

# Análise de Dados

1.ª Aula Prática Laboratorial

Mestrado Integrado em Engenharia Informática

Ano Letivo 2019/2020

Marisa Esteves

27 de Setembro de 2019



**Universidade do Minho**

# Plano de Aula

1. Breve apresentação do plano de aula, do docente, dos objetivos das aulas práticas laboratoriais, bem como dos recursos necessários para as mesmas;
2. Contextualização prática sobre o MySQL;
3. Instalação do MySQL;
4. Resolução da 1.<sup>a</sup> ficha prática laboratorial pelos alunos em grupo;
5. Correção da ficha com os alunos.

# Docente

- Marisa Esteves responsável pelas aulas práticas laboratoriais – sextas-feiras das 10h às 12h (PL1);
- *E-mail* institucional: [marisa@di.uminho.pt](mailto:marisa@di.uminho.pt).

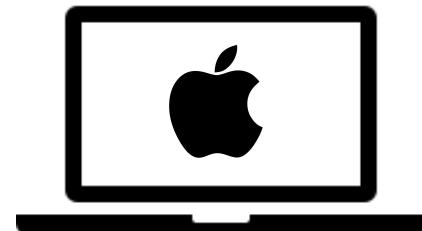
# Objetivos das Aulas Práticas Laboratoriais

- Resolução de fichas práticas laboratoriais em grupo;
- Apoio na realização do trabalho práctico final – últimas aulas do semestre letivo;
- Esclarecimento de dúvidas aos alunos.

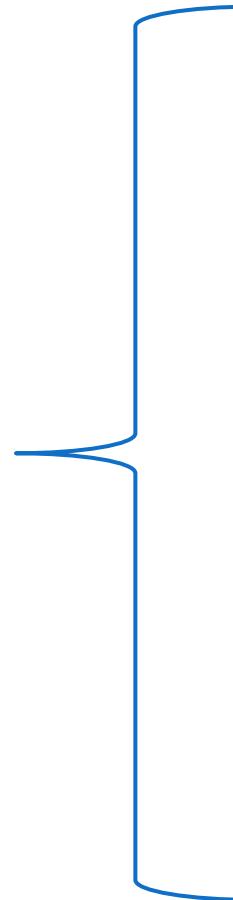
# Repositório

- <https://github.com/mesteves9/AD20192020>

# Recursos Necessários para as Aulas Práticas Laboratoriais (Opcional)



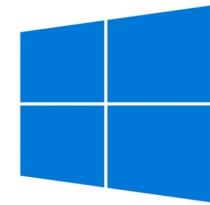
macOS



[tecnicos@di.uminho.pt](mailto:tecnicos@di.uminho.pt)

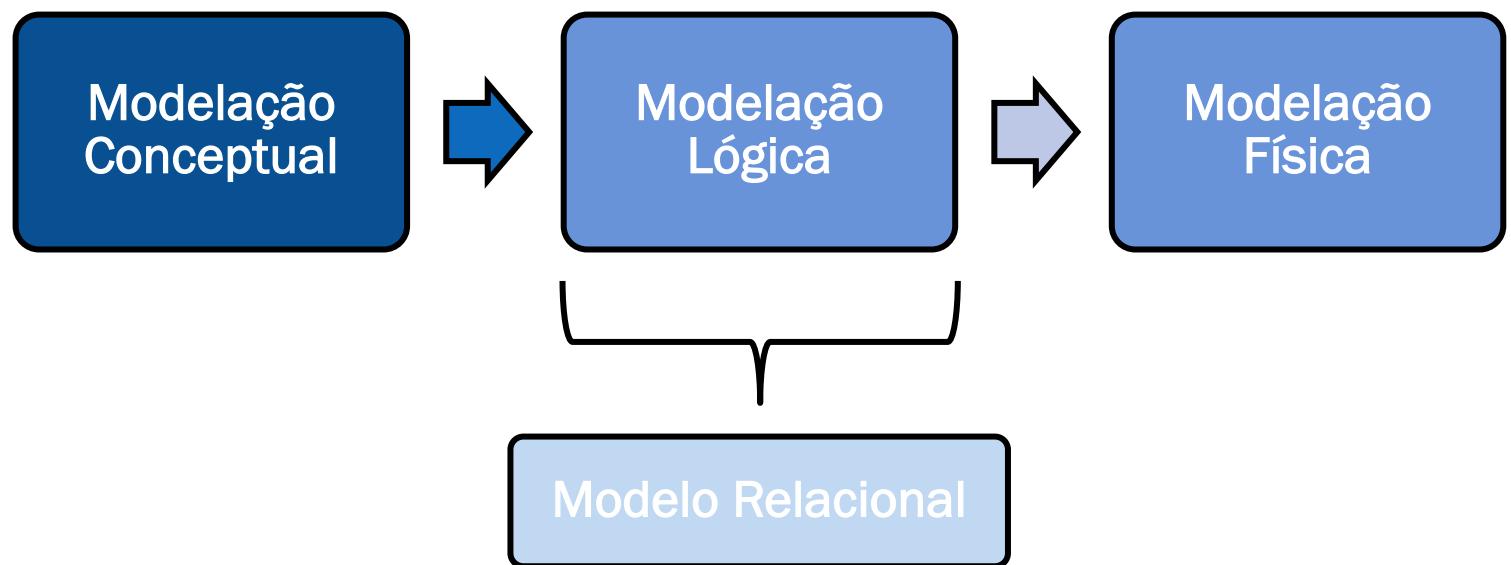


VMware Fusion



Windows 10

# Modelação Lógica em Sistemas de Bases de Dados



# Demonstração

*MySQL*

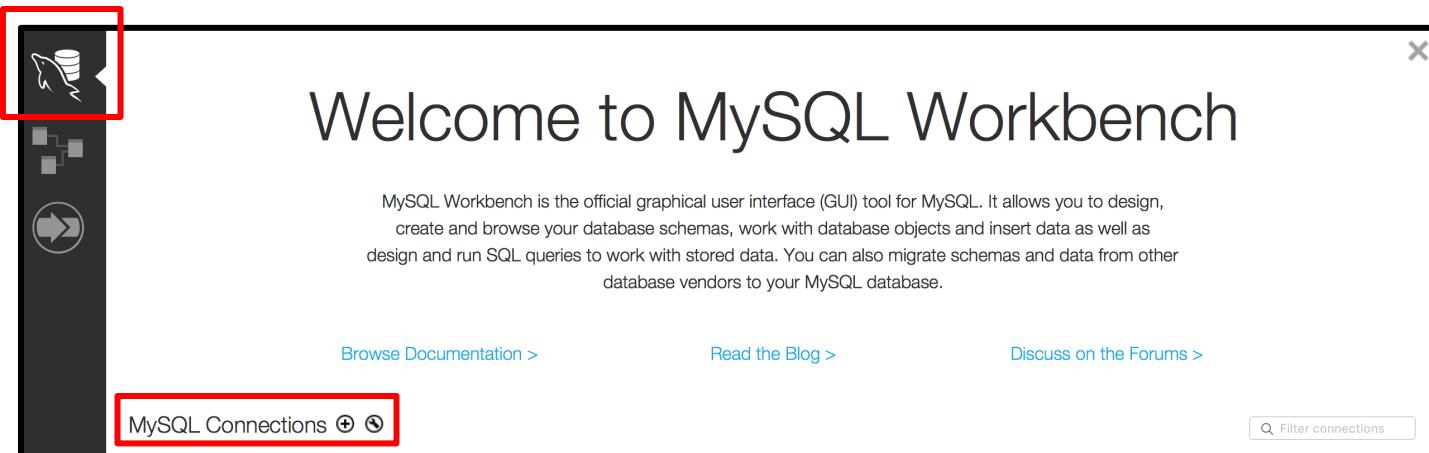


# Demonstração

## MySQL

- Criação de uma Conexão MySQL:

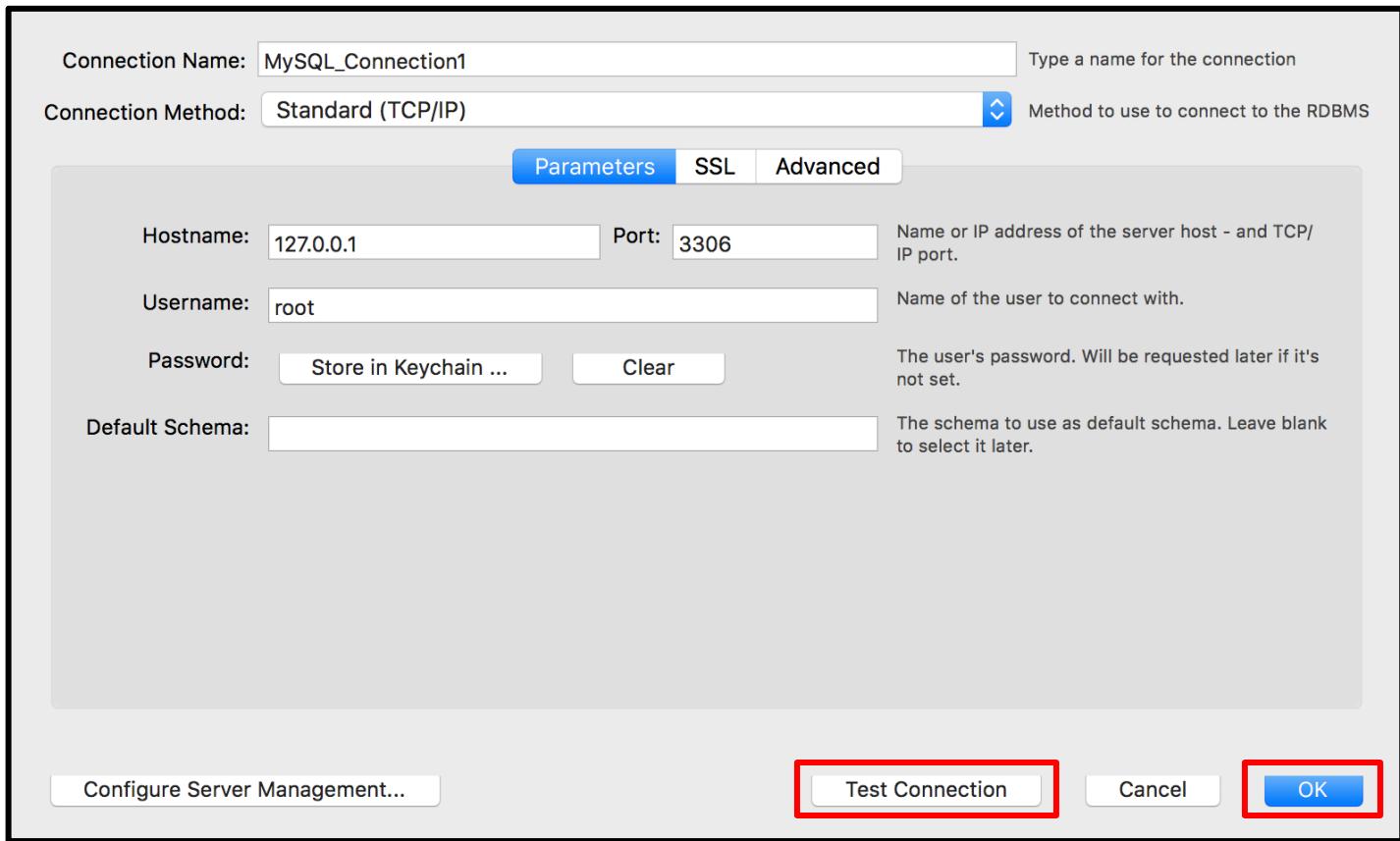
O MySQL Server tem de  
estar em modo “Start”!



# Demonstração

## MySQL

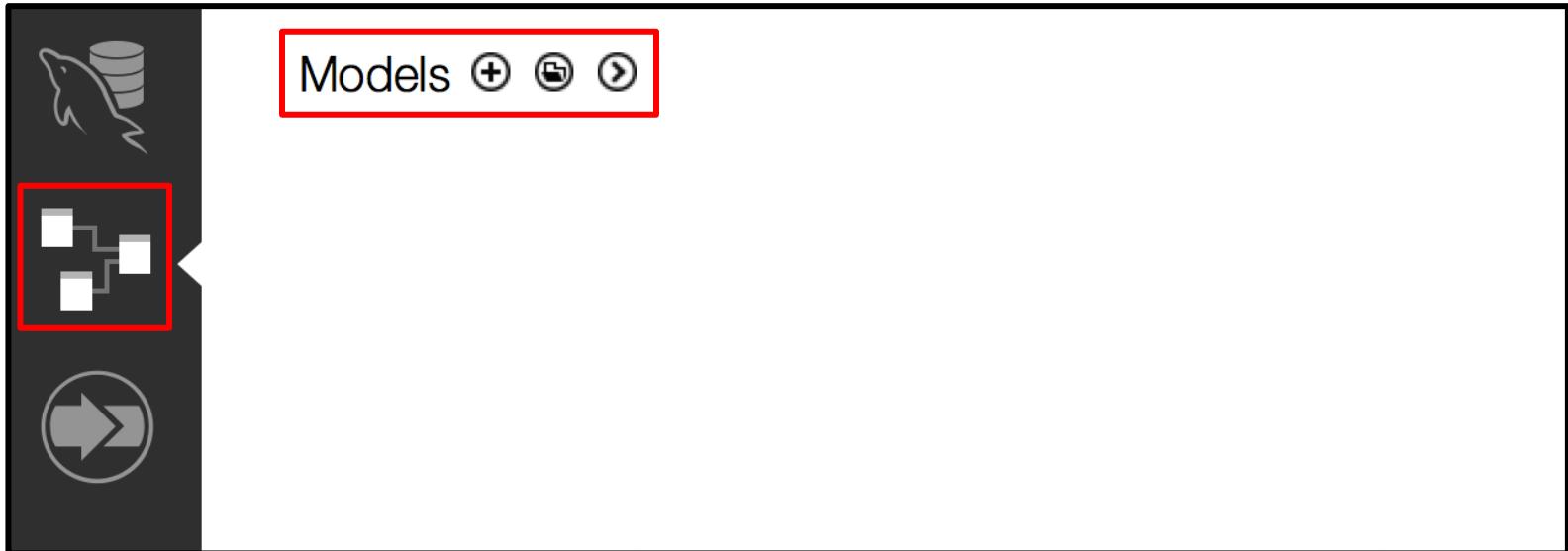
- Criação de uma Conexão MySQL:



# Demonstração

## *MySQL*

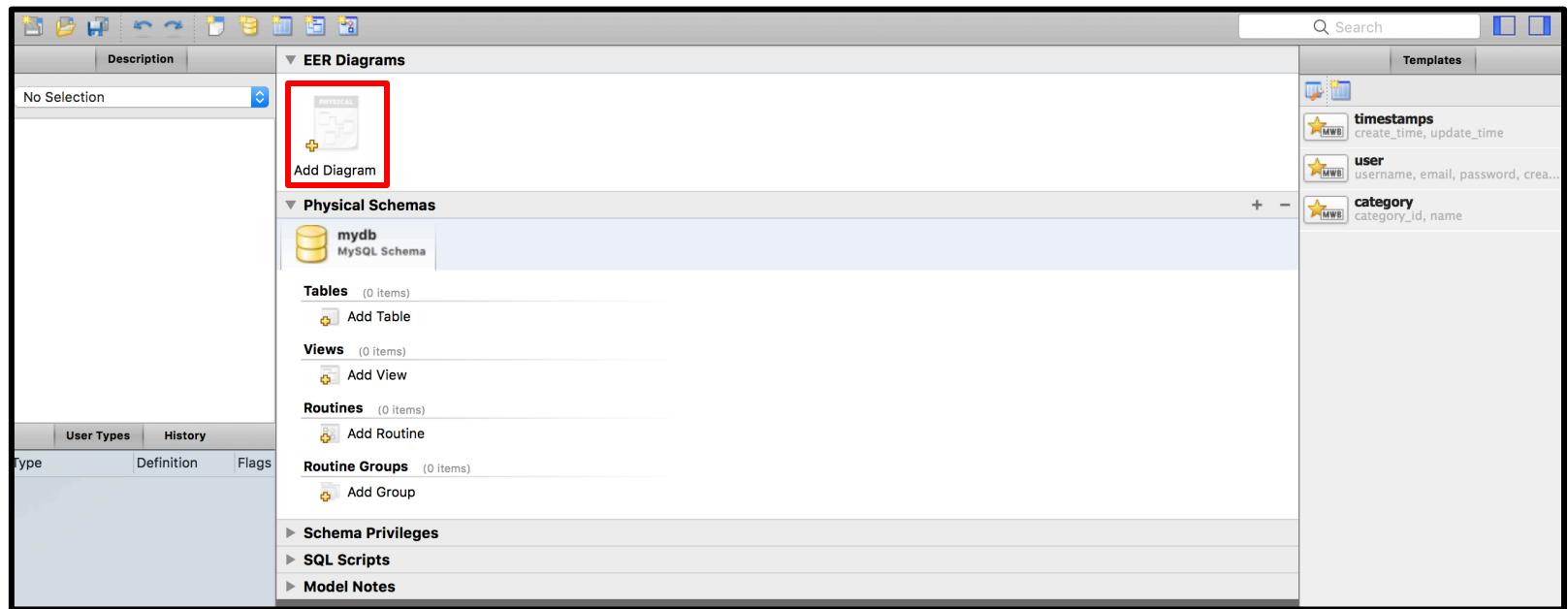
- Criação do Modelo Lógico (Modelo Relacional) no MySQL:



# Demonstração

## MySQL

- Criação do Modelo Lógico (Modelo Relacional) no MySQL:



# Demonstração

## MySQL

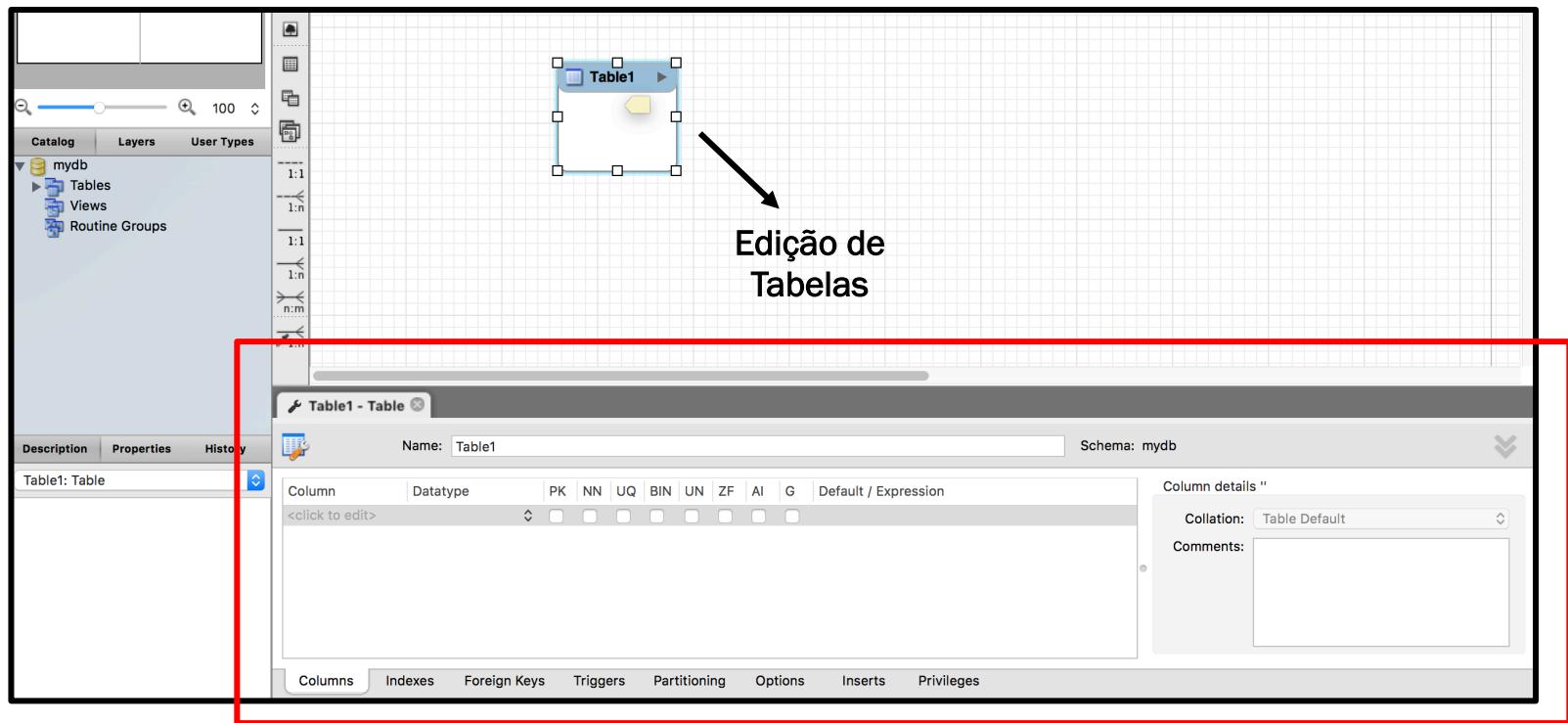
- Criação do Modelo Lógico (Modelo Relacional) no MySQL:



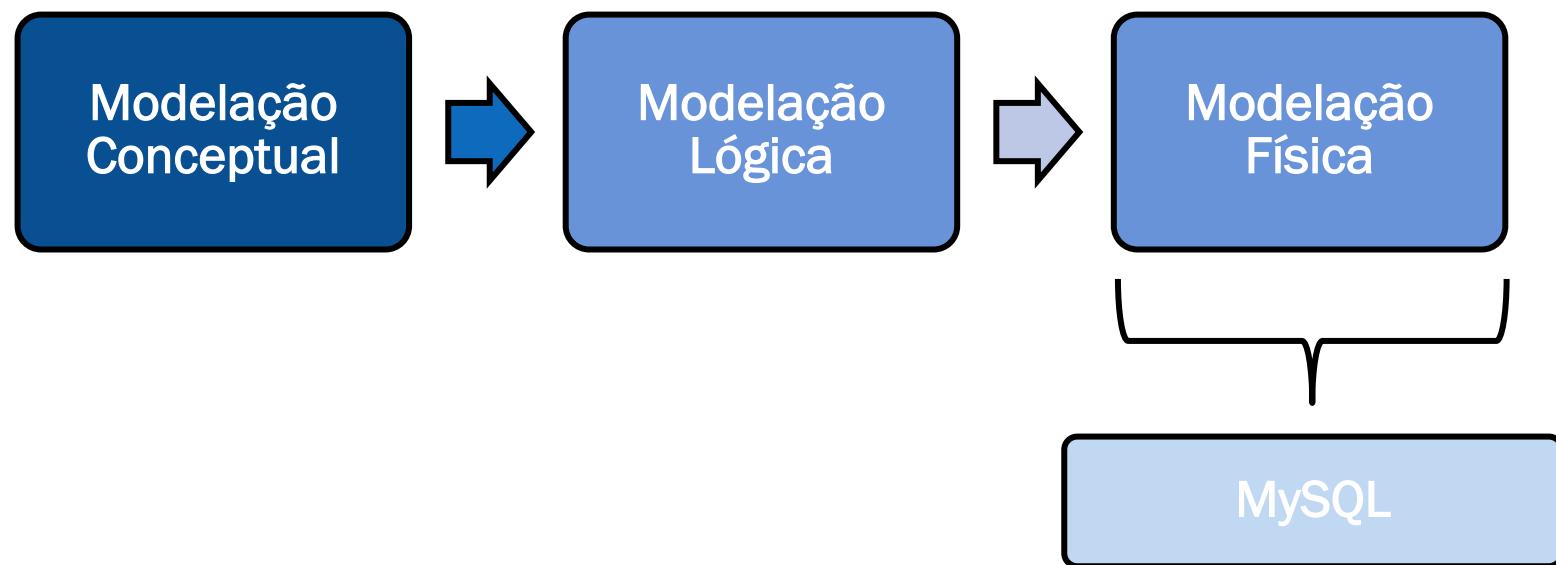
# Demonstração

## MySQL

- Criação do Modelo Lógico (Modelo Relacional) no MySQL:



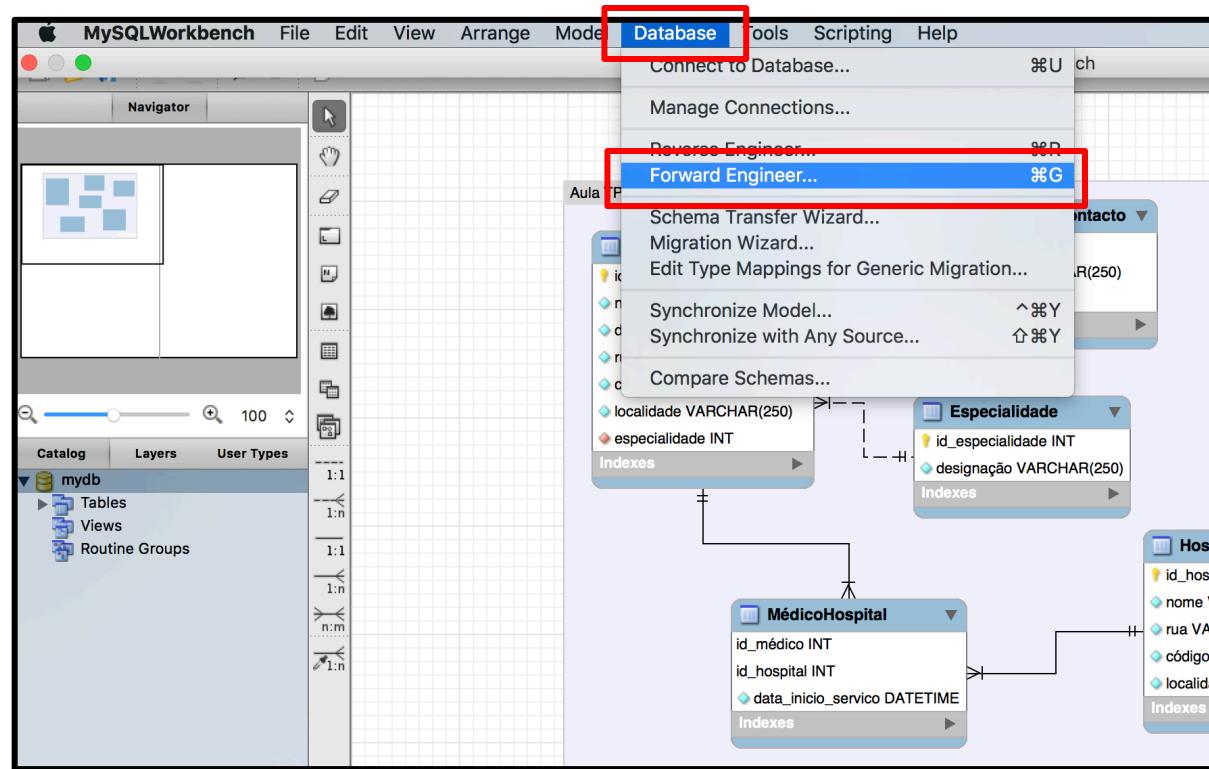
# Modelação Física em Sistemas de Bases de Dados



# Demonstração

## MySQL

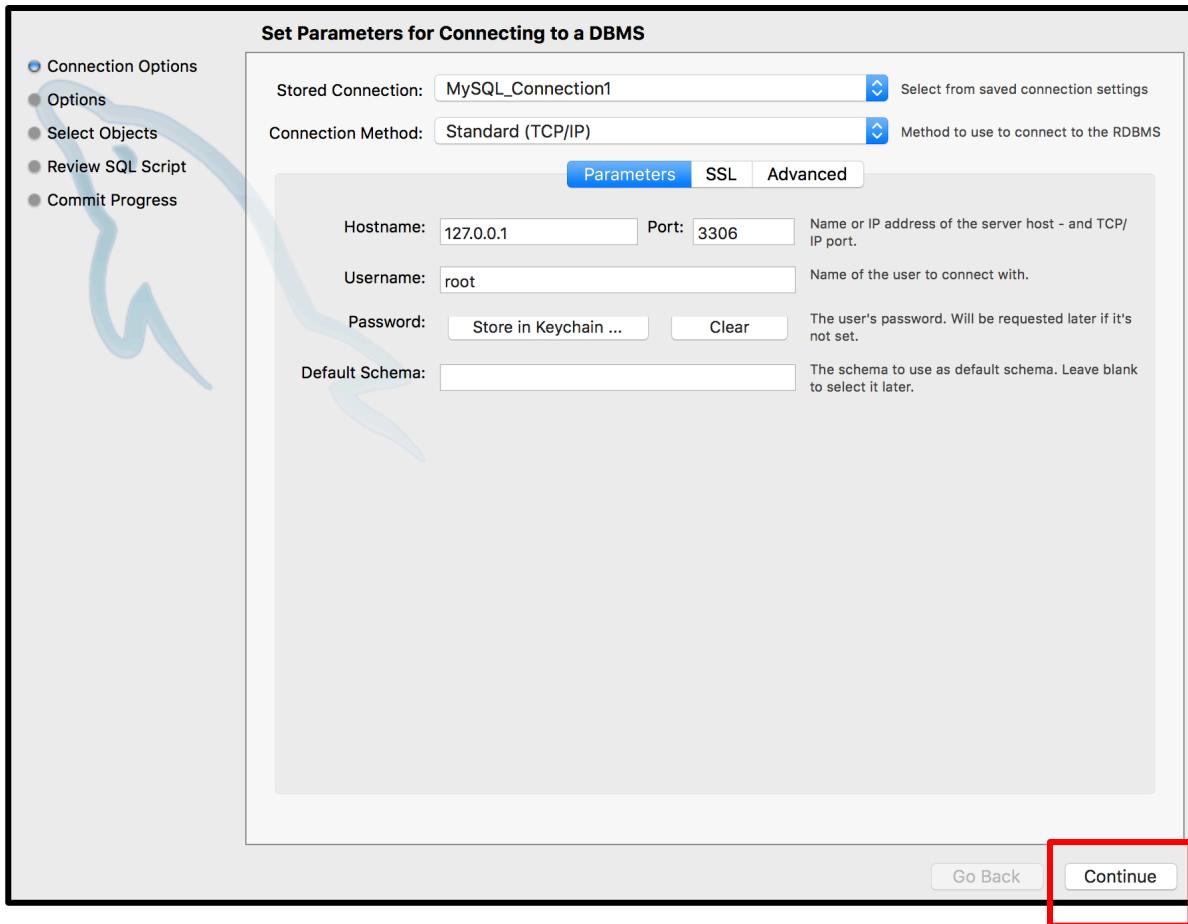
- Criação do Modelo Físico no MySQL:



# Demonstração

## MySQL

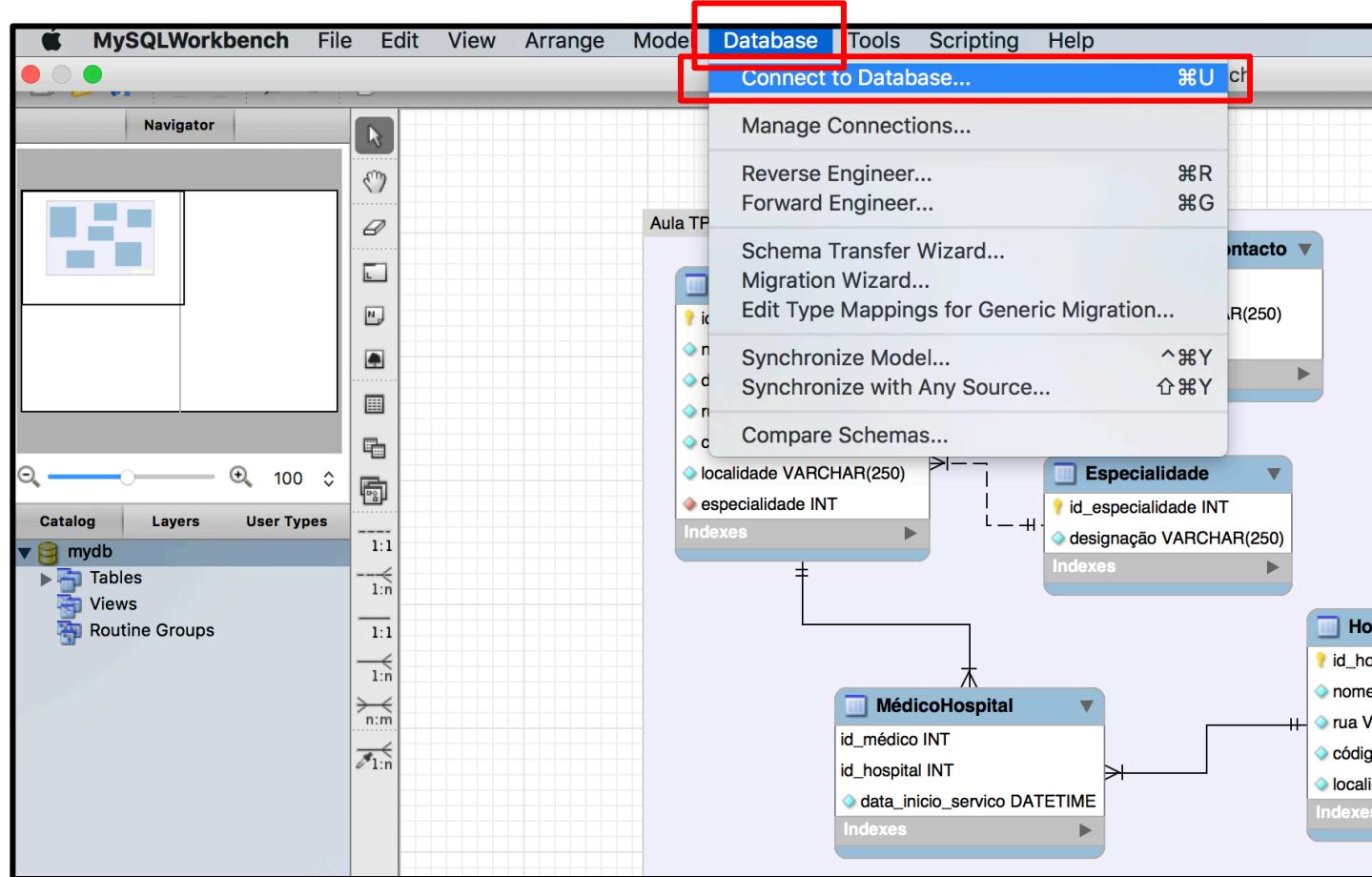
- Criação do Modelo Físico no MySQL:



# Demonstração

## MySQL

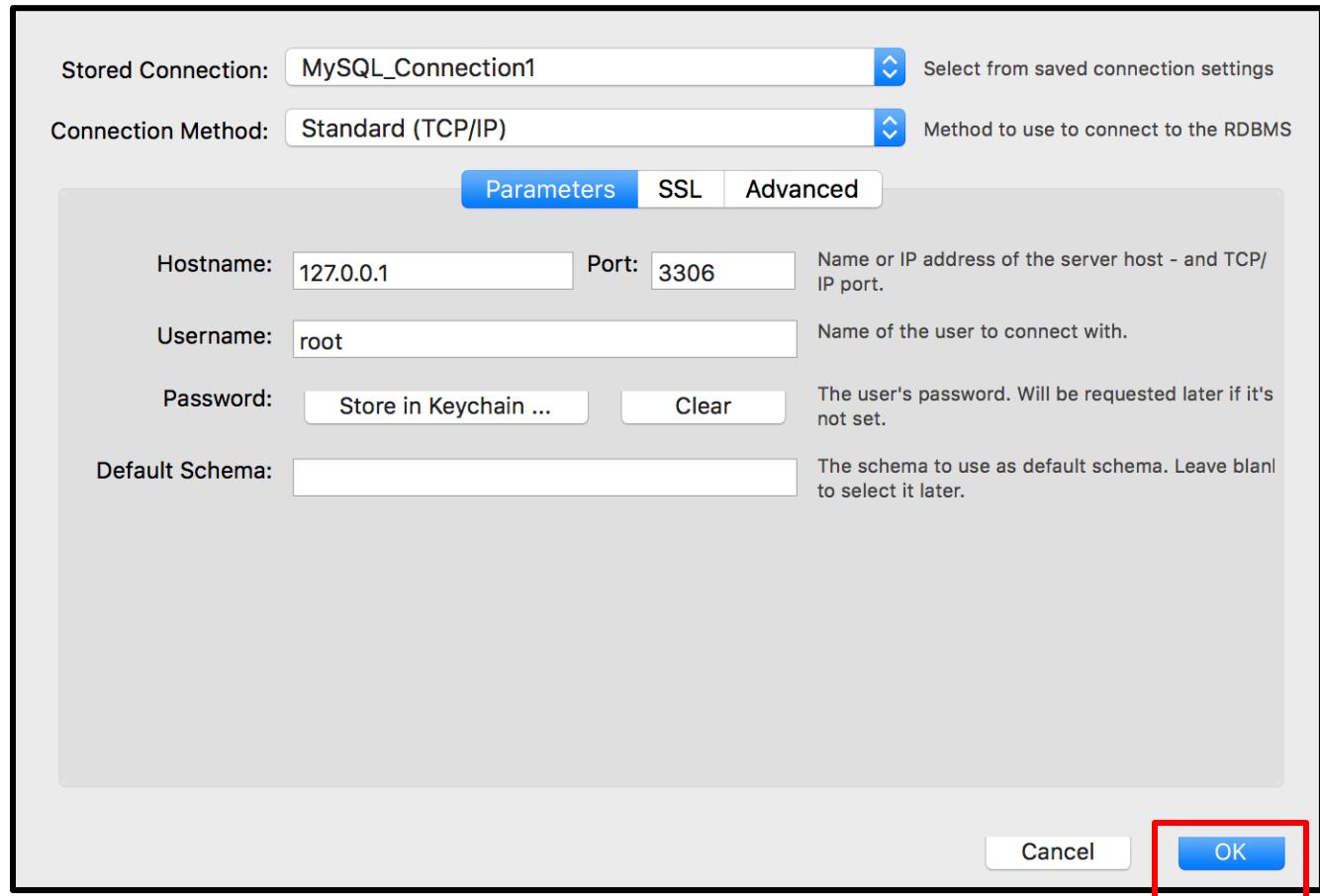
- Criação do Modelo Físico no MySQL:



# Demonstração

## MySQL

- Criação do Modelo Físico no MySQL:

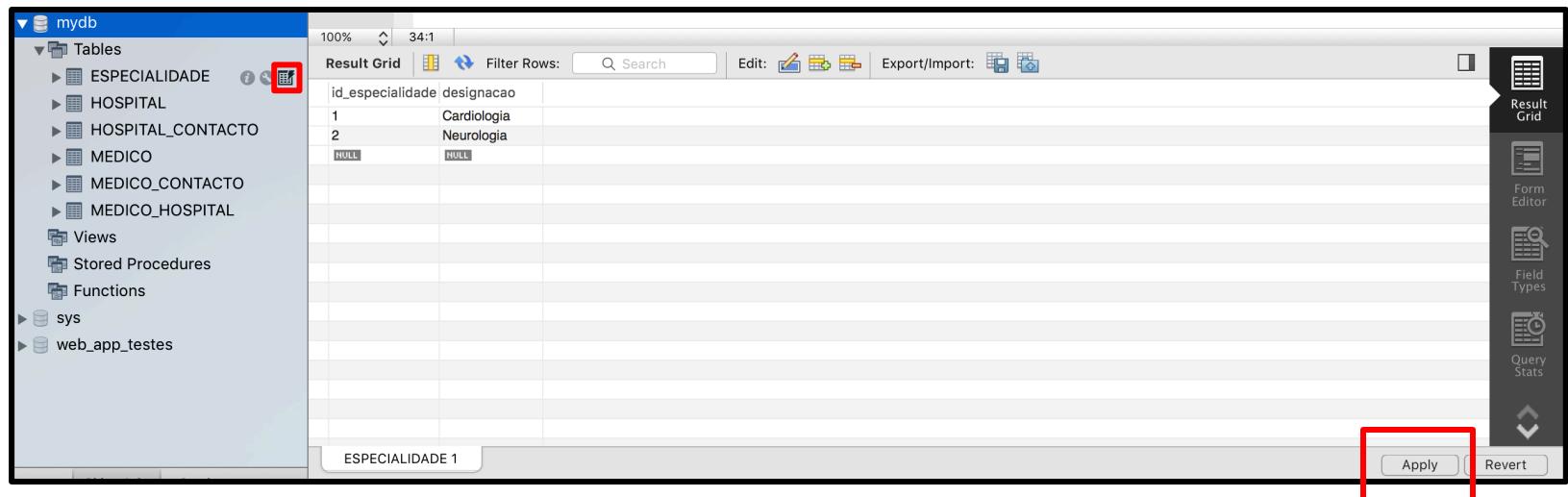


# Demonstração

## MySQL

- Povoamento das Tabelas no MySQL:

**Manualmente**

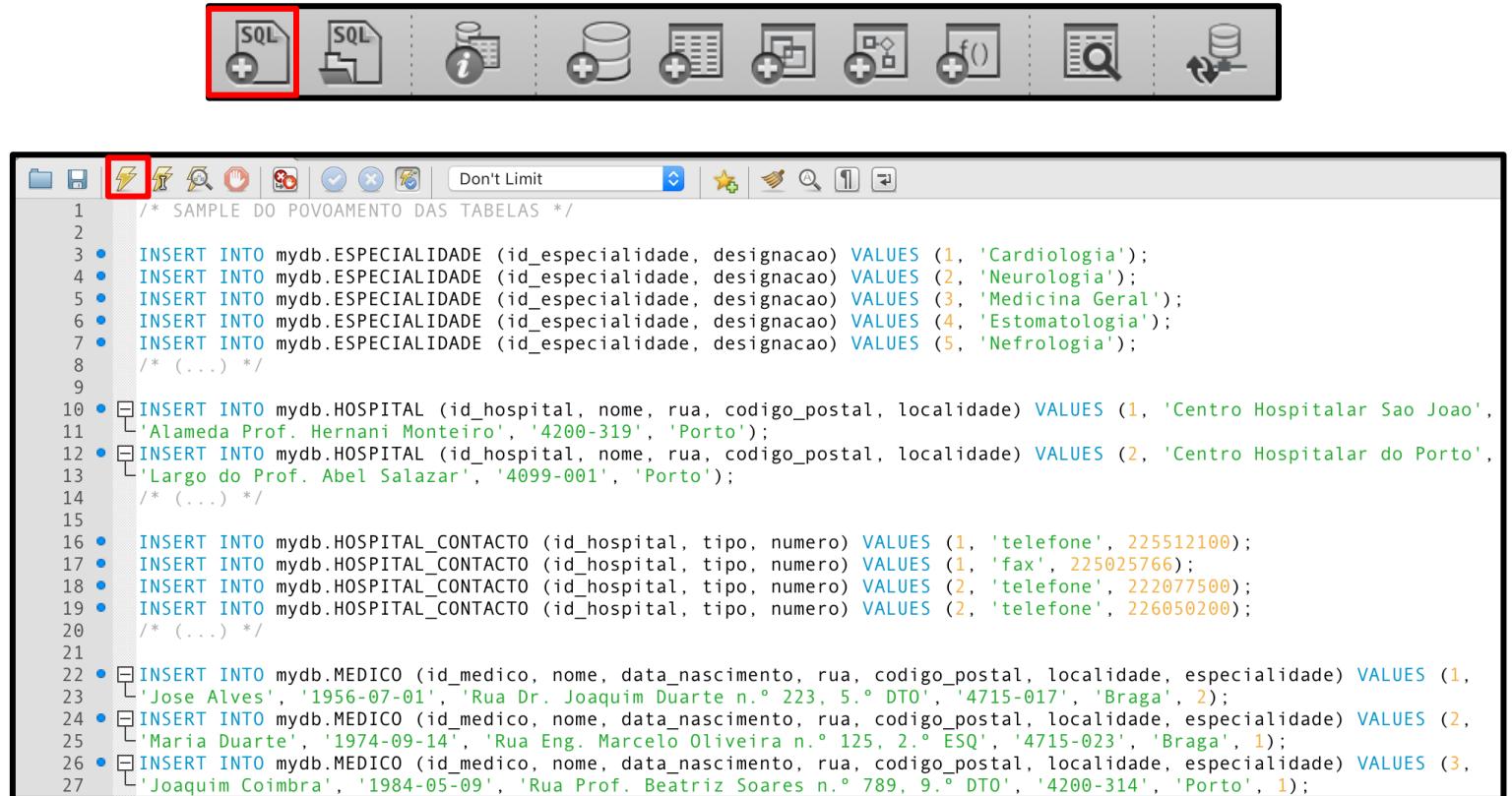


# Demonstração

MySQL

- Povoamento das Tabelas no MySQL:

Com Script



The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. One icon, which is a file with a plus sign and the text 'SQL', is highlighted with a red box. Below the toolbar is a menu bar with options like 'File', 'Edit', 'View', etc., followed by a 'Don't Limit' button. The main area is a code editor containing a SQL script for populating a database named 'mydb'. The script includes several 'INSERT INTO' statements for creating records in tables such as 'ESPECIALIDADE', 'HOSPITAL', and 'MEDICO'. The code is color-coded for syntax highlighting.

```
/* SAMPLE DO PovoAMENTO DAS TABELAS */
1
2
3 • INSERT INTO mydb.ESPECIALIDADE (id_especialidade, designacao) VALUES (1, 'Cardiologia');
4 • INSERT INTO mydb.ESPECIALIDADE (id_especialidade, designacao) VALUES (2, 'Neurologia');
5 • INSERT INTO mydb.ESPECIALIDADE (id_especialidade, designacao) VALUES (3, 'Medicina Geral');
6 • INSERT INTO mydb.ESPECIALIDADE (id_especialidade, designacao) VALUES (4, 'Estomatologia');
7 • INSERT INTO mydb.ESPECIALIDADE (id_especialidade, designacao) VALUES (5, 'Nefrologia');
8 /* (...) */
9
10 • └─ INSERT INTO mydb.HOSPITAL (id_hospital, nome, rua, codigo_postal, localidade) VALUES (1, 'Centro Hospitalar Sao Joao', 'Alameda Prof. Hernani Monteiro', '4200-319', 'Porto');
11
12 • └─ INSERT INTO mydb.HOSPITAL (id_hospital, nome, rua, codigo_postal, localidade) VALUES (2, 'Centro Hospitalar do Porto', 'Largo do Prof. Abel Salazar', '4099-001', 'Porto');
13
14 /* (...) */
15
16 • INSERT INTO mydb.HOSPITAL_CONTACTO (id_hospital, tipo, numero) VALUES (1, 'telefone', 225512100);
17 • INSERT INTO mydb.HOSPITAL_CONTACTO (id_hospital, tipo, numero) VALUES (1, 'fax', 225025766);
18 • INSERT INTO mydb.HOSPITAL_CONTACTO (id_hospital, tipo, numero) VALUES (2, 'telefone', 222077500);
19 • INSERT INTO mydb.HOSPITAL_CONTACTO (id_hospital, tipo, numero) VALUES (2, 'telefone', 226050200);
20
21
22 • └─ INSERT INTO mydb.MEDICO (id_medico, nome, data_nascimento, rua, codigo_postal, localidade, especialidade) VALUES (1, 'Jose Alves', '1956-07-01', 'Rua Dr. Joaquim Duarte n.º 223, 5.º DTO', '4715-017', 'Braga', 2);
23
24 • └─ INSERT INTO mydb.MEDICO (id_medico, nome, data_nascimento, rua, codigo_postal, localidade, especialidade) VALUES (2, 'Maria Duarte', '1974-09-14', 'Rua Eng. Marcelo Oliveira n.º 125, 2.º ESQ', '4715-023', 'Braga', 1);
25
26 • └─ INSERT INTO mydb.MEDICO (id_medico, nome, data_nascimento, rua, codigo_postal, localidade, especialidade) VALUES (3, 'Joaquim Coimbra', '1984-05-09', 'Rua Prof. Beatriz Soares n.º 789, 9.º DTO', '4200-314', 'Porto', 1);
```

# Queries em SQL

**SQL (Structured Query Language)** é uma linguagem padrão para aceder e manipular bases de dados.

- SQL pode executar consultas numa base de dados;
- SQL pode recuperar dados numa base de dados;
- SQL pode inserir registos numa base de dados;
- SQL pode atualizar registos numa base de dados;
- SQL pode apagar registos numa base de dados;
  - SQL pode criar novas bases de dados;
- SQL pode criar novas tabelas numa base de dados;
- SQL pode criar procedimentos armazenados numa base de dados;
  - SQL pode criar *views* numa base de dados;
- SQL pode definir permissões em tabelas, procedimentos e *views*.

# Queries em SQL

- `SELECT column1, column2, ... FROM table_name;`
- `SELECT DISTINCT column1, column2, ... FROM table_name;`
- `SELECT column1, column2, ... FROM table_name WHERE condition;`
  - `SELECT column1, column2, ... FROM table_name WHERE condition1 AND condition2 AND condition3 ...;`
  - `SELECT column1, column2, ... FROM table_name WHERE condition1 OR condition2 OR condition3 ...;`
  - `SELECT column1, column2, ... FROM table_name WHERE NOT condition;`
- `SELECT column1, column2, ... FROM table_name ORDER BY column1, column2, ... ASC|DESC;`

# Queries em SQL

- **INSERT INTO** *table\_name* (*column1*, *column2*, *column3*, ...) **VALUES** (*value1*, *value2*, *value3*, ...);
- **UPDATE** *table\_name* **SET** *column1* = *value1*, *column2* = *value2*, ...  
**WHERE** *condition*;
- **DELETE FROM** *table\_name* **WHERE** *condition*;
- **SELECT MIN**(*column\_name*) **FROM** *table\_name* **WHERE** *condition*;
- **SELECT MAX**(*column\_name*) **FROM** *table\_name* **WHERE** *condition*;
- **SELECT COUNT**(*column\_name*) **FROM** *table\_name* **WHERE** *condition*;
- **SELECT AVG**(*column\_name*) **FROM** *table\_name* **WHERE** *condition*;

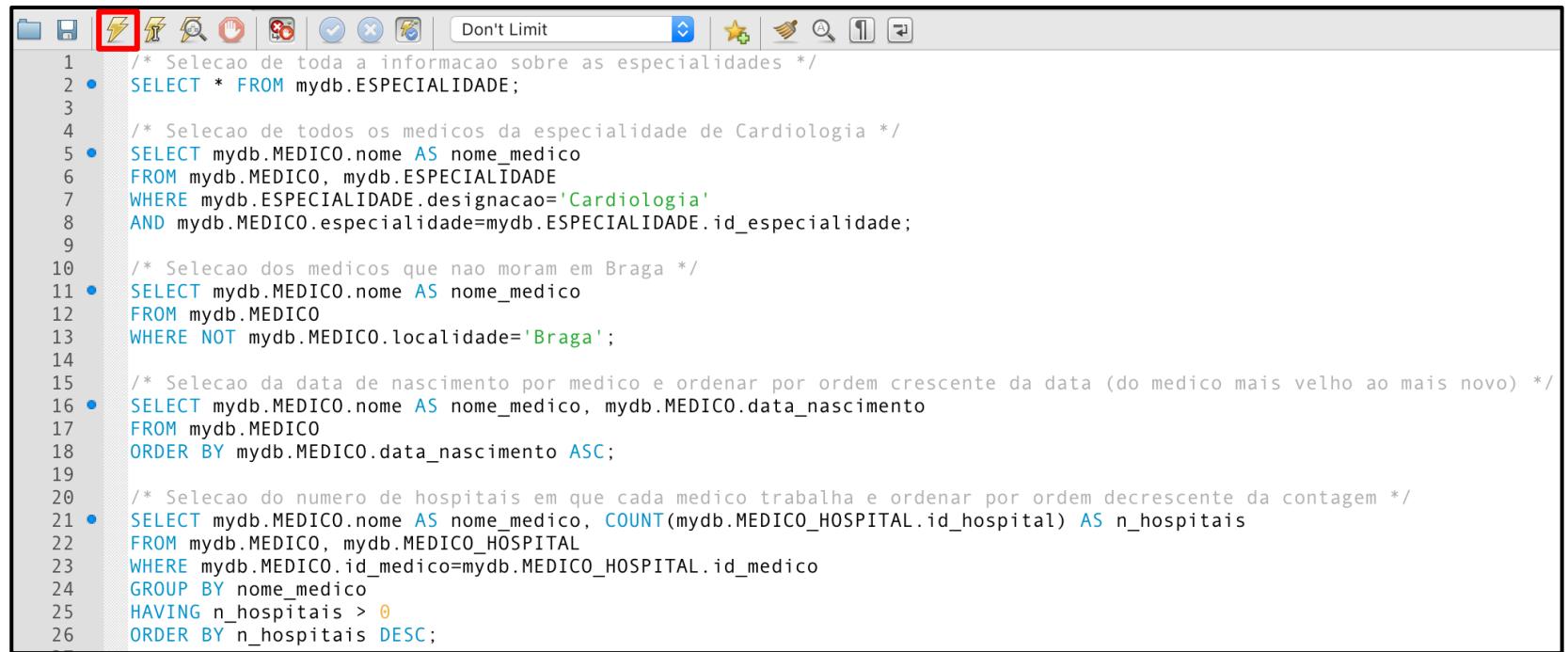
# Queries em SQL

- `SELECT SUM(column_name) FROM table_name WHERE condition;`
- `SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) ORDER BY column_name(s);`
- `SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) HAVING condition ORDER BY column_name(s);`
- **Operadores:** = (igual), > (maior), < (menor), >= (maior ou igual), <= (menor e igual), <> (diferente).

# Demonstração

## MySQL

- Questões à Base de Dados:



```
1  /* Selecao de toda a informacao sobre as especialidades */
2 • SELECT * FROM mydb.ESPECIALIDADE;
3
4  /* Selecao de todos os medicos da especialidade de Cardiologia */
5 • SELECT mydb.MEDICO.nome AS nome_medico
6    FROM mydb.MEDICO, mydb.ESPECIALIDADE
7   WHERE mydb.ESPECIALIDADE.designacao='Cardiologia'
8     AND mydb.MEDICO.especialidade=mydb.ESPECIALIDADE.id_especialidade;
9
10 /* Selecao dos medicos que nao moram em Braga */
11 • SELECT mydb.MEDICO.nome AS nome_medico
12    FROM mydb.MEDICO
13   WHERE NOT mydb.MEDICO.localidade='Braga';
14
15 /* Selecao da data de nascimento por medico e ordenar por ordem crescente da data (do medico mais velho ao mais novo) */
16 • SELECT mydb.MEDICO.nome AS nome_medico, mydb.MEDICO.data_nascimento
17    FROM mydb.MEDICO
18   ORDER BY mydb.MEDICO.data_nascimento ASC;
19
20 /* Selecao do numero de hospitais em que cada medico trabalha e ordenar por ordem decrescente da contagem */
21 • SELECT mydb.MEDICO.nome AS nome_medico, COUNT(mydb.MEDICO_HOSPITAL.id_hospital) AS n_hospitais
22    FROM mydb.MEDICO, mydb.MEDICO_HOSPITAL
23   WHERE mydb.MEDICO.id_medico=mydb.MEDICO_HOSPITAL.id_medico
24   GROUP BY nome_medico
25   HAVING n_hospitais > 0
26   ORDER BY n_hospitais DESC;
```

# Instalação do MySQL

## Download e instalação do MySQL:

- i. MySQL Installer (Windows):  
<https://dev.mysql.com/downloads/windows/installer/8.0.html>;
- ii. MySQL Workbench: <https://dev.mysql.com/downloads/workbench/>;
- iii. MySQL Community Server: <https://dev.mysql.com/downloads/mysql/>.

# Resolução da 1.<sup>a</sup> Ficha Prática Laboratorial

## 1 Registo das Milhas de Passageiros em Voos

Considere o seguinte comando em SQL para uma base de dados MySQL:

```
CREATE TABLE tabMiles (
    passengerNumber VARCHAR(20),
    flightNumber VARCHAR(20),
    flightDate DATE,
    classe VARCHAR(1),
    miles INT(11),
    AwardOrUse VARCHAR(1),
    PRIMARY KEY (passengerNumber, flightNumber, flightDate)
);
```

Na tabela *tabMiles* é registado o número de milhas que um passageiro ganha (quando *AwardOrUse* = 'A') ou usa (quando *AwardOrUse* = 'U') num determinado voo.

Considere as seguintes funções e o seguinte trigger da base de dados MySQL em questão:

```
CREATE FUNCTION 'totalAwarded'(passenger VARCHAR(20)) RETURNS INT
BEGIN
    DECLARE total INT(11) DEFAULT 0;
    SELECT SUM(miles) INTO total FROM tabMiles
    WHERE passengerNumber=passenger and awardOrUse='A';
    IF (total IS NULL) THEN
        SET total=0;
    END IF;
    RETURN total;
END
```

# Resolução da 1.<sup>a</sup> Ficha Prática Laboratorial

```
CREATE FUNCTION 'totalUsed'(passenger VARCHAR(20)) RETURNS INT
BEGIN
DECLARE total INT(11) DEFAULT 0;
SELECT SUM(miles) INTO total FROM tabMiles
WHERE passengerNumber=passenger and awardOrUse='U';
IF (total IS NULL) THEN
SET total=0;
END IF;
RETURN total;
END

CREATE TRIGGER 'validateMiles' BEFORE INSERT ON tabMiles FOR EACH ROW
BEGIN
DECLARE totalAwarded INT(11) DEFAULT 0;
DECLARE totalUsed INT(11) DEFAULT 0;
IF (new.AwardOrUse = 'U') THEN
IF (totalAwarded(new.passengerNumber)-totalUsed(new.passengerNumber))
< new.miles THEN
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Milhas insuficientes!';
END IF;
END IF;
END
```

# Resolução da 1.ª Ficha Prática Laboratorial

Estão registados 10000 passageiros com cartão de milhas. Cada passageiro faz em média 5 viagens por ano nas quais ganha milhas. Admita que todos os passageiros tem cartão de milhas.

O voo AAL409 de hoje realiza-se num BOEING 767, com capacidade para 20 passageiros em primeira classe, 40 em classe executiva e 200 em classe económica. 20 passageiros do voo AAL409 pagaram a sua viagem descontando milhas, por isso não ganharam milhas. Embarcaram 260 passageiros no total neste voo.

É necessário, para responder às perguntas, saber o nome, o género e a data de nascimento dos passageiros. É necessário também, para cada voo, saber os aeroportos de origem e de destino, assim como a distância em milhas. O mesmo voo pode realizar-se em datas diferentes (por exemplo todos os dias). É necessário também saber para cada viagem a marca do avião, o modelo do avião, o número de lugares em primeira classe (P), em classe executiva (B) e em classe económica (E). As milhas em primeira classe são multiplicadas por 3 e em classe executiva são multiplicadas por 2. Na tabela *tabMiles* o atributo *miles* já considera as milhas multiplicadas pelo fator supramencionado. Assume-se que uma viagem é um voo realizado numa determinada data. Em datas diferentes, um voo pode realizar-se em aviões diferentes.

Com base no caso apresentado, pretende-se que:

1. Instale o sistema de gestão de bases de dados relacionais MySQL:
  - (a) Sistema operativo Windows (MySQL Installer): <https://dev.mysql.com/downloads/installer/>;
  - (b) Sistema operativo macOS (MySQL Community Server & MySQL Workbench):  
<https://dev.mysql.com/downloads/>.
2. Sugere e desenhe um possível esquema conceptual da base de dados apresentada. O modelo conceptual deverá ser constituído pelas entidades *Passageiros*, *Voos* e *Viagens*, bem como os seus respetivos atributos e relacionamentos.

# Resolução da 1.<sup>a</sup> Ficha Prática Laboratorial

3. Utilizando o MySQL Workbench, faça a conversão do modelo conceptual definido na alínea anterior para o seu respetivo modelo lógico (*EER Diagram*). Poderá ser conveniente alterar a estrutura apresentada no enunciado da tabela *tabMiles*.
4. Seguidamente, faça a geração do esquema físico para uma base de dados (Database > Forward Engineer).
5. Implemente as funções e o trigger definidos no enunciado desta ficha prática laboratorial na base de dados criada.
6. Povoe a base de dados criada adequadamente com uma *script* em SQL.
7. Utilizando SQL, desenvolva as *queries*, bem como a função *idade*, para responder às seguintes questões:
  - (a) Quais são os nomes e as idades dos passageiros que ganharam milhas no voo AAL409 de hoje?
  - (b) Quais são os nomes e os géneros dos passageiros que gastaram milhas, em classe económica, em voos realizados hoje, a partir do Aeroporto Francisco Sá Carneiro, em aeronaves da marca BOEING?
  - (c) Quais são os nomes dos passageiros que ganharam mais de 5000 milhas num determinado voo?
  - (d) Quais são os nomes dos passageiros que nunca gastaram milhas?