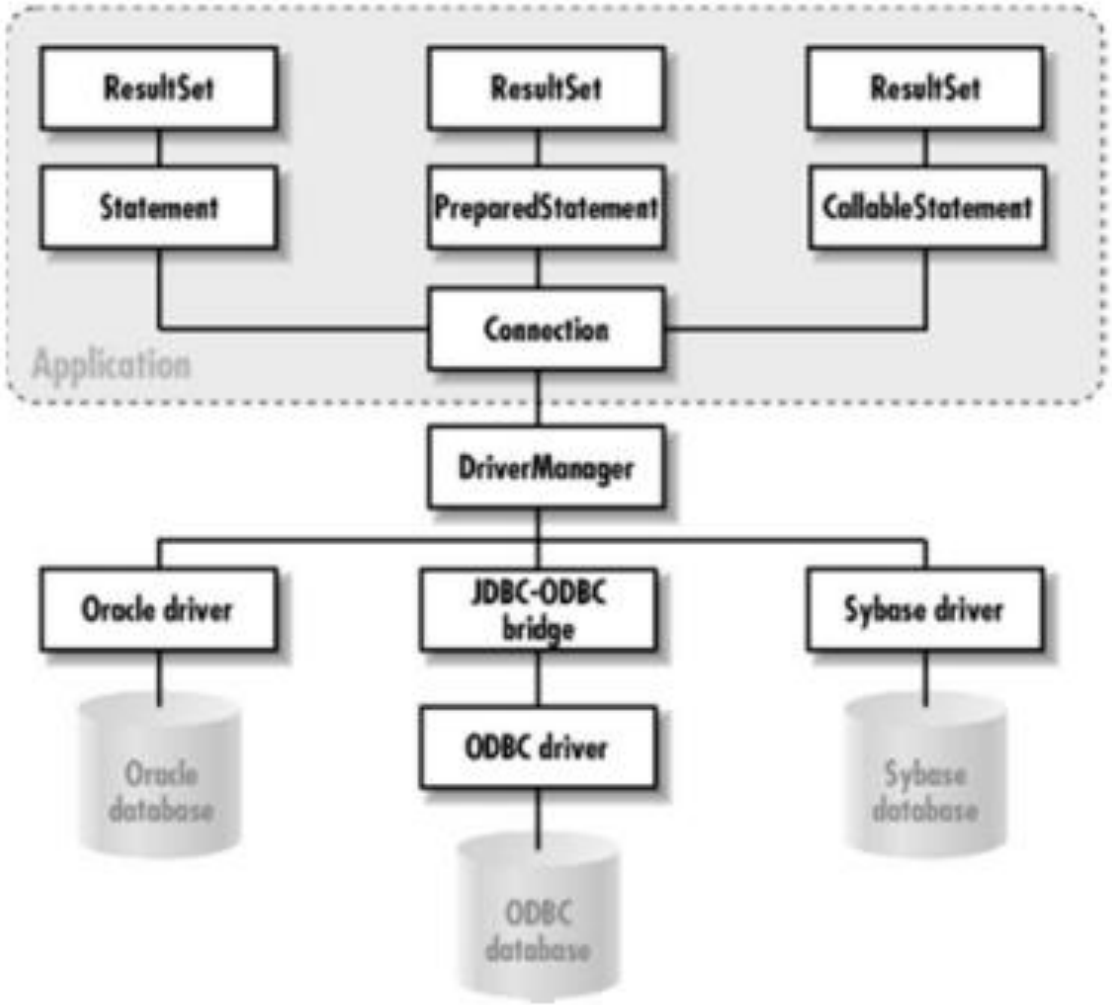
Introdução

- Os programas escritos em **Java** comunicam-se com o **SGBDR** e manipulam seus dados utilizando a **API Java DataBase Connectivity (JDBC);**

- Além do **JDBC**, para realizar a conexão com um SGBDR é preciso utilizar um **driver JDBC** – **específico** para cada **SGBDR;**

- O **driver JDBC** permite a uma **aplicação Java conectar-se** a um banco de dados e manipular os dados via **instruções SQL;**

Introdução


ResultSet
ResultSet
ResultSet
Statement
PreparedStatement
CallableStatement
Connection
Application
DriverManager
Oracle driver
JDBC-ODBC bridge
Sybase driver
Oracle database
ODBC driver
Sybase database
ODBC database

Principais instruções SQL

- **Criação de Tabela, CREATE TABLE:**

```
CREATE TABLE TB_CLIENTE (
  ID_CLIENTE NUMBER NOT NULL PRIMARY KEY,
  NOME VARCHAR2(100) NOT NULL,
  CPF CHAR(11) NOT NULL,
  DDD NUMBER(2),
  TELEFONE VARCHAR2(20),
  ATIVO NUMBER(1) DEFAULT 0,
  VALOR_ULTIMA_COMPRA NUMBER(10,2) DEFAULT 0
) SEGMENT CREATION IMMEDIATE;

CREATE SEQUENCE SQ_CLIENTE INCREMENT BY 1 START WITH 1 NOCYCLE ORDER;
```

Principais instruções SQL

- **Exclusão de Tabela, DROP TABLE:**

```
DROP SEQUENCE SQ_CLIENTE;

DROP TABLE TB_CLIENTE;
```

- **Alteração, UPDATE:**

```
UPDATE TB_CLIENTE SET DDD = 11 WHERE DDD IS NULL;
```

- **Exclusão, DELETE:**

```
DELETE FROM TB_CLIENTE WHERE NOME = 'Felix';
```

Principais instruções SQL

- **Seleção, SELECT:**

```
SELECT * FROM TB_CLIENTE;

SELECT ATIVO, NOME, CPF FROM TB_CLIENTE;
```

- **Condição, WHERE:**

```
SELECT * FROM TB_CLIENTE WHERE ATIVO = 1;

SELECT CPF, NOME FROM TB_CLIENTE WHERE NOME LIKE  '%ia%';

SELECT * FROM TB_CLIENTE WHERE (DDD IS NULL) OR (CPF IS NOT NULL) AND
((Nome = 'Maria') OR (NOME = 'Tiago')) AND VALOR_ULTIMA_COMPRA <> 1000;
```

Principais instruções SQL

- **Ordenação, ORDER BY:**

```sql
SELECT * FROM TB_CLIENTE ORDER BY NOME ASC;

SELECT ATIVO, NOME, CPF FROM TB_CLIENTE ORDER BY ATIVO DESC, NOME ASC;

SELECT * FROM TB_CLIENTE WHERE VALOR_ULTIMA_COMPRA > 0 ORDER BY NOME ASC;
```

Java DataBase Connectivity - JDBC

- **JDBC** é um conjunto de classes e interfaces escritas em Java que faz o envio de **instruções SQL para qualquer banco de dados relacional;**

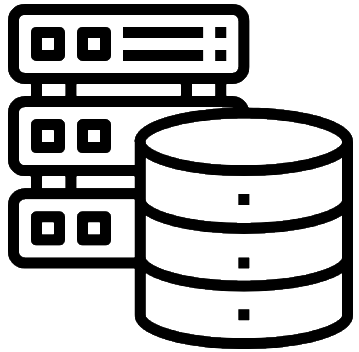
- **Biblioteca padrão** para atividades de persistência em base de dados, é uma **API de baixo nível e base para APIs de alto nível;**

- Um programa **Java** utiliza uma **API JDBC** única que **independe do banco de dados** ou driver que estiver sendo utilizado;
  - Assim, o **JDBC permite conectar com diversas base de dados** diferentes sem ter que alterar a implementação do programa em si;
  - A forma de **conexão, acesso de leitura e escrita, execução de comandos** etc. é padronizado na arquitetura JDBC;

Java DataBase Connectivity - JDBC

- Para **cada banco** de dados há um **driver específico;**

- Os **drivers para conexão e acesso** aos principais banco de dados existentes são **fornecidos pelos seus fabricantes;**



Drivers – Implemetação JDBC

- Para **conectar a base de dados**, deve-se primeiramente adicionar a **implementação (arquivo .jar)** desejada ao seu projeto (em algum lugar visível ou adicionado a variável de ambiente CLASSPATH);

- Feito isso, na classe onde irá fazer a conexão, deve-se registrar o **driver escolhido**. Isso é feito utilizando o método **Class.forName;**

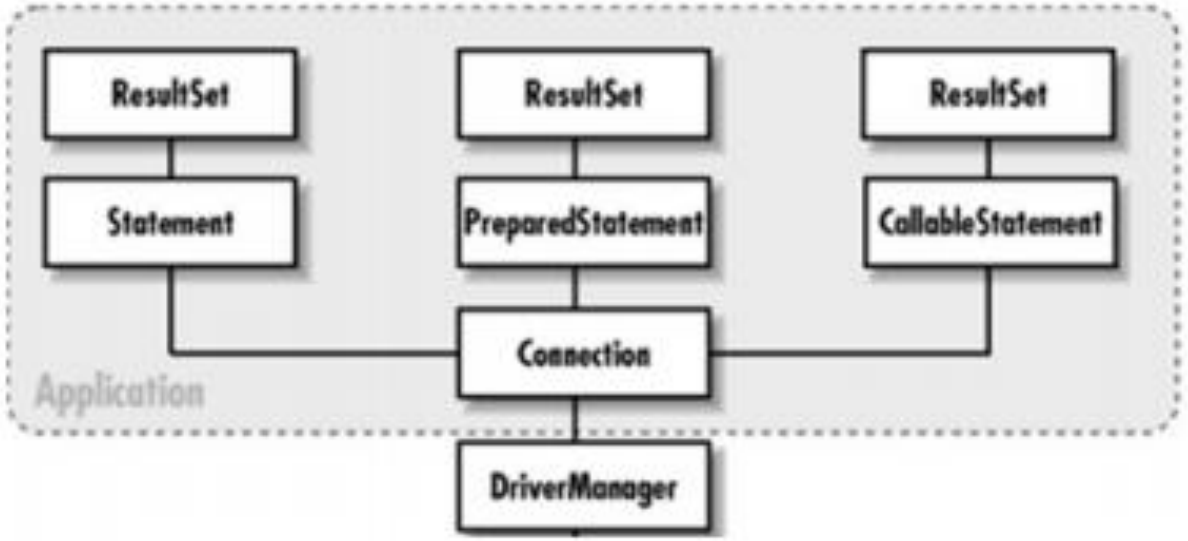
```
Class.forName("com.mysql.jdbc.Driver");
Class.forName("net.sourceforge.jtds.jdbc.Driver");
Class.forName("oracle.jdbc.driver.OracleDriver");
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Class.forName("org.hsqldb.jdbcDriver");
```

Drivers – Implemetação JDBC

- Os drivers devem ser baixados diretamente com os fabricantes do SGBDR, no caso do Oracle o driver é o **ojdbc**:

https://www.oracle.com/database/technologies/appdev/jdbc-downloads.html

Principais componentes do JDBC



Principais componentes do JDBC

Componente	Descrição
DriverManager	Responsável por encontrar o driver e estabelecer a conexão com o SGBDR
Connection	Representa a conexão com o SGBDR por onde serão passados os comandos SQL
Statement	Execução de comandos SQL
PreparedStatement	
CallableStatement	
ResultSet	Representa os registros retornados de um Statement, PreparedStatement ou CallableStatement

Conexão com o Banco de Dados

FIAP

- As classes para manipulação de banco de dados estão no pacote **java.sql;**

- Para **estabelecer uma conexão** com um SGBDR é preciso realizar duas tarefas distintas:
  - **registrar o driver;**
  - solicitar do gerenciador de drivers(**java.sql.DriverManager**) a abertura da conexão:
    - Se o **DriverManager** conseguir estabelecer a conexão com o **SGBDR** um objeto do tipo **java.sql.Connection** é retornado, caso contrário uma **exceção** é gerada;

Driver Manager

- O **DriverManager** é responsável por encontrar o driver que será utilizado na aplicação. Como registramos o driver anteriormente, o **DriverManager** será capaz de encontrá-lo;

- A partir do **DriverManager** serão criadas as conexões para a base de dados utilizando o método **getConnection** e passando a JDBC URL para a conexão;

- Para conectar com o Oracle por exemplo, a JDBC URL será:

- **Exemplo:**

```
jdbc:oracle:thin:@<IP ou nome da maquina>:<porta>:<serviço>

jdbc:oracle:thin:@192.168.60.15:1521:ORCL
```

Connection

- Uma **conexão** representa uma **sessão** com uma **base de dados** específica;

- Dentro do contexto de uma conexão, **comandos SQL são executados** e **resultados são retornados;**

- Podem existir **diversas conexões** abertas ao mesmo tempo;

- **Não esqueça de fechar** as conexões após sua utilização;

Exemplo de Conexão

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConexaoBanco {

 public static void main(String[] args) {

  try {
   Class.forName("oracle.jdbc.driver.OracleDriver"); // registra o driver
   Connection conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@oracle.fiap.com.br:1521:orcl", "usuario", "senha");

   System.out.println("Conectado!");

   conn.close(); // fecha a conexão

  } catch (ClassNotFoundException e) {
   System.out.println("O driver JDBC não foi encontrado");
   e.printStackTrace();
  } catch (SQLException e) {
   System.out.println("Não foi possível conectar no banco de dados");
   e.printStackTrace();
  }
 }

}
```

Execução de SQL no Banco de Dados

- Para executar uma SQL no SGBDR é necessário criar um ***Statement***
  - Toda operação no banco de dados é realizada através de um objeto do tipo ***Statement;***
  - O ***Statement*** representa um comando SQL;

- Para executar SQLs no SGBDR, três objetos do tipo ***Statement*** podem ser utilizados:
  - **Statement**
  - **PreparedStatement**
  - **CallableStatement**

Statement

- A interface **Statement** é utilizada para executar um **comando SQL** estático e obter os resultados produzidos por ele;

- **Principais métodos:**
  - **executeUpdate**
    - Executa um comando SQL(INSERT, UPDATE, DELETE) e **retorna o números de linhas afetadas;**
  - **executeQuery**
    - Executa um comando SQL(SELECT) e retorna o(s) resultado(s) num objeto do tipo **ResultSet;**

Statement

- Inclusão:

```
Statement stmt = conn.createStatement();
stmt.executeUpdate("INSERT INTO TB_CLIENTE(ID_CLIENTE,
NOME, CPF) values (SQ_CLIENTE.NEXTVAL, 'Felix', '0123547
8222')");
```

- Alteração:

```
Statement stmt = conn.createStatement();
stmt.executeUpdate("UPDATE TB_CLIENTE SET DDD = 11 WHERE
DDD is null");
```

- Exclusão:

```
Statement stmt = conn.createStatement();
stmt.executeUpdate("DELETE FROM TB_CLIENTE WHERE NOME =
'Felix'");
```

- Seleção:

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM TB_CLIE
NTE");
```

*Nos exemplos **conn** é uma instância de uma classe que implementa a interface **java.sql.Connection***

ResultSet

- Representa um **conjunto de registros** que retornam de uma SQL (SELECT);

- Os registros só podem **ser acessados sequencialmente** - cada chamada ao método **next** move o cursor para o próximo registro;

- Inicialmente o cursor **está posicionado antes do primeiro registro**. O método **next** retorna **true** se for capaz de posicionar na próxima linha;

- As colunas do registro podem ser **acessadas através de um índice inteiro** – onde 1 é o número da primeira coluna – ou através do nome da coluna;

ResultSet

- Principais métodos:
  - **next:** Move o cursor para a próxima linha.
  - **getInt:** Retorna o dados da coluna designada como um int do Java.
  - **getString:** Retorna o dados da coluna designada como uma String do Java.
  - **getBoolean:** Retorna o dados da coluna designada como um boolean do Java.
  - **getDouble:** Retorna o dados da coluna designada como um double do Java.

ResultSet - Exemplo

```
//..

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT * FROM TB_CLIENTE");

while (rs.next()) { //equanto houver registro..

  //pega o valor da coluna e coloca numa variável
  Integer id = rs.getInt("id_cliente");
  String nome = rs.getString("nome");
  boolean ativo = rs.getBoolean("ativo");
  double valorUltimaCompra = rs.getDouble("vl_compra");

  System.out.println(id + " " + " nome" + " " + ativo + " " + valorUltimaCompra);

}
```

Vamos à prática

A partir do arquivo fornecido(TabelaEstoque.csv), escreva um programa em Java para importar dados em um banco de dados.

```
id,tipo,marca,quantidade,valorCompra,valorVenda
1,TV,Philco,3,4999,5599
2,Notebook,Lenovo,2,3299,3555
3,Celular,Aple,1,6222,6777
4,Teclado,Logitech,4,152,183
5,Mouse,Dell,7,151,189
```

TB_ESTOQUE			
id	Integer	<pk>	not null
tipo	VarChar2(100)		not null
marca	VarChar2(100)		not null
quantidade	Integer		not null
valorCompra	Decimal(3,1)		not null
valorVenda	Decimal(3,1)		not null

FIAP

"O cerebro é como um músculo, quando pensamos bem, nos sentimos bem". – Carl Sagan