Acesso a banco de dados

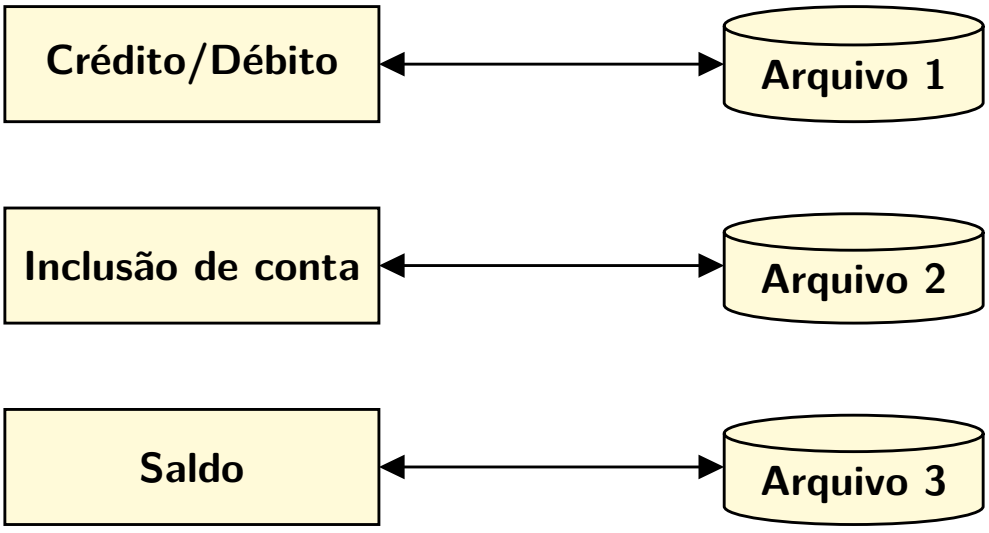
Prof. Renato Pimentel

2023/2

Sumário

Um caso de estudo: **um sistema bancário**

- Mantém informações de clientes e contas em arquivos permanentes do sistema;
- Contém programas (classes) que permitem a **manipulação dos dados**.
  - Creditar ou debitar uma conta;
  - Acrescentar nova conta;
  - Recuperar o saldo;
  - Gerar extratos.



Alguns aspectos:

- Programas desenvolvidos em resposta às demandas do negócio:
  - Novos métodos são adicionados à medida em que as necessidades aparecem;
  - **Novos arquivos** permanentes podem surgir.
- Como o sistema pode ser desenvolvido por diferentes profissionais, arquivos podem ter **formatos diferentes**.

Desvantagens

- **Redundância** de dados e **inconsistência**:
  - Os arquivos e programas são criados por diferentes programadores;
  - Mesma informação pode estar duplicada em diversos arquivos (ex.: endereço do cliente);
  - Maior **custo** de armazenamento;
  - Potencial inconsistência de dados.
- Dificuldade no **acesso** a dados:
  - Ex.: leitura sequencial para determinar o nome dos clientes com CEP 38408-971.

Coleção de arquivos **estruturados**, **não-redundantes** e **inter-relacionados**, que proporciona uma fonte única de dados para uma variedade de aplicações.

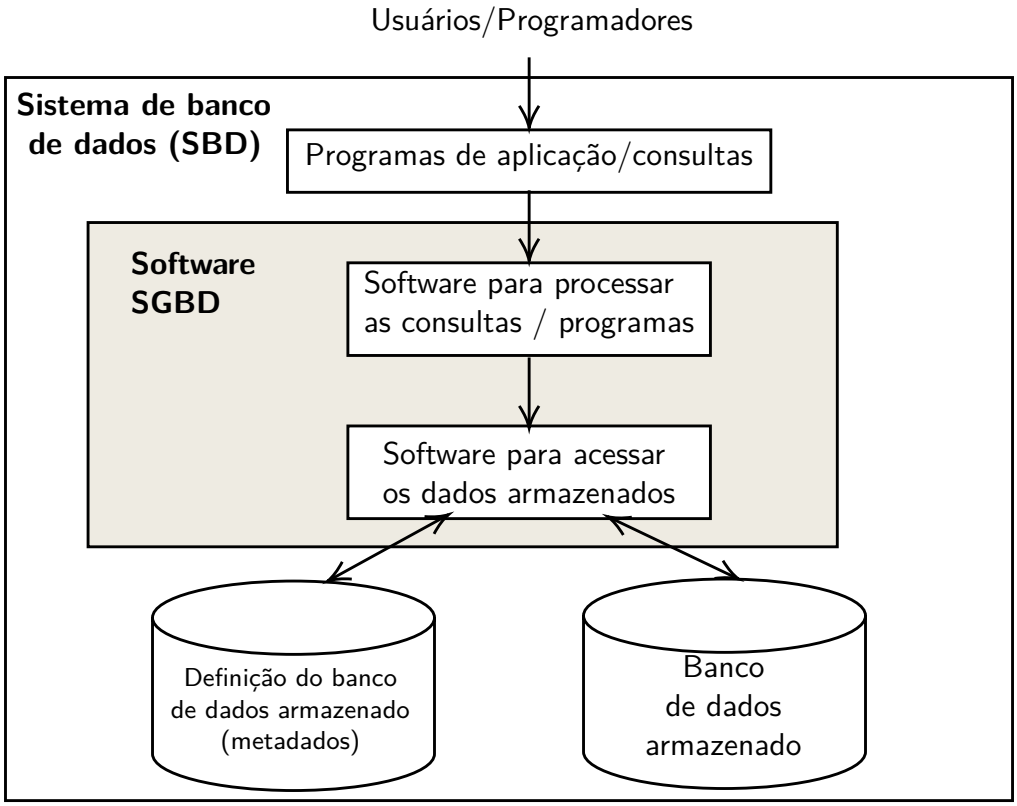
- Informações estruturadas:
  - São armazenadas de forma organizada em bancos de dados.
- Informações não-estruturadas:
  - Documentos físicos, como contratos, comprovantes de entrega, boletos de cobrança e informações do mercado financeiro ou de entidades governamentais.

Objetivos

- Oferecer **rapidez** e **flexibilidade** no acesso às informações;
- Garantir a **integridade** dos dados;
- Garantir o **sigilo** e a **segurança** dos dados;
- **Padronizar** os dados;
- Permitir **independência entre dados e programas**.

Sistema de gerenciamento de banco de dados

- Um **sistema de gerenciamento de banco de dados** (**SGBD**) é um conjunto de programas que permite criar e manter um banco de dados.
- Um banco de dados, juntamente com o SGBD que o gerencia, constitui um **sistema de banco de dados**.
  - Além de outras **aplicações**, que acessam o BD indiretamente, por meio do SGBD.



Classificação dos SGBDs

- Quanto ao modelo de dados adotado:
  - Hierárquicos;
  - De rede;
  - **Relacionais**;
  - Orientados a objetos.
- Quanto ao número de usuários suportados:
  - Mono-usuários;
  - Multi-usuários.
- Quanto à localização dos dados:
  - Centralizados;
  - Distribuídos.

Popularidade

419 systems in ranking, April 2024

Rank Apr 2024	Rank Mar 2024	Rank Apr 2023	DBMS	Database Model	Score Apr 2024	Score Mar 2024	Score Apr 2023
1.	1.	1.	Oracle	Relational, Multi-model	1234.27	+13.21	+5.99
2.	2.	2.	MySQL	Relational, Multi-model	1087.72	-13.77	-70.06
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	829.80	-16.01	-88.73
4.	4.	4.	PostgreSQL	Relational, Multi-model	645.05	+10.15	+36.64
5.	5.	5.	MongoDB	Document, Multi-model	423.96	-0.57	-17.93
6.	6.	6.	Redis	Key-value, Multi-model	156.44	-0.56	-17.11
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	134.78	-0.01	-6.29
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model	127.49	-0.26	-18.00
9.	9.	↑ 12.	Snowflake	Relational	123.20	-2.18	+12.07
10.	10.	↓ 9.	SQLite	Relational	116.01	-2.15	-18.53

Fonte: `https://db-engines.com/en/ranking`

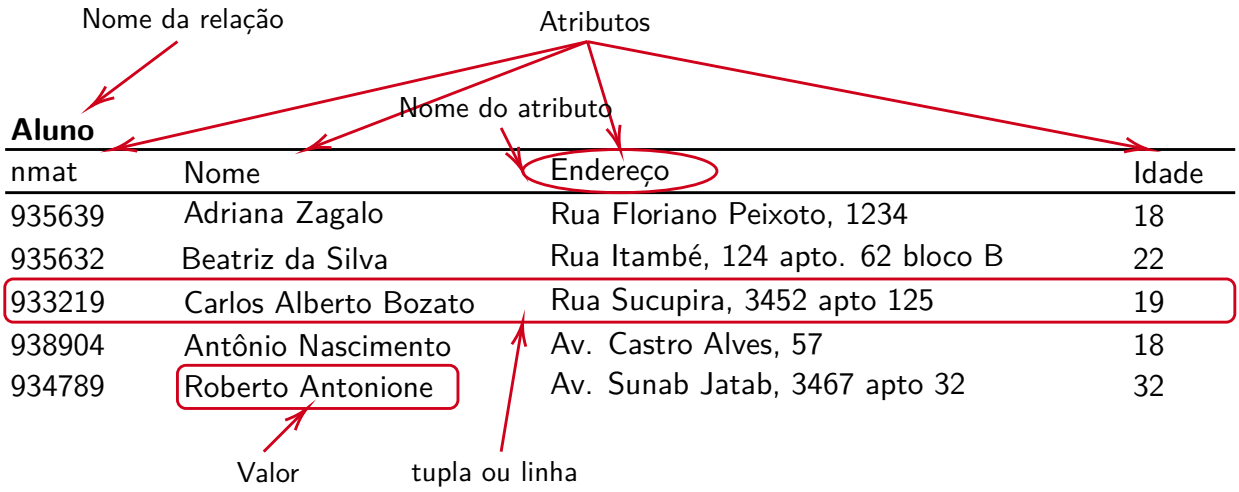
Modelo relacional

O modelo relacional representa um banco de dados como um **conjunto de relações**:

- Uma **relação** consiste numa **tabela** de valores, onde cada linha representa uma coleção de dados relacionados;
- Cada linha de uma tabela representa um "fato", que tipicamente corresponde a uma entidade ou relacionamento do mundo real.

Relação

- As linhas de uma relação (tabela) são chamadas de **tuplas**;
- Ao cabeçalho de cada coluna dá-se o nome de um **atributo** ou **campo**.
- O conjunto de valores que pode aparecer em cada coluna é chamado de **domínio**.



- Cabeçalho:
  - Número fixo de atributos (colunas).
- Corpo:
  - Número variável de linhas;
  - Não há ordenação de linhas.
  - Uma mesma linha não pode aparecer mais de uma vez.

Exemplo: parte de uma universidade

- Algumas entidades:
  - Alunos;
  - Disciplinas;
  - Departamentos.
- Alguns relacionamentos:
  - **Disciplinas** são oferecidas por **departamentos**;
  - **Alunos** estão matriculados em **disciplinas**.

ALUNO

Nome	Numero_aluno	Tipo_aluno	Curso
Silva	17	1	CC
Braga	8	2	CC

DISCIPLINA

Nome_disciplina	Numero_disciplina	Creditos	Departamento
Introdução à ciência da computação	CC1310	4	CC
Estruturas de dados	CC3320	4	CC
Matemática discreta	MAT2410	3	MAT
Banco de dados	CC3380	3	CC

TURMA

Identificador_turma	Numero_disciplina	Semestre	Ano	Professor
85	MAT2410	Segundo	07	Kleber
92	CC1310	Segundo	07	Anderson
102	CC3320	Primeiro	08	Carlos
112	MAT2410	Segundo	08	Chang
119	CC1310	Segundo	08	Anderson
135	CC3380	Segundo	08	Santos

REGISTRO_NOTA

Numero_aluno	Identificador_turma	Nota
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PRE_REQUISITO

Numero_disciplina	Numero_pre_requisito
CC3380	CC3320
CC3380	MAT2410
CC3320	CC1310

A linguagem SQL

- Desenvolvida e implementada pelo laboratório de pesquisa da IBM em San Jose – 1974.
- Inicialmente chamada de **SEQUEL** (*Structured English QUEry Language*).
- Criada como interface entre usuários e o primeiro SGBDR – **SYSTEM R**.

Exemplos de SGBD que utilizam SQL

- Oracle
- Informix
- Ingress
- MS SQL Server
- Interbase/Firebird
- Sybase
- DB2
- MySQL
- PostgreSQL

Padronização do SQL

Objeto de um esforço conjunto da *American National Standard Institute* (ANSI) e da *International Organization for Standardization* (ISO).

- SQL:2011
- SQL:2008
- SQL:2006
- SQL:2003
- SQL:1999 (SQL3)
- SQL-92 (SQL2)
- SQL-86 (primeira padronização ANSI)

Composição do SQL

Linguagem de Definição dos Dados (DDL – *Data definition language*)

Comandos para a definição, a modificação e a remoção de relações, além da criação e da remoção de índices.

Linguagem Interativa de Manipulação dos Dados (DML – *Data manipulation language*)

Comandos para a consulta, a inserção, a remoção e a modificação de tuplas no banco de dados.

- Linguagem de Manipulação dos Dados Embutida
  - Pode ser utilizada a partir de **linguagens de programação** de propósito geral

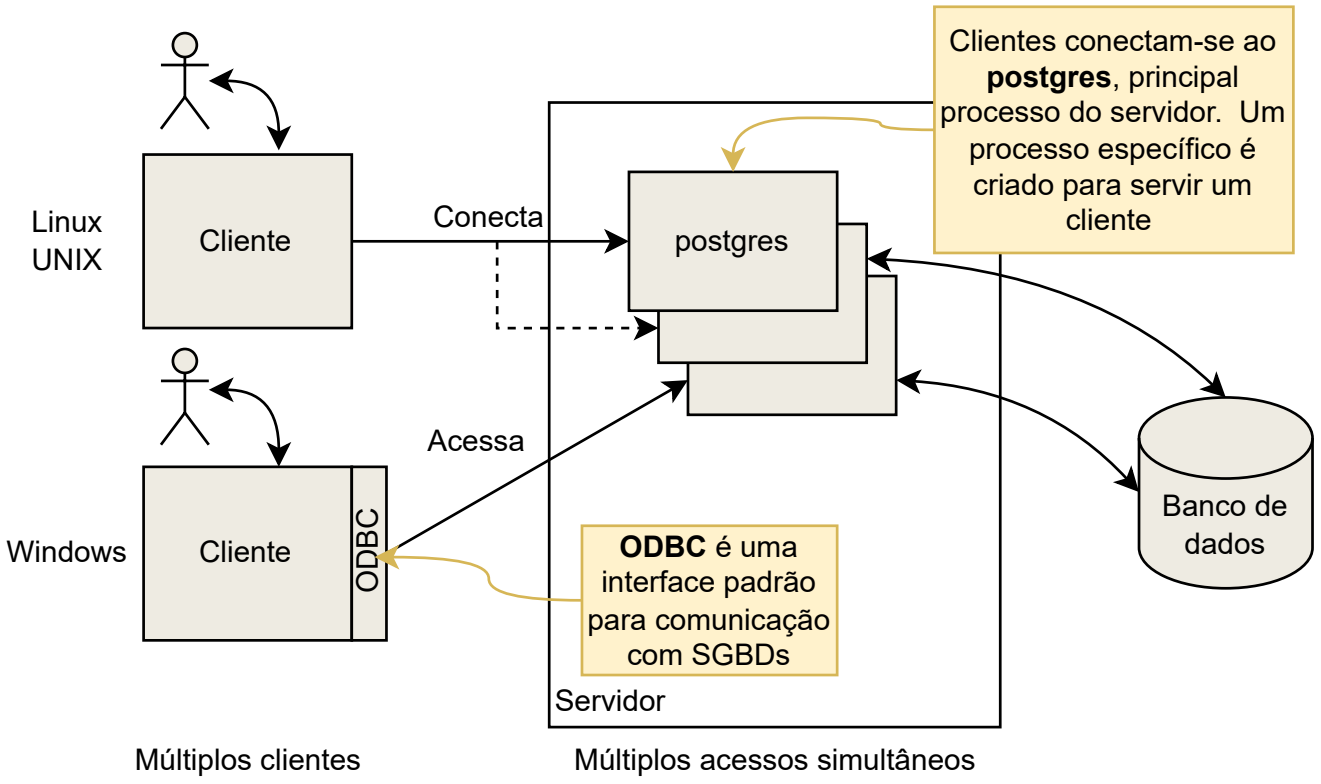
Exemplos de comandos da SQL

- `SELECT` é o mais usado da DML (linguagem de manipulação de dados), comanda e permite ao usuário especificar uma *query* como uma descrição do resultado desejado;
- `INSERT` é usada para somar uma linha (formalmente uma tupla) a uma tabela existente;
- `UPDATE` para mudar os valores de dados em uma linha de tabela existente;
- `DELETE` permite remover tuplas existentes numa tabela.

- **PostgreSQL**[1] é um SGBD que incorpora o **modelo relacional** para o banco de dados e suporta a **linguagem SQL** como linguagem de consulta padrão.
  - Em sua documentação[2], descrito como SGBDOR (SGBD Objeto-relacional).
- Ferramenta **multi-plataforma**: possui pacotes de instalação para o Windows (64 bits somente a partir da versão 11); MacOS (amd64, arm64); e UNIX (BSD, Linux e Solaris).
- É um **software livre** com **código-fonte aberto** (*open-source*).

[1] https://www.postgresql.org
[2] https://www.postgresql.org/docs/

Arquitetura cliente/servidor do PostgreSQL



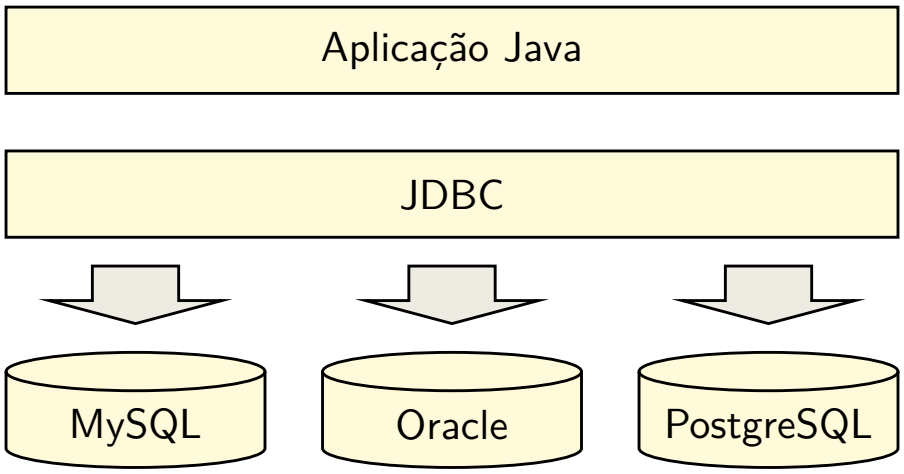
Acessando os dados em PostgreSQL

- Aplicação gráfica: **pgAdmin** (versão atual: 4).
  - Comumente utilizado no aprendizado de BD.
- Acesso indireto via ODBC ou **JDBC** (**Java Database Connectivity**).
- etc. (acessos via linha de comando, servidor web, ...)

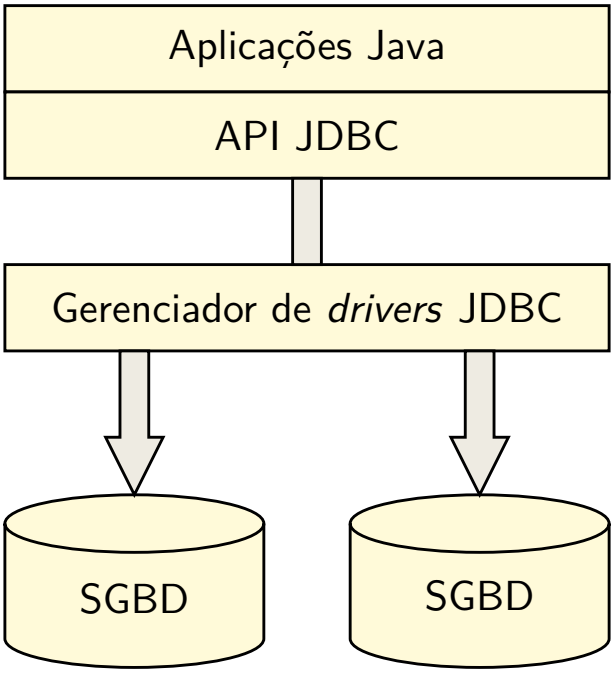
- Diferentes SGBDs possuem formas distintas de se comunicar com uma aplicação:

Necessidade do aprendizado de várias APIs.

- Solução: **JDBC**



- O JDBC provê um conjunto de interfaces para acesso ao BD.
  - Trata-se de um conjunto de APIs (bibliotecas de classes);
- Cada SGBD possui um *driver JDBC* específico (que é usado de forma padrão – JDBC).
- A Mudança do *driver* **não afeta** a aplicação;
- Os desenvolvedores dos SGBDs são responsáveis por implementar, disponibilizar e atualizar os *drivers* de acesso às suas bases de dados.

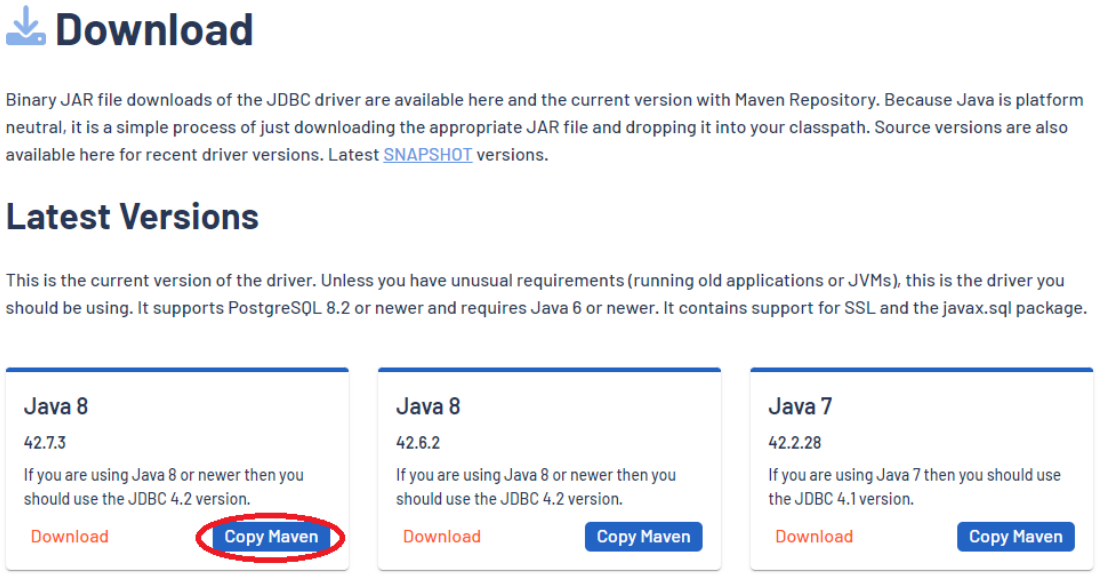


Pacote `java.sql`: fornece **classes** para serem usadas pelas aplicações.
Mais detalhes: `https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html`

Tipos de *drivers* JDBC

- Atualmente os fabricantes disponibilizam *drivers* JDBC do **tipo 4** (*pure Java driver*)
  - Totalmente implementados em Java;
  - Arquivo .jar único;
  - Conhece todo o protocolo de comunicação com o BD, podendo acessá-lo sem software extra.
- *Drivers* obsoletos:
  - Tipo 1: ponte JDBC-ODBC;
  - Tipo 2: *driver* com API nativa (parcialmente Java);
  - Tipo 3: *driver* de rede (comunica com servidor de aplicação).

1. Disponível em https://jdbc.postgresql.org/download/;



2. Para usá-lo, abra o NetBeans e crie um novo projeto Java;
3. Usando o painel `Projects` à esquerda, abra o arquivo `pom.xml` (pasta `Project Files`) e, após `</properties>`, insira a sub-árvore `<dependencies>`, como segue:

```
1    <dependencies>
2            <dependency>
3                <groupId>org.postgresql</groupId>
4                <artifactId>postgresql</artifactId>
5                <version>42.7.3</version>
6            </dependency>
7    </dependencies>
```

- As linhas 2 a 6 acima são obtidas diretamente do site do pgJDBC, bastando clicar em `Copy Maven`, como destacado na figura anterior.

- `DriverManager` – gerencia o *driver* e cria uma conexão com o banco.
  - O método `DriverManager.getConnection()` é chamado para efetuar a conexão com o banco de dados;
  - Mais detalhes: https://docs.oracle.com/javase/8/docs/api/java/sql/DriverManager.html
- `Connection` – é a interface que representa uma conexão (sessão) com o banco de dados.
  - Mais detalhes: https://docs.oracle.com/javase/8/docs/api/java/sql/Connection.html

- `Statement` – interface que controla e executa uma instrução SQL, retornando seu resultado.
  - `executeUpdate()` – utilizado para comandos `INSERT`, `UPDATE` e `DELETE`;
  - `executeQuery()` – utilizado para o comando `SELECT`.
- `PreparedStatement` – subinterface de `Statement` que controla e executa uma instrução SQL pré-compilada.
- `ResultSet` – interface que contém o conjunto de dados retornado por uma consulta SQL.
- `ResultSetMetaData` – é a interface que trata dos metadados do banco.

JDBC – etapas

1. Carregar *driver*;
2. Definir URL (ou IP) para a conexão;
3. Estabelecer conexão;
4. Criar objeto da classe `Statement`;
5. Executar uma consulta;
6. Processar resultado;
7. Encerrar a conexão.

Carregando o *driver*

```
1  System.out.print("Testando se o driver está registrado
       com DriverManager: ");
2
3  try {
4      Class.forName("org.postgresql.Driver"); // classe de
       driver do SGBD
5  } catch (ClassNotFoundException cnfe) {
6      System.out.println("Não foi possível localizar o
       driver.");
7      cnfe.printStackTrace();
8      System.exit(1);
9  }
10 System.out.println("O registro do driver está ok");
```

Criando uma conexão

```java
Connection conexao = null;
Statement sentenca = null;
try {
    // Estabelece conexão com o banco de dados
    System.out.print("Conectando com o servidor: ");
    String url = "jdbc:postgresql://localhost/postgres?user=
    postgres&password=12345";
    conexao = DriverManager.getConnection(url);
    System.out.println("conectado!");
    // Cria uma sentença para consultar o banco de dados
    sentenca = conexao.createStatement();
} catch (SQLException se) {
    System.out.println("Não foi possível conectar ao banco de
    dados.");
    se.printStackTrace();
     System.exit(1);
}
```


Criando uma sentença para alterar esquema

```java
try {
    // Criar tabela (relação)
    sentenca.execute("create table pessoa(id decimal(10)
    primary key,"
            + "nome varchar(20), endereco varchar(20))");

    // Inserir dados
    sentenca.execute("insert into pessoa values (123, 'André
    Silva', 'Av. Brasil, 100')");
    sentenca.execute("insert into pessoa values (234, 'João
    Bezerra', 'Av. João Naves, 300')");
    sentenca.execute("insert into pessoa values (345, 'Maria
    Bonita', 'Av. Tiradentes, 400')");
    sentenca.execute("insert into pessoa values (456, 'Joana
    Darc', 'Rua Principal, 200')");
} catch (SQLException se) {
    System.out.println("Não foi possível criar e povoar a relaç
    ão");
    se.printStackTrace();
    System.exit(1);
}
```

Criando uma sentença para atualizar o BD

```java
try {
    ResultSet resposta = sentenca.executeQuery("select * from pessoa");
    ResultSetMetaData metaDados = resposta.getMetaData();
    int nroColunas = metaDados.getColumnCount();
    for (int i = 1; i <= nroColunas; i++) {
        System.out.printf("%-8s\t", metaDados.getColumnName(i));
    }
    System.out.println();

    // exibir o conteúdo da tabela
    while (resposta.next()) {
        for (int i = 1; i <= nroColunas; i++) {
            System.out.printf("%-8s\t", resposta.getObject(i));
        }
        System.out.println();
    }
} catch (SQLException se) {
    System.out.println("Não foi possível executar a consulta");
    se.printStackTrace();
    System.exit(1);
}
```


Encerrando a conexão

```java
// fechar conexões
try {
    sentenca.close();
    conexao.close();
} catch (SQLException se) {
    System.out.println("Não foi possível encerrar a conexão");
    se.printStackTrace();
    System.exit(1);
}
```

Referências

1. ELMASRI R.; NAVATHE, S. *Sistemas de banco de dados*, Tradução da 7a. edição, Addison-Wesley, São Paulo, 2018.
2. PostgreSQL: *The world's most advanced open source database*. Disponível em: <https://www.postgresql.org/>. Acesso em: 27 out. 2022.
3. TRAVENÇOLO, B. A. N. (FACOM/UFU). Notas de aula.

Os materiais de parte desta seção foram gentilmente cedidos por Bruno A. N. Travençolo e Marcelo Z. do Nascimento (FACOM/UFU)
Adaptações: Renato Pimentel, FACOM/UFU