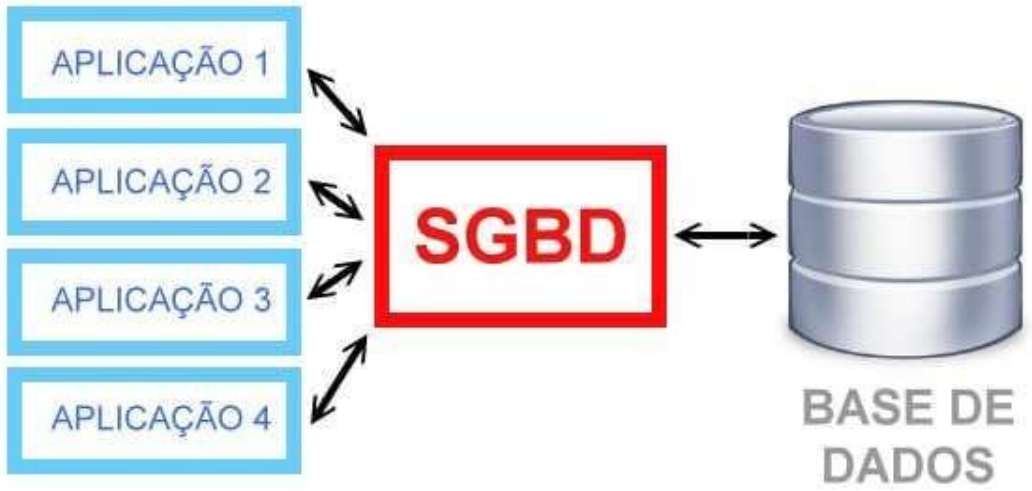
Banco de Dados

SGBD

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



existem diversos SGDBs

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)

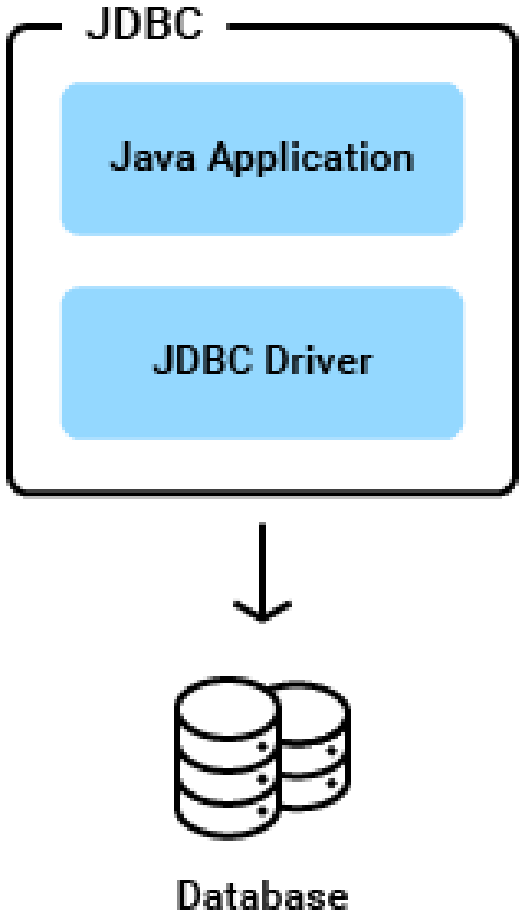
JDBC (Java Database Connectivity)

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).

Permite conectar e interagir com Banco de Dados.

O JDBC é compatível com diversos sistemas de banco de dados, tais como:

- MySQL
- Oracle
- PostgreSQL



JDBC controlador do driver

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).

java.sql e javax.sql.

java.sql.DriverManager: criar a conexão com SGBD.

java.sql.Conection: representar a conexão com o SGBD e fornecer acesso às consultas.

java.sql.Statement: executar as consultas e comandos no SGDB.

java.sql.ResultSet: recuperar os dados que foram buscados, por exemplo, um comando de select.

javax.sql.DataSource: agrupar conexões com o SGBD.

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

public class ConexaoMySQL {
    public static void main(String[] args) {
        // Dados de conexão com o banco
        String url = "jdbc:mysql://localhost:3306/escola"; // nome do banco = escola
        String usuario = "root";
        String senha = "1234";
```

URL de conexão

```
String url = "jdbc:mysql://localhost:3306/escola";
```

- jdbc:mysql:// → indica que o tipo de banco é MySQL.
- localhost → o banco está na mesma máquina.
- 3306 → porta padrão do MySQL.
- escola → nome do banco de dados que você criou.

```java
// Carregar o driver do MySQL
try {
    Class.forName("com.mysql.cj.jdbc.Driver"); // driver atualizado
    System.out.println("Driver JDBC carregado com sucesso!");
} catch (ClassNotFoundException e) {
    System.out.println("Erro ao carregar o driver: " + e.getMessage());
}
```

Carregando o driver

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

Isso garante que o Java reconheça o driver JDBC do MySQL.

```java
// Conectar ao banco de dados
try (Connection conexao = DriverManager.getConnection(url, usuario, senha))
    System.out.println("Conexão estabelecida com sucesso!");

    // Criar um objeto Statement para enviar comandos SQL
    Statement stmt = conexao.createStatement();
```

Criando a conexão

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

O DriverManager usa o driver carregado e cria a **ponte com o banco de dados**.

Criar o objeto

```
Statement stmt = conexao.createStatement();
```

Esse objeto serve para **enviar comandos SQL** para o banco.

```java
// Executar uma consulta SQL (SELECT)
String sql = "SELECT * FROM alunos";
ResultSet resultado = stmt.executeQuery(sql);

// Percorrer os resultados
while (resultado.next()) {
    int id = resultado.getInt("id");
    String nome = resultado.getString("nome");
    int idade = resultado.getInt("idade");

    System.out.println("ID: " + id + " | Nome: " + nome + " | Idade: " + idade);
}
```

Executando uma consulta (SELECT)

```
ResultSet resultado = stmt.executeQuery("SELECT * FROM alunos");
```

- O método executeQuery é usado para comandos **SELECT**.
- Ele retorna um **ResultSet**, que guarda os resultados da consulta.

```
            // Fechar recursos
            resultado.close();
            stmt.close();

        } catch (SQLException e) {
            System.out.println("Erro de conexão ou consulta: " + e.getMessage());
        }
    }
}
```