MySQL
/
MariaDB

Horácio Fernandes
horacio@icomp.ufam.edu.br

Slides: http://tinyurl.com/slides-tp

MySQL / MariaDB
Sistema de Gerenciamento de Banco de Dados

- Sistema de Gerenciamento de Banco de Dados
  - SGBD
  - 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"*

MySQL / MariaDB
Sistema de Gerenciamento de Banco de Dados

- 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

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
Instalação

- Instalação do MySQL 5
  - Ubuntu: `sudo apt-get install mysql-server`
  - Outros sistemas: `http://www.mysql.com`
- Deverá pedir uma senha para o '`root`'
  - O '`root`' do MySQL não é o mesmo do Linux e não precisa (nem deve) ter a mesma senha
- Entrando no MySQL:

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.27-0ubuntu1 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' ...

mysql>
```

MySQL
Criando um novo Banco de Dados

- Um Banco de Dados (BD) é um conjunto de Tabelas, que irão armazenar os dados
  - Cada aplicação irá precisar de um BD
- Criando um novo BD

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

- Listando os BDs

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

MySQL
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
- Criando um novo usuário para acessar o BD

```
mysql> GRANT ALL PRIVILEGES ON CitacoesBD.*
       TO citacoes_admin@"localhost"
       IDENTIFIED BY 'Teste123' WITH GRANT OPTION;
Query OK, 0 rows affected (0,00 sec)
```

- Saindo do MySQL e entrando com o novo Usuário no BD

```
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
```

MySQL
Criando uma nova Tabela

- Uma tabela é composta por um conjunto de:
  - Colunas: são os tipos de informações que a tabela armazena
  - Linhas: são os dados armazenados
- 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

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

MySQL
Detalhando uma nova Tabela

- Para mostrar as informações 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)
```

MySQL
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)
```

MySQL
Realizando Consultas

- Provavelmente o comando mais útil de um SGBD é o comando para 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

MySQL
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)
```

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

MySQL
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)
```

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

MySQL
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)
```

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

MySQL
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)
```

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

MySQL
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)
```

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

MySQL
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)
```

MySQL
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*

MySQL
Relacionando Tabelas - Exemplo

- 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`*

MySQL
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)
```

  - Entretanto, a seguinte inserção dará erro, pois o `id` (4) não existe

```
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`))
```

MySQL
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)
```

MySQL
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)
```

MySQL
Consultas em Tabelas Relacionadas

- E 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)
```

MySQL
Consultas em Tabelas Relacionadas

- E 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)
```

MySQL
Consultas em Tabelas Relacionadas

- E 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)
```

MySQL
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
```

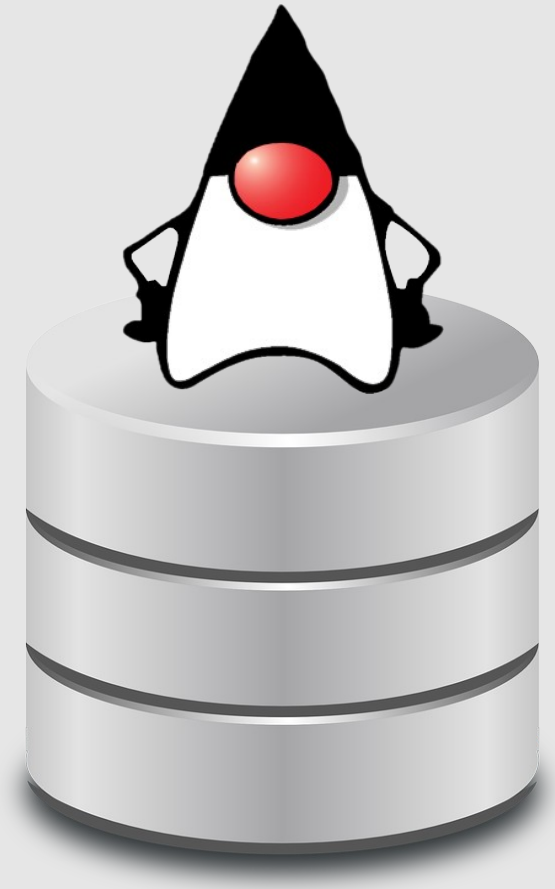
```
mysql> SELECT * FROM personagens;
+----+-----------+----------------+-------------------------+
| id | apelido   | nome           | filme                   |
+----+-----------+----------------+-------------------------+
|  1 | Star-Lord | Peter Quill    | Guardians of the Galaxy |
|  2 | Groot  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

MySQL
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)
```



JDBC: Java Database Connectivity

Horácio Fernandes
horacio@icomp.ufam.edu.br

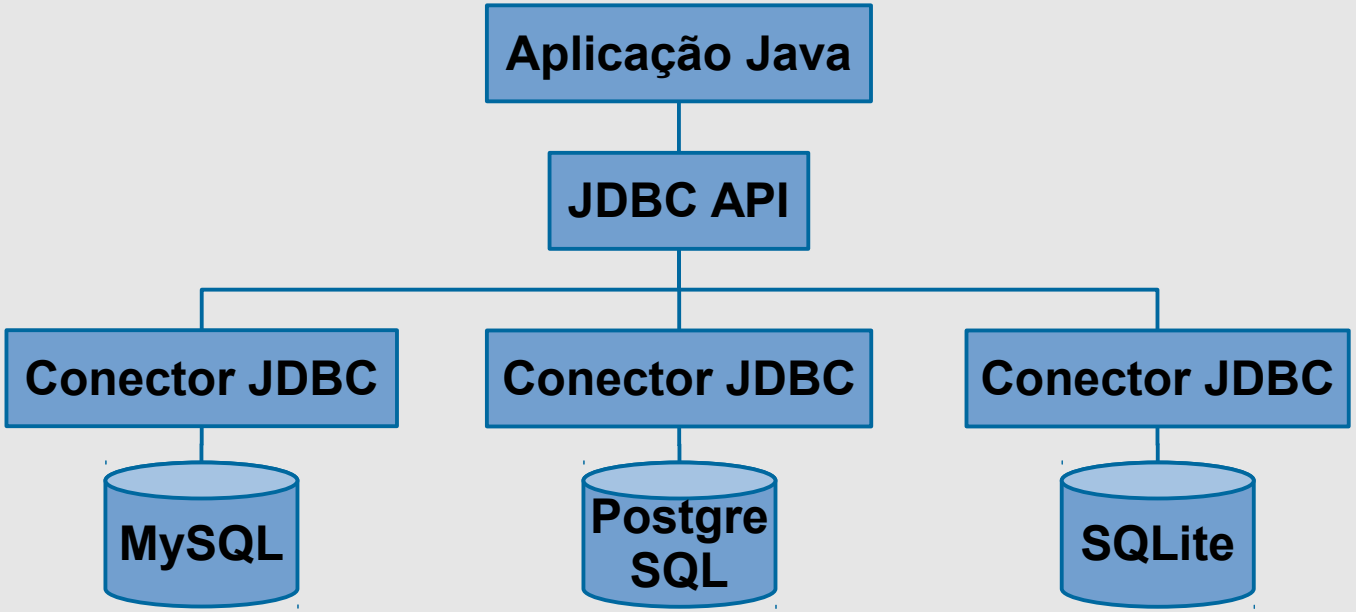
Slides: http://tinyurl.com/slides-tp

JDBC: *Java Database Connectivity*
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*

JDBC: *Java Database Connectivity*
Introdução

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



JDBC: *Java Database Connectivity*
Conector JDBC (MySQL)

- Para acessar o BD a partir do Java, você precisará do Conector JDBC respectivo
  - O conector é distribuído e mantido pelos desenvolvedores do próprio BD
- Para instalar o Conector JDBC do MySQL
  - No Ubuntu
    - `sudo apt-get install libmysql-java`
      - *O conector (arquivo `.jar`) ficará em `/usr/share/java/mysql.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*
  - No Eclipse, é necessário configurar o projeto para usar o JAR
    - *Project → Properties → Java Build Path → Libraries → Add External JARs...*

JDBC: *Java Database Connectivity*
DAO: ***Data Access Objects***

- *Data Access Objects* (DAO):
  - Classe intermediária que permite salvar e recuperar (persistência) 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*

JDBC: *Java Database Connectivity*
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`*

JDBC: *Java Database Connectivity*
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 ...
}
```

JDBC: *Java Database Connectivity*
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

```
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) {
            System.out.println(e.getMessage());
            return false;
        }
    }

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

Conecta ao BD

Fecha a conexão

```java
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();
  }

}
```

Cria um objeto para execução de comandos

Executa uma consulta SQL

Pula para a próxima linha do resultado. Retorna falso caso não exista mais linhas

```
$ 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
```

```java
  // 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) {
      System.out.println(e.getMessage());
      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();
  }
}
```

Executa um comando SQL

```
$ 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 (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
```

JDBC: *Java Database Connectivity*
Outros Métodos ...

- A classe `PersonagemDAO` pode ser continuada com os métodos
  - `public boolean atualizarPersonagem(int id, Personagem p)`
  - `public boolean removerPersonagem(int id, 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)`

JDBC: *Java Database Connectivity*
Conclusões

- Mesmo para quem nunca trabalhou com banco de dados, o uso dele é bem mais fácil do que utilizar arquivos texto para armazenar os dados