Universidade Federal do Amazonas
Instituto de Computação
Projeto de Programas
Técnicas Avançadas de Programação

Banco de Dados MySQL, MariaDB

Horácio Fernandes
horacio@icomp.ufam.edu.br

ColabWeb: bit.ly/pp-colabweb

Sistema de Gerenciamento de Banco de Dados

- Sistema de Gerenciamento de Banco de Dados (SGBD)
  - Do inglês, *Database Management System* (DBMS)
  - Permite o armazenamento, modificação, acesso e remoção de dados de um ou mais banco de dados.
  - Dados são armazenados em tabelas. Diversas tabelas formam um banco de dados.
  - Comandos são executados através do SQL:
    - *Structured Query Language*
    - *Linguagem para “consultas"*

Principais SGBDs

MySQL
- *mysql.com*
- Comprado/Oracle
- Rápido, Leve
- Código Aberto

MariaDB
- *mariadb.org*
- Fork do MySQL
- Rápido, Leve
- Código Aberto

PostgreSQL
- *postgresql.org*
- Mais robusto
- Mais recursos
- Código Aberto

SQLite
- *sqlite.org*
- Biblioteca
- Usado no Android
- Código Aberto

Firebird
- *firebirdsql.org*
- Fork do Interbase
- Desv. desde 1981
- Código Aberto

Oracle
- *oracle.com*
- Rápido
- Vários recursos
- Comercial

DB2
- *ibm.com*
- Rápido
- Vários recursos
- Comercial

SQL Server
- *microsoft.com*
- Só p/ Windows
- Código fechado
- Comercial

MySQL

- Código aberto
- É rápido, relativamente leve mas, ainda, relativamente poderoso
- Perfeito para sistemas de pequeno e médio porte, mas perfeitamente capaz de ser usado em sistemas de grande porte
  - Já foi ou ainda é usado por sistemas como Joomla, Wordpress, Facebook, Twitter, YouTube, etc

MySQL - Instalação

- Instalação do MySQL
  - Ubuntu: `sudo apt install mysql-server`
  - Outros sistemas: http://www.mysql.com

MySQL - Acessando

- Entrando no MySQL usando a linha de comando

```
$ sudo mysql -u root

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43815
Server version: 8.0.21-0ubuntu0.20.04.4 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Criando um Novo Banco de Dados

- O MySQL gerencia bancos de dados
- Um banco de dados (BD) é, principalmente, um conjunto de tabelas
  - Tabelas são as estruturas que irão armazenar os dados
- Em geral, cada aplicação terá o seu próprio BD
  - Ou seja, um BD para cada aplicação

Criando um Novo Banco de Dados

- Criando um novo BD

```
mysql> CREATE DATABASE CitacoesBD;
Query OK, 1 row affected (0,00 sec)
```

- Listando todos o BDs

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| CitacoesBD         |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0,00 sec)
```

Entrando no Novo Banco de Dados

- Entrando no novo BD criado

```
mysql> USE CitacoesBD;
Database changed
```

  - Após o comando `USE`, todos os comandos seguintes irão ser executados no banco de dados especificado

  - OBS: Os comandos SQL, em geral, ignoram maiúsculas ou minúsculas
    - *O comando acima poderia ser:* `use CitacoesBD`
    - *Entretanto, os nomes (do BD, tabelas, colunas, etc) consideram sim letras maiúsculas ou minúsculas*
    - *Por fim, é comum escrever os comandos SQL todos em maiúsculas*

Criando um Novo Usuário

- Preferencialmente, cada BD terá pelo menos um usuário
  - Este usuário terá acesso apenas a este BD, e não a todos os BDs

```
mysql> CREATE USER 'citacoes_admin'@'localhost' IDENTIFIED BY 'Teste123';
Query OK, 0 rows affected (0,02 sec)
mysql> GRANT ALL PRIVILEGES ON CitacoesBD.* TO 'citacoes_admin'@'localhost' WITH
            GRANT OPTION;
Query OK, 0 rows affected (0,08 sec)
```

- Saindo e entrando com o novo usuário

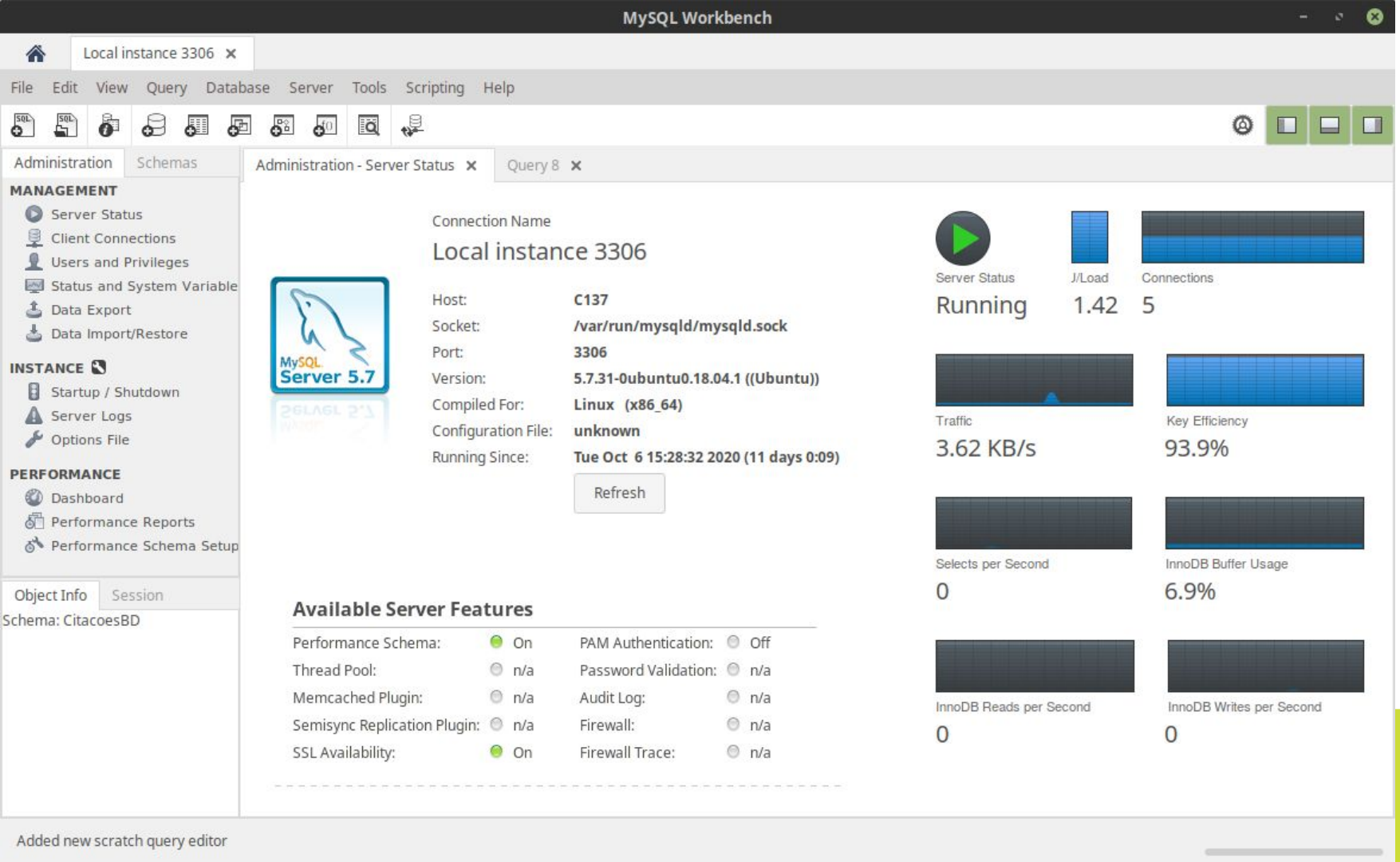
```
mysql> exit
Bye
$ mysql -u citacoes_admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
(...)
Type 'help;' or '\h' for help. Type '\c' ...

mysql> USE CitacoesBD;
Database changed
```

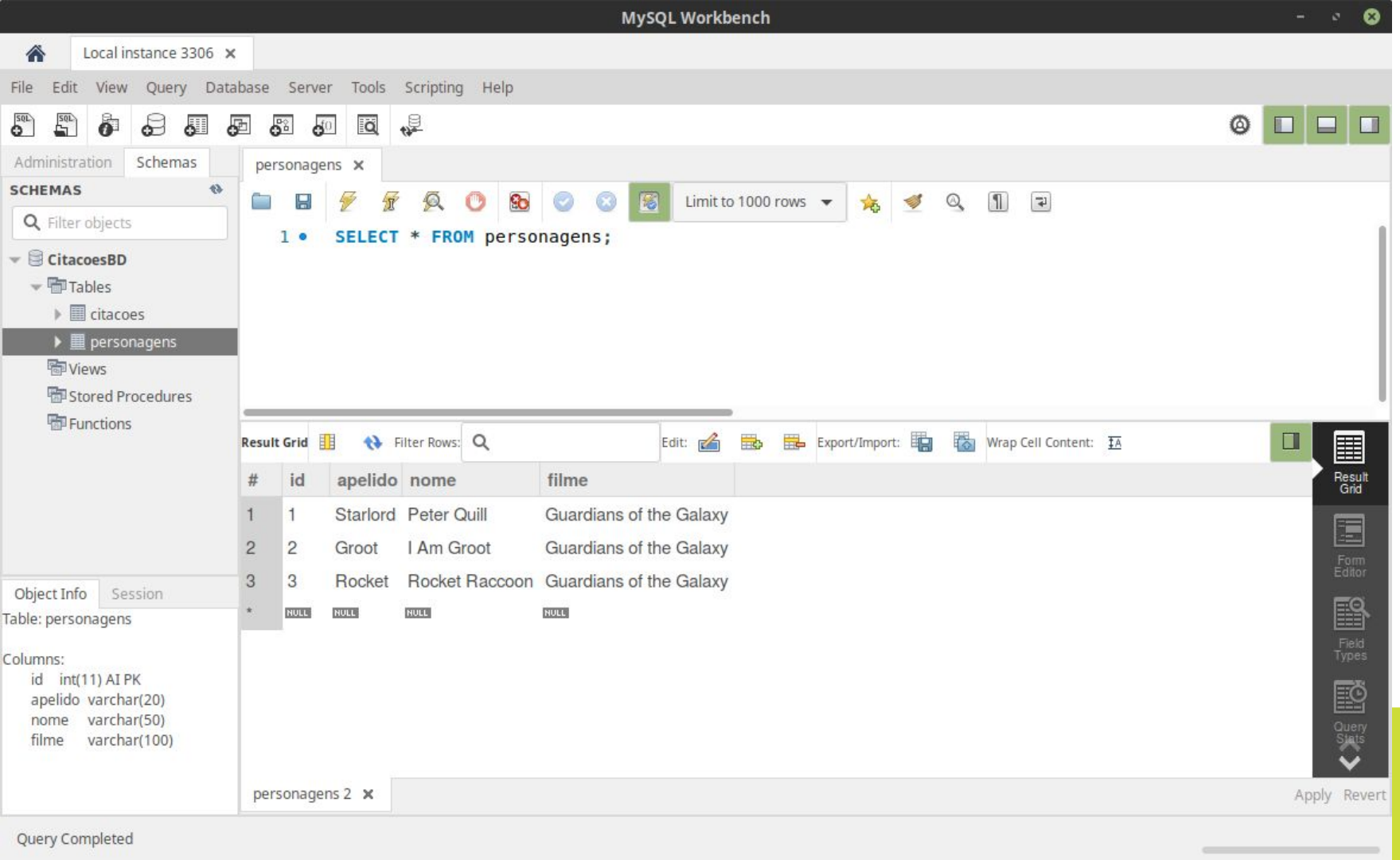
Usando Interface Gráfica

- Existem diversos aplicativos gráficos que facilitam o acesso e gerenciamento do banco de dados usando janelas
  - Auxiliam na conexão ao BD, execução de comandos e visualização dos resultados
  - Em alguns casos, possuem também suporte ao projeto das tabelas do BD

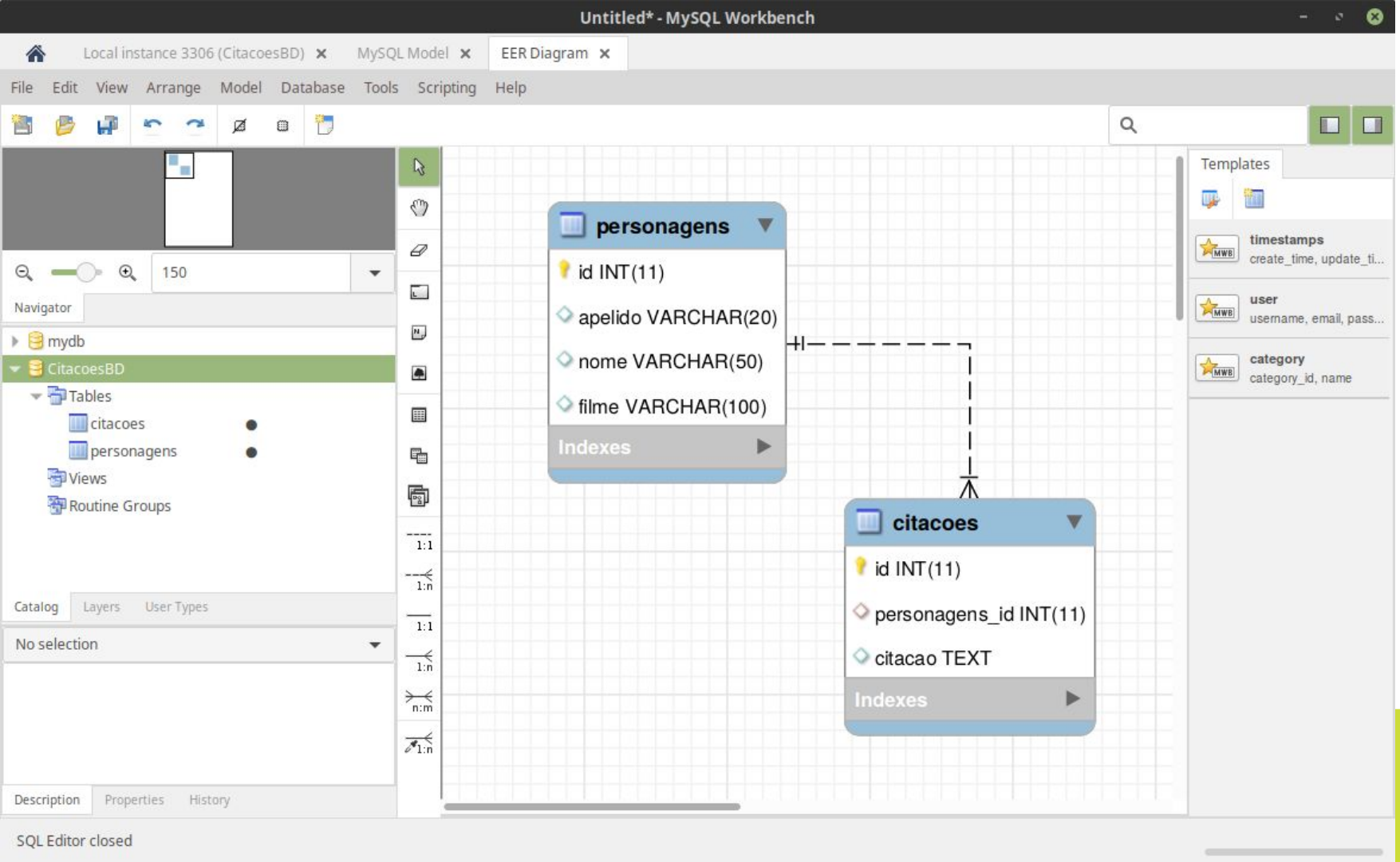
- MySQL Workbench
  - Um dos mais utilizados para gerenciamento do MySQL
  - Precisa ser instalado à parte: https://dev.mysql.com/downloads/workbench/
  - Será necessário criar um novo usuário no MySQL, usando a linha de comando, para poder usar o Workbench. Explicado nos slides anteriores



MySQL Workbench



MySQL Workbench



MySQL Workbench

MySQL Workbench

- Apesar de nos próximos slides nós não usarmos mais o MySQL Workbench, o seu uso é sim **recomendado**
  - Todos os comandos mostrados nos slides seguintes podem ser executados no MySQL Workbench

Criando uma Nova Tabela

- Uma tabela possui:
  - Um nome
  - Descrição das Colunas
    - *Cada coluna possui um nome e o tipo de dados dele*
  - Linhas
    - *São os dados realmente armazenados*
    - *Cada linha possui as colunas descritas pela tabela*

Criando uma Nova Tabela

- Criando uma tabela chamada “personagens”

```
mysql> CREATE TABLE personagens (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    apelido VARCHAR(20),
    nome VARCHAR(50),
    filme VARCHAR(100)
);
Query OK, 0 rows affected (0,01 sec)
```

- Listando as tabelas criadas no BD atual

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_TesteBD |
+------------------+
| personagens       |
+------------------+
1 row in set (0,00 sec)
```

Criando uma Nova Tabela

- Mostrando a descrição de uma tabela

```
mysql> DESCRIBE personagens;
+---------+--------------+------+-----+---------+-------------+
| Field   | Type         | Null | Key | Default | Extra       |
+---------+--------------+------+-----+---------+-------------+
| id      | int(11)      | NO   | PRI | NULL    | auto_increm |
| apelido | varchar(20)  | YES  |     | NULL    |             |
| nome    | varchar(50)  | YES  |     | NULL    |             |
| filme   | varchar(100) | YES  |     | NULL    |             |
+---------+--------------+------+-----+---------+-------------+
4 rows in set (0,00 sec)
```

Inserindo Dados na Tabela

- Quando uma tabela é criada, ela não possui nenhum dado armazenado
  - Não possui nenhuma “linha”

- Para armazenar dados na tabela:

```
mysql> INSERT INTO personagens VALUES (NULL, "Starlord", "Peter Quill",
"Guardians of the Galaxy");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO personagens VALUES (NULL, "Groot", "I Am Groot",
"Guardians of the Galaxy");
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO personagens VALUES (NULL, "Rocket", "Rocket
Raccoon", "Guardians of the Galaxy");
Query OK, 1 row affected (0.00 sec)
```

Realizando Consultas

- Provavelmente o comando mais útil de um SGBD é o comando para realizar consultas
  - Comando `SELECT`
- Uma consulta retorna uma tabela virtual com as colunas consultadas e os dados retornados (linhas)
- Em uma consulta, é possível:
  - Indicar as colunas que se quer acessar
  - Filtrar os dados
  - Ordenar os dados (por qualquer coluna)
  - Relacionar dados de tabelas diferentes
  - Agrupar informações
  - Dentre diversas outras possibilidades

Realizando Consultas

- A consulta mais simples que tem é:

```
mysql> SELECT * FROM personagens;
+----+----------+----------------+-------------------------+
| id | apelido  | nome           | filme                   |
+----+----------+----------------+-------------------------+
|  1 | Starlord | Peter Quill    | Guardians of the Galaxy |
|  2 | Groot    | I Am Groot     | Guardians of the Galaxy |
|  3 | Rocket   | Rocket Raccoon | Guardians of the Galaxy |
+----+----------+----------------+-------------------------+
3 rows in set (0,00 sec)
```

  - Nesta consulta, estamos “selecionando” todas as colunas (por isso o asterisco) da tabela personagens
    - *Note como os valores da coluna id foram incrementados*

Realizando Consultas

- Listando as colunas que queremos retornar

```
mysql> SELECT apelido, nome FROM personagens;
+----------+----------------+
| apelido  | nome           |
+----------+----------------+
| Starlord | Peter Quill    |
| Groot    | I Am Groot     |
| Rocket   | Rocket Raccoon |
+----------+----------------+
3 rows in set (0,00 sec)
```

  - Nesta consulta, estamos “selecionando” as colunas `apelido` e `nome` da tabela `personagens`

Realizando Consultas

- Ordenando as linhas (por alguma coluna)

```
mysql> SELECT apelido, nome FROM personagens ORDER BY nome;
+----------+----------------+
| apelido  | nome           |
+----------+----------------+
| Groot    | I Am Groot     |
| Starlord | Peter Quill    |
| Rocket   | Rocket Raccoon |
+----------+----------------+
3 rows in set (0,00 sec)
```

  - Nesta consulta, estamos “selecionando” as colunas `apelido` e `nome` da tabela `personagens` e ordenando o resultado pela coluna `nome`

Realizando Consultas

- Ordenando as linhas de forma decrescente

```
mysql> SELECT apelido, nome FROM personagens ORDER BY nome DESC;
+----------+----------------+
| apelido  | nome           |
+----------+----------------+
| Rocket   | Rocket Raccoon |
| Starlord | Peter Quill    |
| Groot    | I Am Groot     |
+----------+----------------+
3 rows in set (0,00 sec)
```

  - Nesta consulta, estamos “selecionando” as colunas `apelido` e `nome` da tabela `personagens` e ordenando (de forma decrescente) o resultado pela coluna `nome`

Realizando Consultas

- Filtrando os dados retornados

```
mysql> SELECT * FROM personagens WHERE apelido='groot';
+----+---------+------------+-------------------------+
| id | apelido | nome       | filme                   |
+----+---------+------------+-------------------------+
|  2 | Groot   | I Am Groot | Guardians of the Galaxy |
+----+---------+------------+-------------------------+
1 row in set (0,00 sec)
```

  - Nesta consulta, estamos retornando apenas as linhas da tabela `personagens` “onde” a coluna `apelido` possui valor “`groot`”

Realizando Consultas

- Filtrando os dados retornados (por mais de uma coluna)

```
mysql> SELECT * FROM personagens WHERE apelido='groot' AND
                                       nome='I Am Groot';
+----+---------+------------+--------------------------+
| id | apelido | nome       | filme                    |
+----+---------+------------+--------------------------+
|  2 | Groot   | I Am Groot | Guardians of the Galaxy |
+----+---------+------------+--------------------------+
1 row in set (0,00 sec)
```

Relacionando Tabelas

- Ao criar uma tabela, é possível definir “restrições” nos dados que a tabela pode armazenar
  - Por exemplo, o “`primary key`” que estamos usando, indica que aquela coluna é a chave primária
    - *chave que identifica unicamente cada uma das linhas*
  - Restrições da chave primária
    - *não pode ser nulo (o `auto_increment` garante isso)*
    - *não pode ter linhas com valores repetidos (o `auto_increment` garante isso)*
- Uma outra possibilidade é poder relacionar uma coluna de uma tabela com a chave primária (coluna) de outra tabela
  - Isso é conhecido como “chave estrangeira”
  - Permite “conectar” duas tabelas relacionadas por uma coluna em comum

Relacionando Tabelas

- Vamos criar uma tabela contendo “citações” dos personagens

```
mysql> CREATE TABLE citacoes (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    personagens_id INT,
    citacao TEXT,
    FOREIGN KEY (personagens_id) REFERENCES personagens(id)
);
Query OK, 0 rows affected (0,01 sec)
```

  - Como indicado na última parte do comando, a coluna `personagens_id` é uma chave estrangeira da coluna `id` na tabela `personagens`
    - *Isso significa que somente poderemos inserir citações nesta tabela de personagens que já existem na tabela personagens*

Relacionando Tabelas - Exemplo

- Vamos popular a tabela com algumas frases

```
mysql> INSERT INTO citacoes VALUES (NULL, 1, "That's a fake laugh.");
Query OK, 1 row affected (0,00 sec)

mysql> INSERT INTO citacoes VALUES (NULL, 1, "I have part of a plan.");
Query OK, 1 row affected (0,00 sec)

mysql> INSERT INTO citacoes VALUES (NULL, 1, "Dude, just chill out!");
Query OK, 1 row affected (0,00 sec)

mysql> INSERT INTO citacoes VALUES (NULL, 2, "I am Groot.");
Query OK, 1 row affected (0,00 sec)

mysql> INSERT INTO citacoes VALUES (NULL, 2, "We are Groot.");
Query OK, 1 row affected (0,00 sec)
```

Relacionando Tabelas - Exemplo

- Não é possível inserir uma frase de um personagem que não existe
  - Neste caso, o `id` 4 não existe na tabela `personagens`

```
mysql> INSERT INTO citacoes VALUES (NULL, 4, "Nothing goes over my head!");
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint
fails (`CitacoesBD`.`citacoes`, CONSTRAINT `citacoes_ibfk_1` FOREIGN KEY
(`personagens_id`) REFERENCES `personagens` (`id`))
```

Relacionando Tabelas - Exemplo

- Listando todos os dados da tabela `citacoes`

```
mysql> SELECT * FROM citacoes;
+----+----------------+------------------------+
| id | personagens_id | citacao                |
+----+----------------+------------------------+
|  1 |              1 | That's a fake laugh.   |
|  2 |              1 | I have part of a plan. |
|  3 |              1 | Dude, just chill out!  |
|  4 |              2 | I am Groot.            |
|  5 |              2 | We are Groot.          |
+----+----------------+------------------------+
5 rows in set (0,00 sec)
```

Consultas em Tabelas Relacionadas

- O comando `SELECT` permite pegar colunas de diferentes tabelas e relacioná-las com o parâmetro `WHERE`

```
mysql> SELECT * FROM personagens, citacoes WHERE citacoes.personagens_id=personagens.id;

+----+----------+-------------+-------------------------+----+----------------+-----------------------+
| id | apelido  | nome        | filme                   | id | personagens_id | citacao               |
+----+----------+-------------+-------------------------+----+----------------+-----------------------+
|  1 | Starlord | Peter Quill | Guardians of the Galaxy |  1 |              1 | That's a fake laugh.  |
|  1 | Starlord | Peter Quill | Guardians of the Galaxy |  2 |              1 | I have part of a plan. |
|  1 | Starlord | Peter Quill | Guardians of the Galaxy |  3 |              1 | Dude, just chill out! |
|  2 | Groot    | I Am Groot  | Guardians of the Galaxy |  4 |              2 | I am Groot.           |
|  2 | Groot    | I Am Groot  | Guardians of the Galaxy |  5 |              2 | We are Groot.         |
+----+----------+-------------+-------------------------+----+----------------+-----------------------+
5 rows in set (0,00 sec)
```

Consultas em Tabelas Relacionadas

- No exemplo anterior, note como as colunas da tabela `personagens` foram combinadas com as colunas da tabela `citacoes` usando as colunas:
  - `id`, da tabela `personagens`, e
  - `personagens_id`, da tabela `citacoes`,
  - como “ligação” entre as duas tabelas
- Como `Starlord` possui três citações, seus valores na tabela `personagens` foram repetidos três vezes

Consultas em Tabelas Relacionadas

- Podemos combinar isso com tudo que já vimos:

```
mysql> SELECT personagens.apelido, citacoes.citacao
              FROM personagens, citacoes
              WHERE citacoes.personagens_id=personagens.id;
+----------+------------------------+
| apelido  | citacao                |
+----------+------------------------+
| Starlord | That's a fake laugh.   |
| Starlord | I have part of a plan. |
| Starlord | Dude, just chill out!  |
| Groot    | I am Groot.            |
| Groot    | We are Groot.          |
+----------+------------------------+
5 rows in set (0,00 sec)
```

  - Aqui, ao invés de mostrar todas as colunas, estamos mostrando apenas a coluna `apelido` da tabela `personages` e a coluna `citacao` da tabela `citacoes`.

Consultas em Tabelas Relacionadas

- Podemos combinar isso com tudo que já vimos (2):

```
mysql> SELECT citacoes.citacao
              FROM personagens, citacoes
              WHERE citacoes.personagens_id=personagens.id AND
                    apelido="StarLord";
+------------------------+
| citacao                |
+------------------------+
| That's a fake laugh.   |
| I have part of a plan. |
| Dude, just chill out!  |
+------------------------+
3 rows in set (0,00 sec)
```

  - Nesta consulta, estamos mostrando todas as citações do personagem `Starlord`

Consultas em Tabelas Relacionadas

- Podemos combinar isso com tudo que já vimos (3):

```
mysql> SELECT citacoes.citacao
              FROM personagens, citacoes
              WHERE citacoes.personagens_id=personagens.id AND
                    apelido="StarLord"
              ORDER BY citacoes.citacao;
+-------------------------+
| citacao                 |
+-------------------------+
| Dude, just chill out!   |
| I have part of a plan.  |
| That's a fake laugh.    |
+-------------------------+
3 rows in set (0,00 sec)
```

  - Nesta consulta, estamos mostrando todas as citações do personagem `Starlord` ordenado alfabeticamente

Modificando Dados de uma Linha

- Para alterar uma linha de uma tabela

```
mysql> UPDATE personagens SET apelido='Star-Lord' WHERE id=1;
Query OK, 1 row affected (0,00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * FROM personagens;
+----+-----------+----------------+-------------------------+
| id | apelido   | nome           | filme                   |
+----+-----------+----------------+-------------------------+
|  1 | Star-Lord | Peter Quill    | Guardians of the Galaxy |
|  2 | Groot     | I Am Groot     | Guardians of the Galaxy |
|  3 | Rocket    | Rocket Raccoon | Guardians of the Galaxy |
+----+-----------+----------------+-------------------------+
3 rows in set (0,00 sec)
```

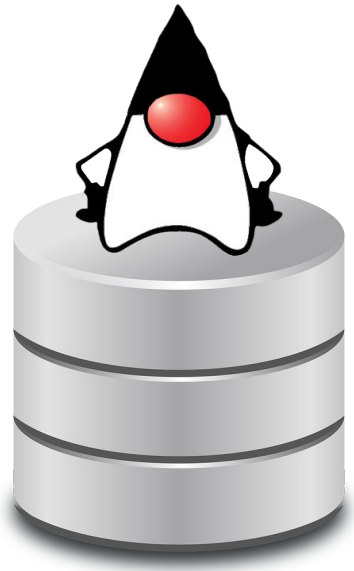
Removendo Linhas de uma Tabela

- Para remover uma ou mais linhas de uma tabela

```
mysql> DELETE FROM citacoes WHERE id=5;
Query OK, 1 row affected (0,00 sec)
```

```
mysql> SELECT * FROM citacoes;
+----+----------------+-------------------------+
| id | personagens_id | citacao                 |
+----+----------------+-------------------------+
|  1 |              1 | That's a fake laugh.    |
|  2 |              1 | I have part of a plan.  |
|  3 |              1 | Dude, just chill out!   |
|  4 |              2 | I am Groot.             |
+----+----------------+-------------------------+
4 rows in set (0,00 sec)
```

Universidade Federal do Amazonas
Instituto de Computação
Projeto de Programas
Técnicas Avançadas de Programação



JDBC: Java Database Connectivity


Horácio Fernandes
horacio@icomp.ufam.edu.br

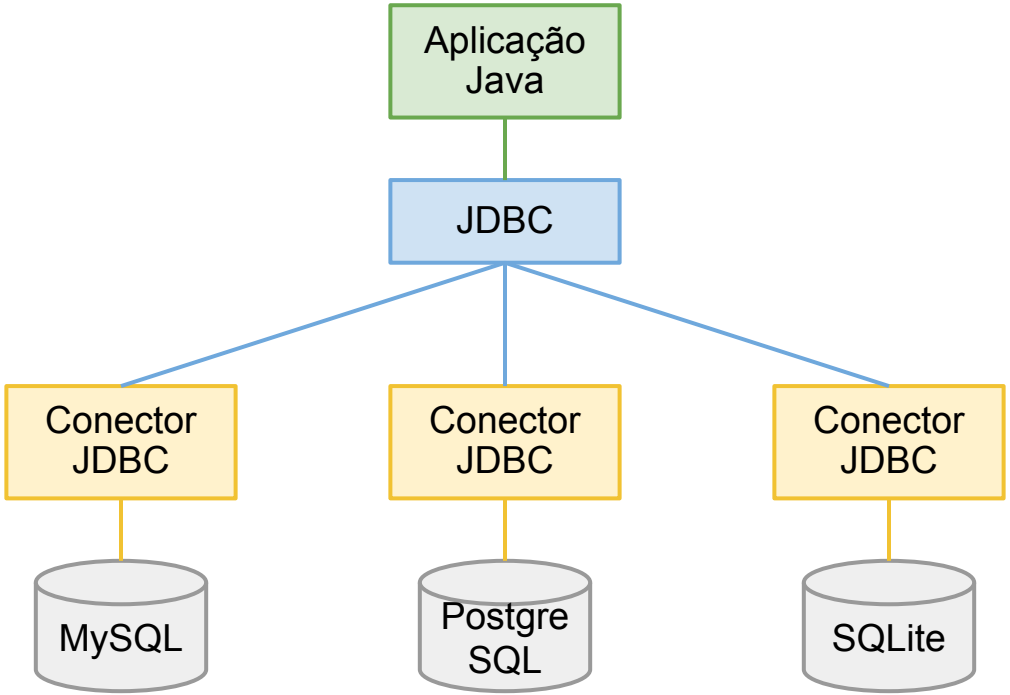
ColabWeb: bit.ly/pp-colabweb

Introdução

- O JDBC é a biblioteca padrão do Java para acesso ao BD
  - Permite manipular, de forma padronizada
    - *Conectar*
    - *Executar comandos e consultas*
    - *Receber os resultados*
  - Qualquer banco de dados:
    - *MySQL / MariaDB*
    - *PostgreSQL*
    - *SQLite*
    - *Oracle Database*
    - *IBM DB2*
    - *MS SQL Server*
    - *Outros*

Introdução

- O JDBC forma uma camada entre a aplicação e o BD



Conector JDBC

- O *Conector JDBC* é como se fosse o driver para acessar o BD usando o JDBC

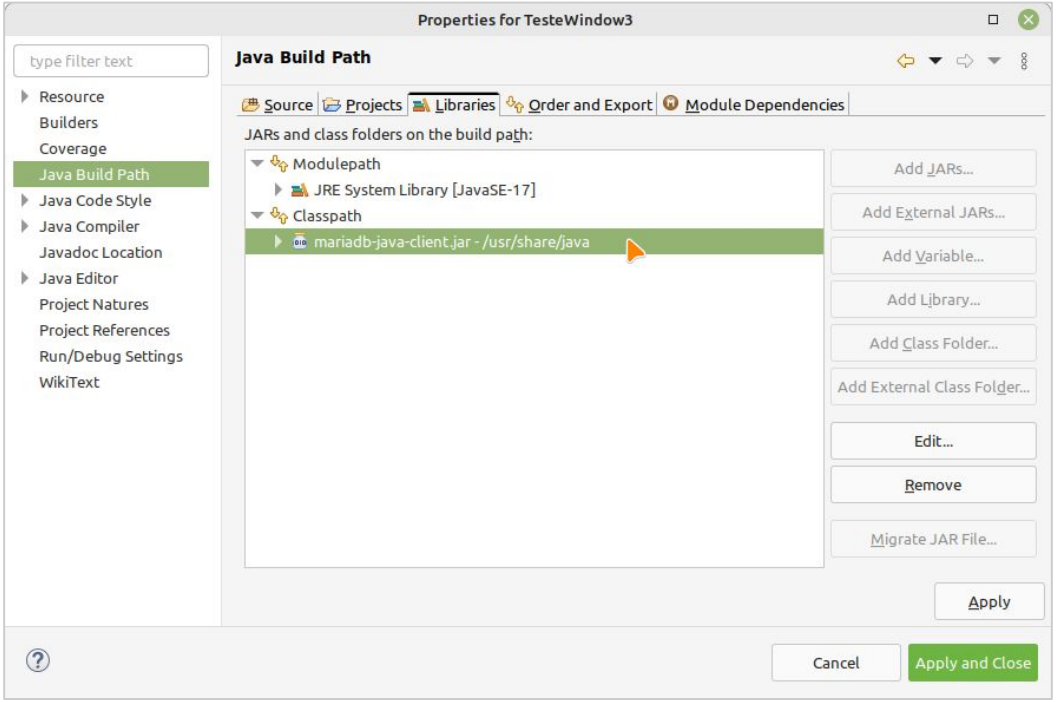
- Cada SGBD tem o seu conector JDBC
  - O conector é distribuído e mantido pelos desenvolvedores do próprio SGBD
  - Ele não vem com o JRE ou JDK
    - *Precisa ser instalado à parte*

Conector JDBC

- Para instalar o Conector JDBC do MySQL
  - No Ubuntu:
    - `sudo apt install libmariadb-java`
    - *O conector (arquivo .jar) ficará em /usr/share/java/mariadb-java-client.jar*
  - Em outros sistemas
    - *http://dev.mysql.com/downloads/connector/j*
    - *Baixe e descompacte o arquivo mysql-connector-java-a.b.c.zip*
    - *O arquivo .jar deverá ser colocado em uma pasta de bibliotecas do java*
    - *E a variável CLASSPATH do sistema deverá ser setado para incluir o .jar*

Conector JDBC

- No Eclipse, é necessário configurar o projeto para usar o JAR
  - Project → Properties → Java Build Path → Libraries → Add External JARs...



DAO: *Data Access Objects*

- A forma mais organizada de se utilizar BD em seus sistemas é separando as classes normais da sua aplicação das classes que acessam o BD. Essas últimas serão as *Data Access Objects*

- *Data Access Objects* (DAO):
  - Classe intermediária que permite salvar e recuperar (persistir) os dados (atributos) de uma classe em uma tabela no banco de dados
  - A ideia é que cada tabela do BD tenha uma classe DAO no sistema
  - A classe DAO terá comandos para acesso ao BD
    - *enquanto que as outras classes do sistema não terão*
    - *mas poderão instanciar objetos das classes DAO*

DAO: Exemplo

- Exemplo:
  - Classe `Personagem`
    - *Classe normal, representa um ”personagem” no sistema*
    - *Terá uma lista da classe Citacao (abaixo) com as citações do personagem*
  - Classe `Citacao`
    - *Classe normal, representa uma citação de um personagem no sistema*
    - *Que é basicamente uma String (nesta modelagem)*
  - Classe `BancoDeDados`
    - *Contêm atributos e métodos para acessar o banco de dados*
  - Classe `PersonagemDAO extends BancoDeDados`
    - *Classe intermediária entre a tabela personagens e a classe Personagem*
  - Classe `CitacaoDAO extends BancoDeDados`
    - *Classe intermediária entre a tabela citacoes e a classe Citacao*

Classe Personagem

```
public class Personagem {
  private int id;
  private String apelido;
  private String nome;
  private String filme;
  private Citacao[] citacoes;

  public Personagem(String apelido, String nome, String filme) {
    this.apelido = apelido;
    this.nome = nome;
    this.filme = filme;
  }

  // Getters e Setters ...
}
```

Classe `Citacao`

```
public class Citacao {
  private int id;
  private Personagem personagem;
  private String citacao;

  public Citacao(Personagem personagem, String citacao) {
    this.personagem = personagem;
    this.citacao = citacao;
  }

  // Getters e Setters ...
}
```

Classe BancoDeDados

```java
import java.sql.*;

public class BancoDeDados {
  private static String url = "jdbc:mysql://localhost:3306/CitacoesBD";
  private static String user = "citacoes_admin";
  private static String pass = "Teste123";
  protected static Connection conexao = null;

  public BancoDeDados() {
    if (conexao == null) conecta();
  }

  private static boolean conecta() {
    try {
      conexao = DriverManager.getConnection(url, user, pass);
      return true;
    } catch (SQLException e) { return false; }
  }

  public static boolean desconecta() {
    try {
      conexao.close();
      return true;
    } catch (SQLException e) { return false; }
  }
}
```

Classe PersonagemDAO

```
import java.sql.*;

public class PersonagemDAO extends BancoDeDados {

  public void listarPersonagens() {
    try {
      Statement st = conexao.createStatement();
      ResultSet rs = st.executeQuery("SELECT * FROM personagens");
      while (rs.next()) {
        System.out.println("Personagem " + rs.getString(2) +
                           " (" + rs.getString(3) + ")" +
                           " do filme " + rs.getString(4));
      }
    }
    catch (SQLException e) { }
  }

  public static void main(String args[]) {
    PersonagemDAO personagemDAO = new PersonagemDAO();
    personagemDAO.listarPersonagens();
  }
}
```

```
$ java PersonagemDAO
Personagem Star-Lord (Peter Quill) do filme Guardians of the Galaxy
Personagem Groot (I Am Groot) do filme Guardians of the Galaxy
Personagem Rocket (Rocket Raccoon) do filme Guardians of the Galaxy
```

Classe PersonagemDAO

```
  // Classe PersonagemDAO (continuacao)

  public boolean adicionarPersonagem(Personagem p) {
    try {
      Statement st = conexao.createStatement();
      st.executeUpdate("INSERT INTO personagens VALUES (NULL, '"
                       + p.getApelido() + "'," + " '" + p.getNome()
                       + "', '" + p.getFilme() + "')");
      return true;
    } catch (SQLException e) { return false; }
  }

  public static void main(String args[]) {
    PersonagemDAO personagemDAO = new PersonagemDAO();
    Personagem personagem = new Personagem("Drax", "Drax the Destroyer",
                                           "Guardians of the Galaxy");
    personagemDAO.adicionarPersonagem(personagem);
    personagemDAO.listarPersonagens();
  }
}
```

```
$ java PersonagemDAO
Personagem Star-Lord (Peter Quill) do filme Guardians of the Galaxy
Personagem Groot (I Am Groot) do filme Guardians of the Galaxy
Personagem Rocket (Rocket Raccoon) do filme Guardians of the Galaxy
Personagem Drax (Drax the Destroyer) do filme Guardians of the Galaxy
```

Classe PersonagemDAO

```
  // Classe PersonagemDAO (continuacao)

  public Personagem getPersonagem(String apelido) {
    try {
      Statement st = conexao.createStatement();
      ResultSet rs = st.executeQuery("SELECT * FROM personagens WHERE " +
                                     "apelido='" + apelido + "'");
      if (rs.next()) {
        return new Personagem(rs.getString(2), rs.getString(3),
                              rs.getString(4));
      }
      else return null;
    }
    catch (SQLException e) { return null; }
  }

  public static void main(String args[]) {
    PersonagemDAO personagemDAO = new PersonagemDAO();
    Personagem personagem = personagemDAO.getPersonagem("rocket");
    System.out.println(personagem.getNome());
  }
}
```

```
$ java PersonagemDAO
Rocket Raccoon
```

Finalização das Classes

- Vimos exemplos para executar consultas e outros comandos SQL
  - O que falta (métodos para modificar, remover) é apenas variação do “inserir”
- A classe `PersonagemDAO` pode ser continuada com os métodos
  - `public boolean atualizarPersonagem(Personagem p)`
  - `public boolean removerPersonagem(Personagem p)`
- A classe `CitacaoDAO` pode ser implementada com os métodos
  - `public void listarCitacoes(Personagem p)`
  - `public boolean adicionarCitacao(Personagem p, Citacao c)`
  - `public Citacao[] getCitacoes(Personagem p)`