

# MODELAGEM DE SOFTWARE

Prof. Saulo Popov Zambiasi

Prof. Richard Henrique de Souza

Prof. Ricardo Ribeiro Assink

Prof. Edson Lessa



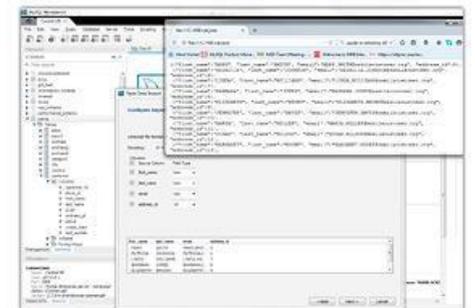
# Vamos instalar as ferramentas

- MySQL
  - <https://dev.mysql.com/downloads/installer/>
    - mysql-installer-community-8.0.29.0
- Workbenach
  - <https://www.mysql.com/products/workbench/>

**MySQL Workbench**

Enhanced Data Migration

Download Now »



# Faça o download do MySQL

General Availability (GA) Releases

Archives



## MySQL Installer 8.0.29

Select Operating System:

Microsoft Windows

[Looking for previous GA versions?](#)

**Windows (x86, 32-bit), MSI Installer**

8.0.29

2.3M

[Download](#)

(mysql-installer-web-community-8.0.29.0.msi)

MD5: 4f735569267527dec28d9e8d977f33d1 | [Signature](#)

**Windows (x86, 32-bit), MSI Installer**

8.0.29

439.6M

[Download](#)

(mysql-installer-community-8.0.29.0.msi)

MD5: 3f4def7aef5e2e030e2dd62e784f246 | [Signature](#)



We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

# Faça o download do Workbench

[General Availability \(GA\) Releases](#)

[Archives](#)



## MySQL Workbench 8.0.29

Select Operating System:

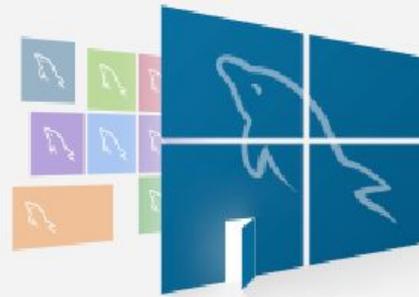
Microsoft Windows

Recommended Download:

### MySQL Installer for Windows

All MySQL Products. For All Windows Platforms.  
In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.



**Windows (x86, 32 & 64-bit), MySQL Installer MSI**

[Go to Download Page >](#)

Other Downloads:

**Windows (x86, 64-bit), MSI Installer**

8.0.29

42.9M

[Download](#)

(mysql-workbench-community-8.0.29-winx64.msi)

MD5: 2c40a1e64dd8be34c91501749961dd9c | [Signature](#)

# Instalando o MySQL

MySQL. Installer  
Adding Community

## Select Products

Please select the products you would like to install on this computer.



Filter:

All Software, Current Bundle, Any

Edit

Select Products

Installation

Product Configuration

Installation Complete

### Available Products:

- Connector/J 8.0
  - Connector/J 8.0.29 - X86
- Connector/NET
  - Connector/NET 8.0
    - Connector/NET 8.0.29 - X86
- Connector/Python
- Documentation
  - MySQL Documentation
    - MySQL Documentation 8.0
      - MySQL Documentation 8.0
  - Samples and Examples
    - Samples and Examples 8.0
      - Samples and Examples 8.0

### Products To Be Installed:

- MySQL Server 8.0.29 - X64
- MySQL Shell 8.0.29 - X64
- Connector/ODBC 8.0.29 - X64
- Connector/C++ 8.0.29 - X64
- Connector/J 8.0.29 - X86
- Connector/NET 8.0.29 - X86
- MySQL Documentation 8.0.29 - X86
- Samples and Examples 8.0.29 - X86



Enable the Select Features page to customize product features

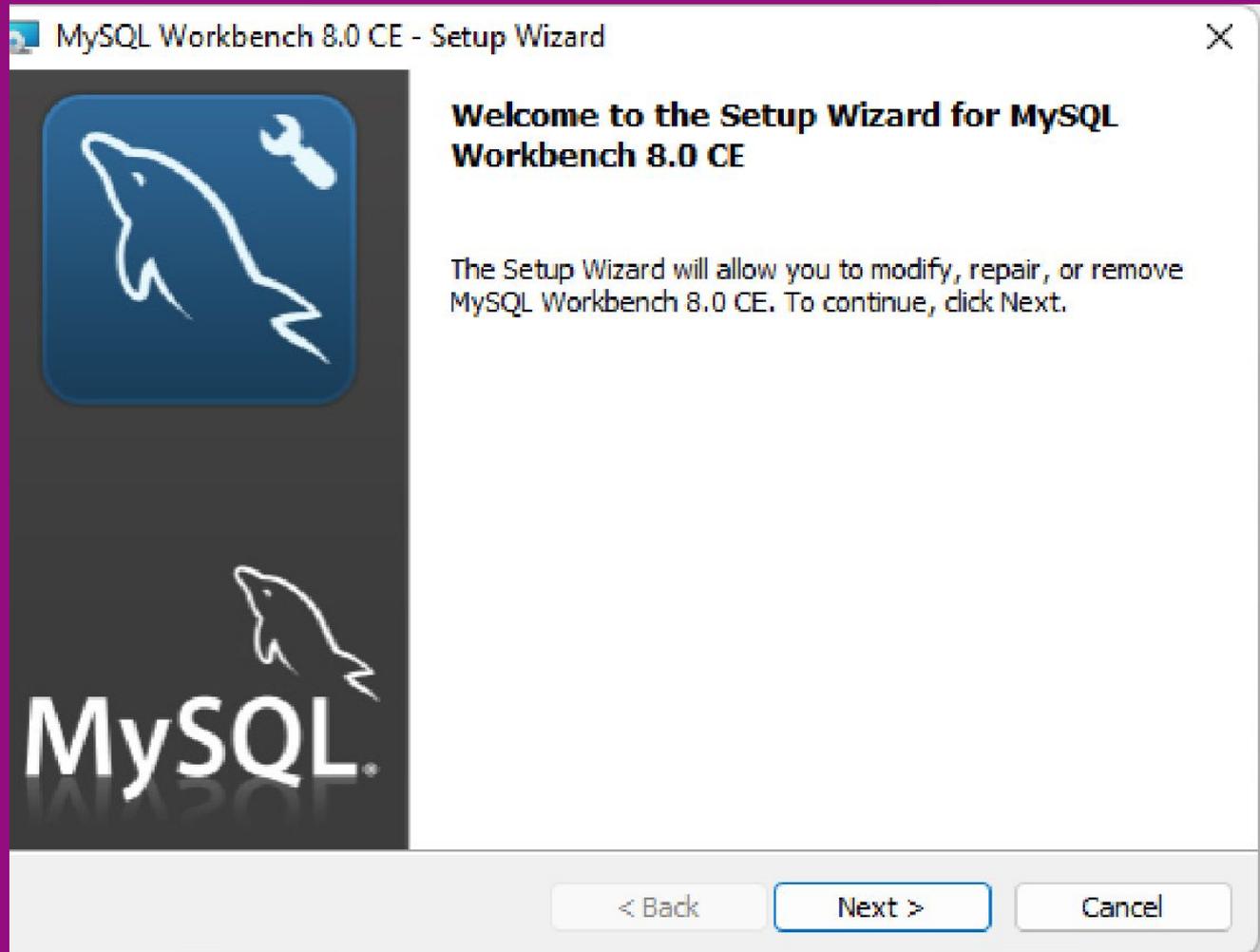
Published: terça-feira, 26 de abril de 2022

Release Notes:

Next >

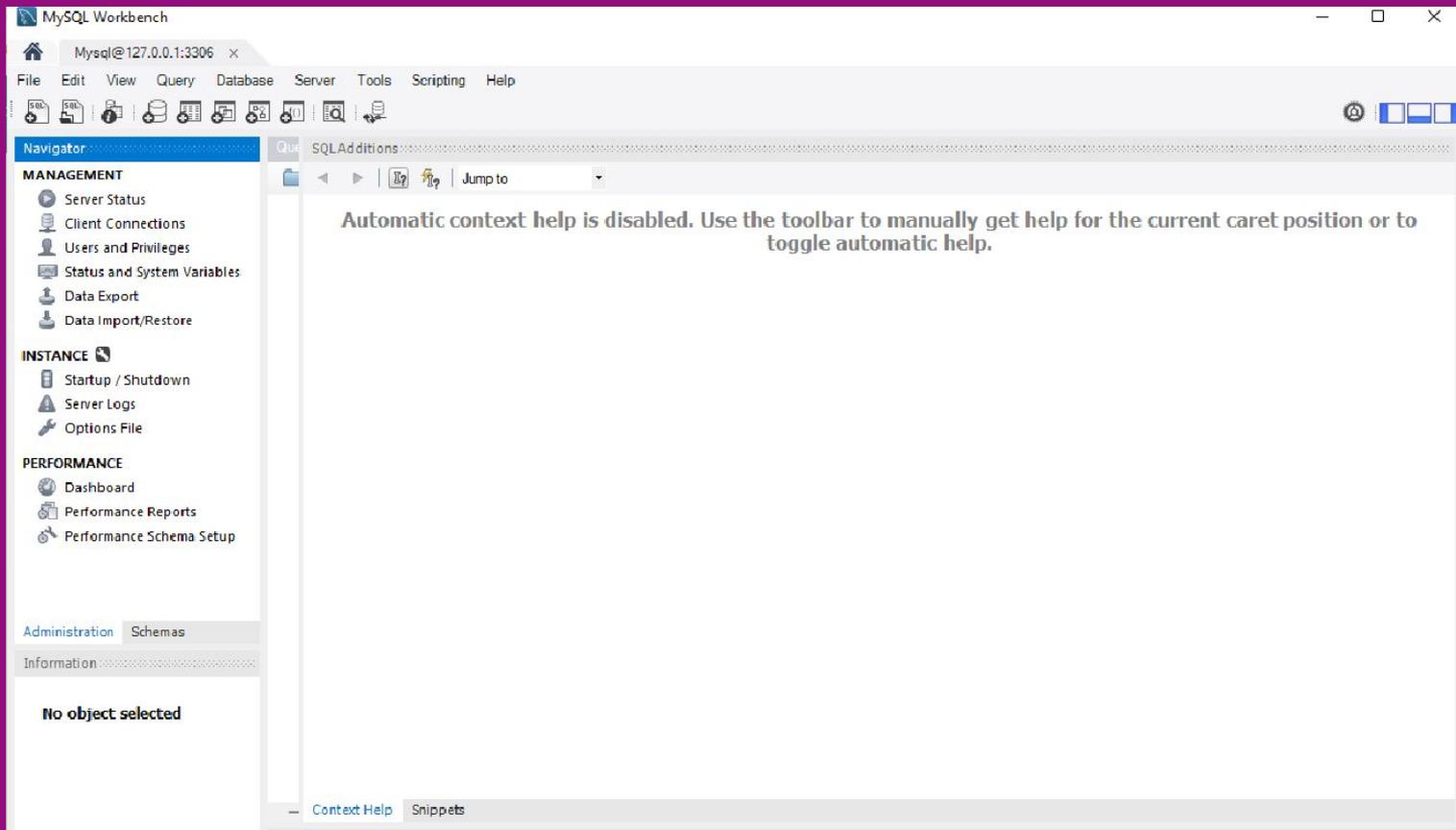
Cancel

# Instale o Workbench



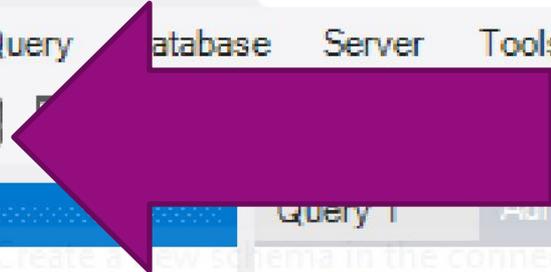
# Workbench

Depois de tudo instalado, abra o workbench



Mysql@127.0.0.1:3306 x

File Edit View Query Database Server Tools Script



Navigator

**MANAGEMENT**

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

**INSTANCE**

- Startup / Shutdown
- Server Logs
- Options File

**PERFORMANCE**

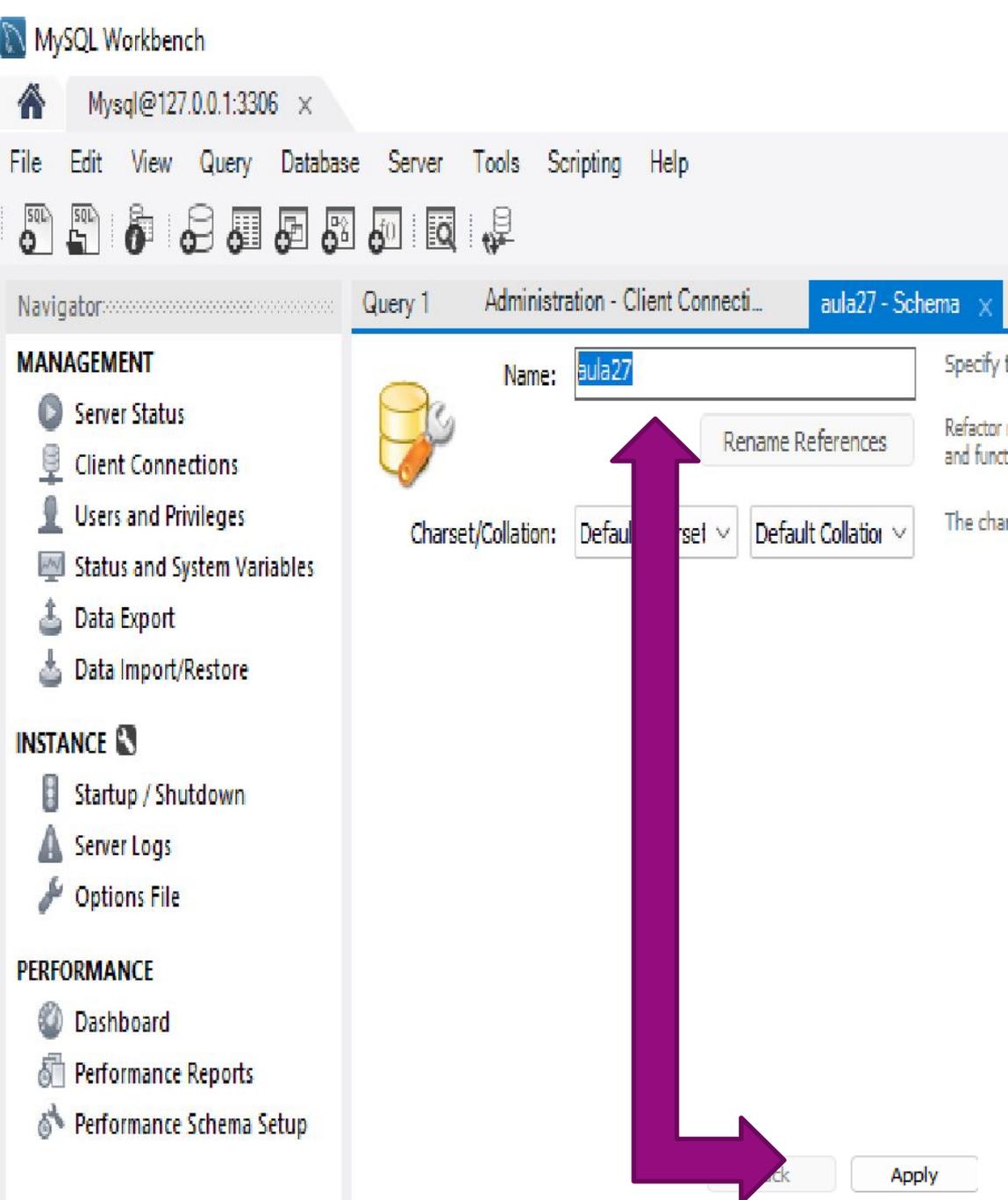
- Dashboard
- Performance Reports
- Performance Schema Setup

Client Connections

Threads Connected: 4  
Total Connections: 14

Id	User	Host
5	event_sched...	lo
7	None	Ne
11	root	lo
12	root	lo
13	root	lo
14	root	lo

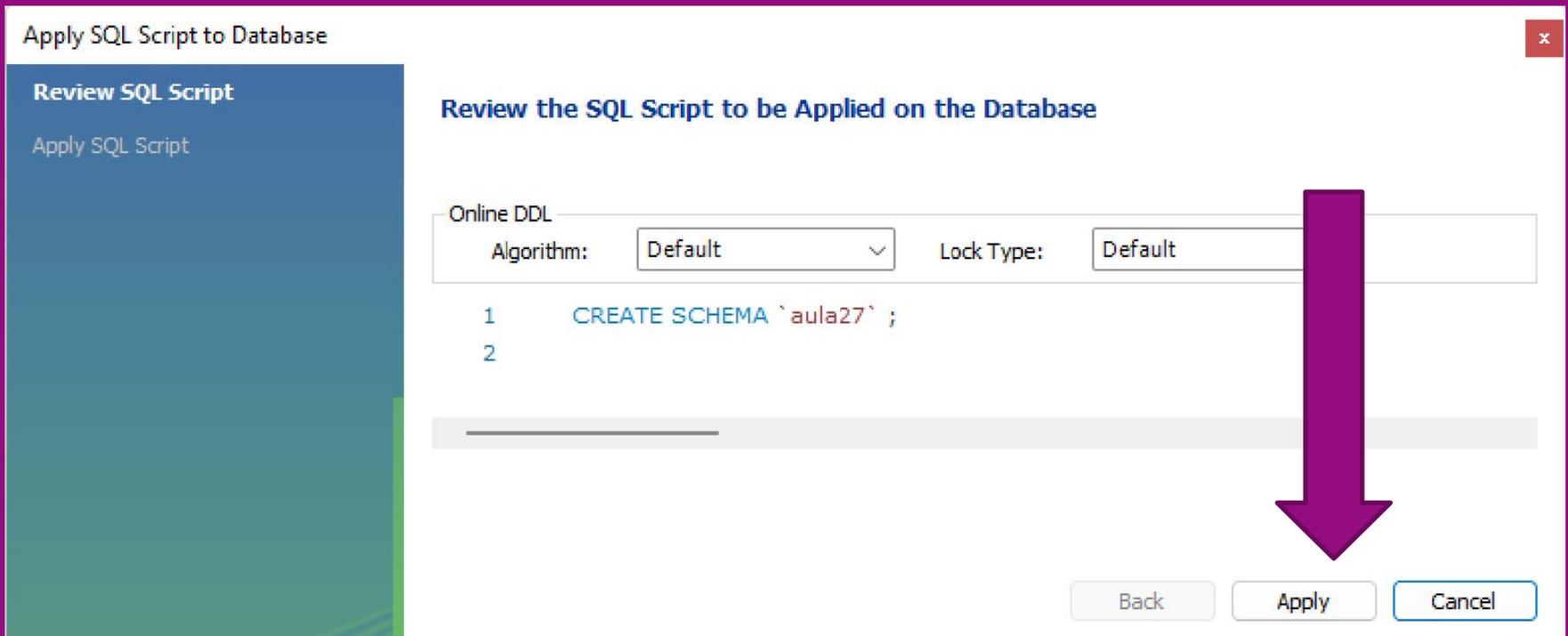
- Crie um Schema



- Indique um nome
- Clique em Apply
- Exemplo aula27

# CREATE SCHEMA

- CREATE SCHEMA `aula27` ;



Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Default Lock Type: Default

```
1 CREATE SCHEMA `aula27` ;
2
```

Back Apply Cancel

A large red arrow points from the top of the 'Apply' button to the bottom of the dialog box.

# Finalise

Apply SQL Script to Database

Review SQL Script

**Apply SQL Script**

### Applying SQL script to the database

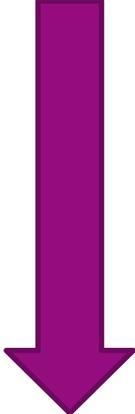
The following tasks will now be executed. Please monitor the execution. Press Show Logs to see the execution logs.

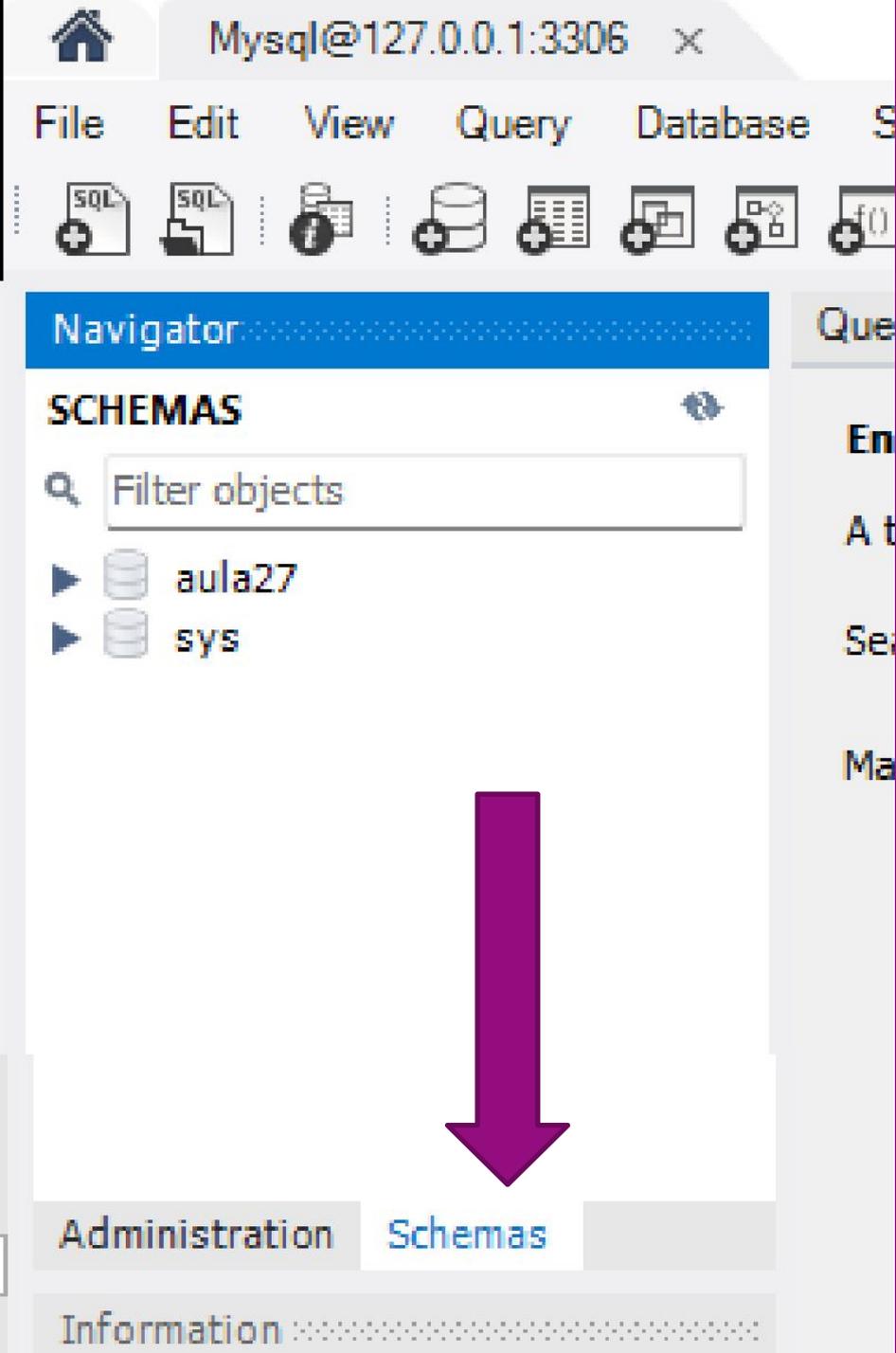
- Execute SQL Statements

SQL script was successfully applied to the database.

Show Logs

Back Finish Cancel





- Selezione a aba Schemas

## SCHEMAS



Filter objects

Enter t

A text s

Load Spatial Data

Set as Default Schema

Filter to This Schema

Schema Inspector

Table Data Import Wizard

Copy to Clipboard

Send to SQL Editor

Create Schema...

Alter Schema...

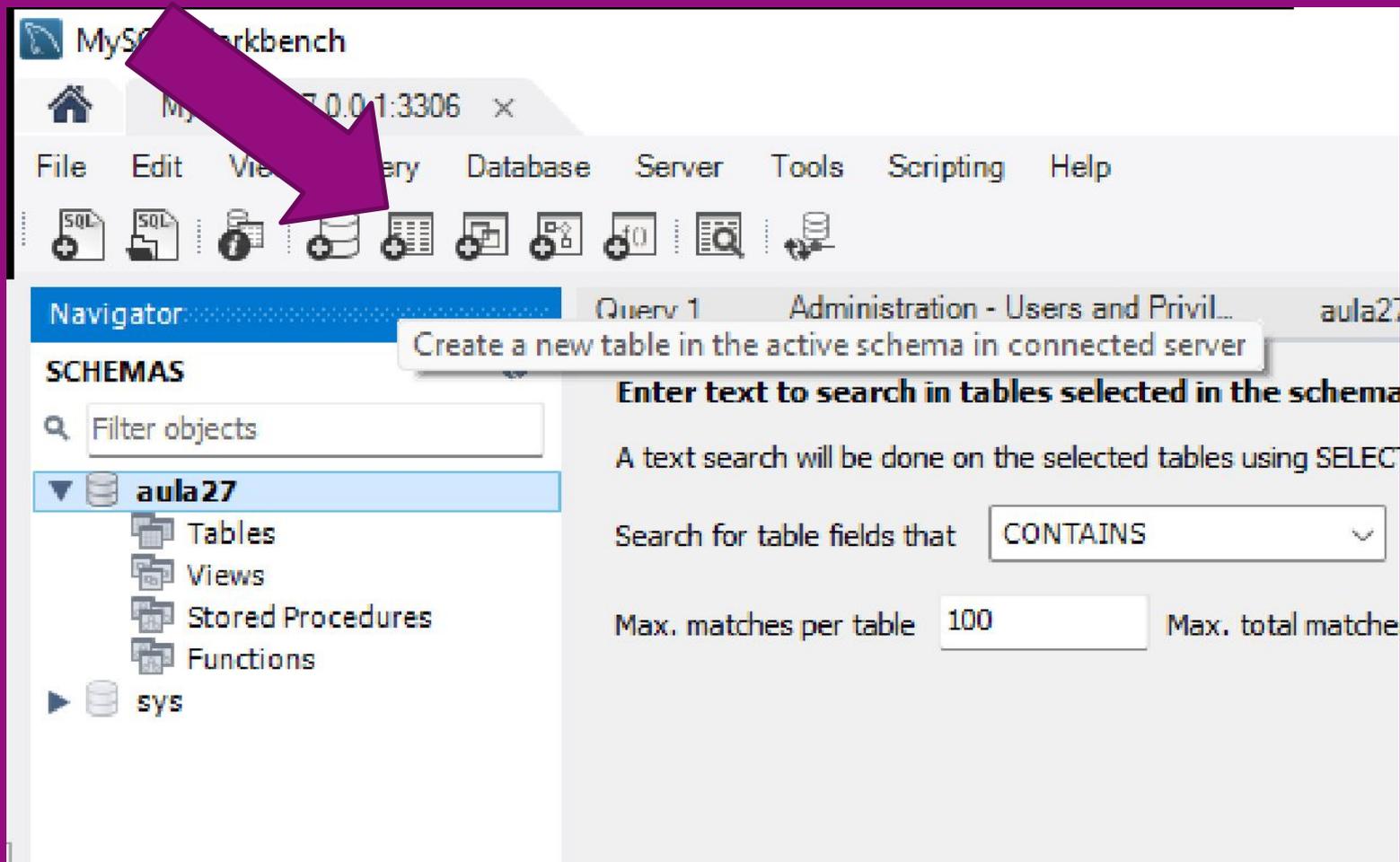
Drop Schema...

Search Table Data...

Refresh All

- Botão direito no schema aula27 e acione a opção “Set as Default Schema”

# Criar Tabela



The screenshot displays the MySQL Workbench application window. The title bar reads "MySQL Workbench". The menu bar includes "File", "Edit", "View", "Query", "Database", "Server", "Tools", "Scripting", and "Help". The toolbar contains various icons for file operations, database management, and search. The "Navigator" pane on the left shows the "SCHEMAS" section with a search box labeled "Filter objects". Underneath, the "aula27" schema is expanded, showing "Tables", "Views", "Stored Procedures", and "Functions". The "sys" schema is also visible. A tooltip is displayed over the "Create a new table" icon in the toolbar, containing the text: "Create a new table in the active schema in connected server".

MySQL Workbench

MySQL 5.0.0:1:3306 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

aula27

- Tables
- Views
- Stored Procedures
- Functions

sys

Create a new table in the active schema in connected server

Enter text to search in tables selected in the schema

A text search will be done on the selected tables using SELECT

Search for table fields that CONTAINS

Max. matches per table 100 Max. total matches

# Vamos usar nosso exemplo anterior

## Campo

Unidade básica de uma tabela. Possui um nome, tipo e tamanho.

Exemplo: Nome, e-mail, endereço

## Tabela Lógica

União de linhas e colunas para armazenamento de dados dos sistemas.

Exemplo: Tabela de clientes



Campos

Documento	Nome	Email	Endereço
012.012-11	João da Silva	<u>js@gmail.com</u>	Rua Acacias
123.123-00	Maria da Silva	<u>ma@gmail.com</u>	Rua Acacias

The screenshot shows a database management interface with the following configuration:

- Table Name:** Cliente
- Schema:** aula27
- Charset/Collation:** Default Charset, Default Collation
- Engine:** InnoDB
- Comments:** (empty text area)

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Documento	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					

Below the table, the configuration for the 'Documento' column is shown:

- Column Name:** Documento
- Data Type:** INT
- Charset/Collation:** Default Charset, Default Collation
- Comments:** (empty text area)
- Storage:**  Virtual,  Stored
- Options:**  Primary Key,  Not Null,  Unique,  Binary,  Unsigned,  Zero Fill,  Auto Increment,  Generated

At the bottom, there are tabs for: Columns, Indexes, ForeignKeys, Triggers, Partitioning, Options.

- A Tabela será Cliente e documento a nossa chave primária



Table Name:

Schema: **aula27**

Charset/Collation:

Engine:

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Documento	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
nome	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
email	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
endereco	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name:

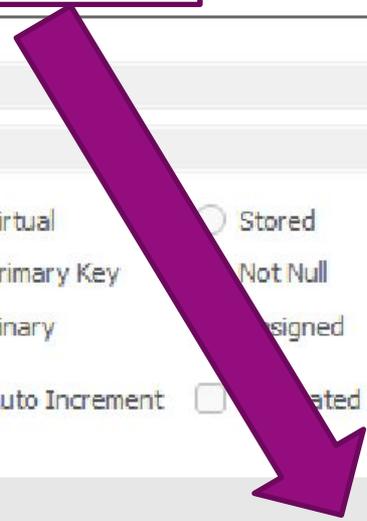
Data Type:

Charset/Collation:

Default:

Comments:

Storage:  Virtual  Stored  
 Primary Key  Not Null  Unique  
 Binary  Unsigned  Zero Fill  
 Auto Increment  Generated



Columns Indexes Foreign Keys Triggers Partitioning Options

- Preencha os demais campos e clique em Apply

## Review the SQL Script to be Applied on the Database

Online DDL

Algorithm:

Default

Lock Type:

Default

```
1  CREATE TABLE `aula27`.`cliente` (  
2    `Documento` INT NOT NULL,  
3    `nome` VARCHAR(45) NULL,  
4    `email` VARCHAR(45) NULL,  
5    `endereco` VARCHAR(45) NULL,  
6    PRIMARY KEY (`Documento`),  
7    UNIQUE INDEX `Documento_UNIQUE` (`Documento` ASC) VISIBLE);  
8
```

Back

Apply

Cancel

- Clique em Apply

# A ferramenta gera o SQL

- ```
CREATE TABLE `aula27`.`cliente` (  
  `Documento` INT NOT NULL, `nome`  
  VARCHAR(45) NULL, `email` VARCHAR(45)  
  NULL, `endereco` VARCHAR(45) NULL,  
  PRIMARY KEY (`Documento`), UNIQUE INDEX  
  `Documento_UNIQUE` (`Documento` ASC)  
  VISIBLE);
```

## Navigator

### SCHEMAS

Filter objects

#### aula27

##### Tables

##### cliente

##### Columns

- ◆ Documento
- ◆ nome
- ◆ email
- ◆ endereco

##### Indexes

##### Foreign Keys

##### Triggers

##### Views

##### Stored Procedures

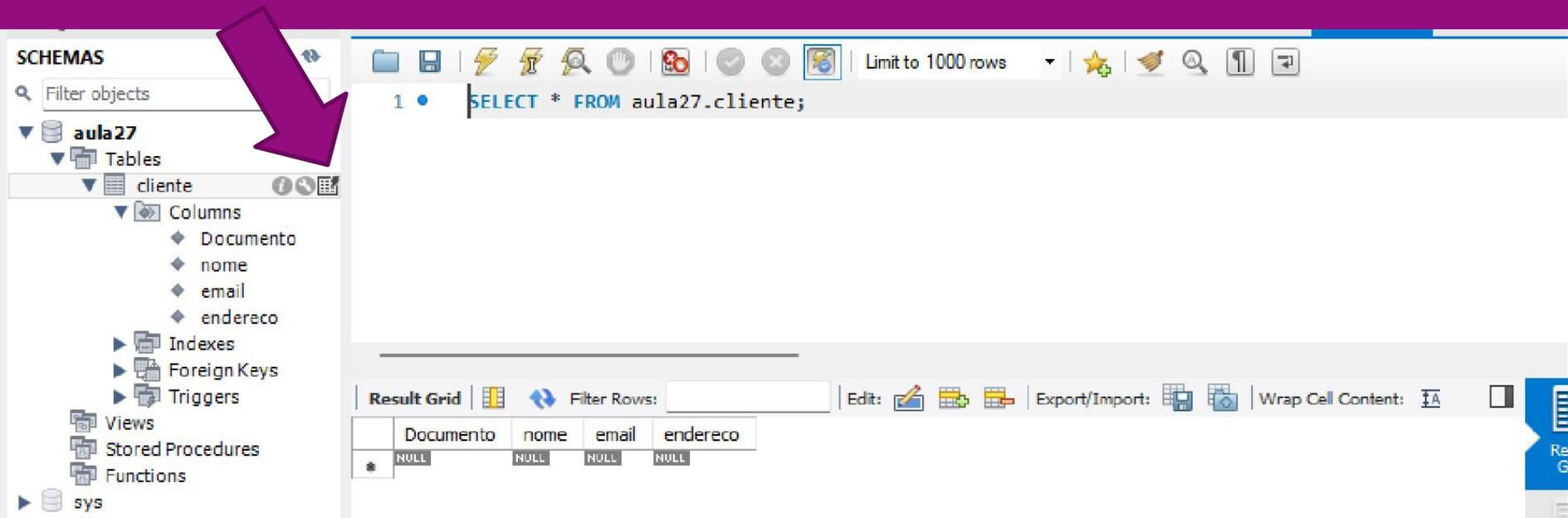
##### Functions

#### sys

- Pronto
- Primeira Tabela criada

# Select

- Clique no símbolo de tabela



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' tree is expanded to show the 'aula27' database, with the 'cliente' table selected. A red arrow points to the table icon next to 'cliente'. The central query window contains the SQL statement: `SELECT * FROM aula27.cliente;`. Below the query window, the 'Result Grid' is visible, showing a single row of data with all columns (Documento, nome, email, endereco) containing NULL values.

|   | Documento | nome | email | endereco |
|---|-----------|------|-------|----------|
| * | NULL      | NULL | NULL  | NULL     |

# Vamos usar nosso exemplo anterior

## Campo

Unidade básica de uma tabela. Possui um nome, tipo e tamanho.

Exemplo: Nome, e-mail, endereço

## Tabela Lógica

União de linhas e colunas para armazenamento de dados dos sistemas.

Exemplo: Tabela de clientes

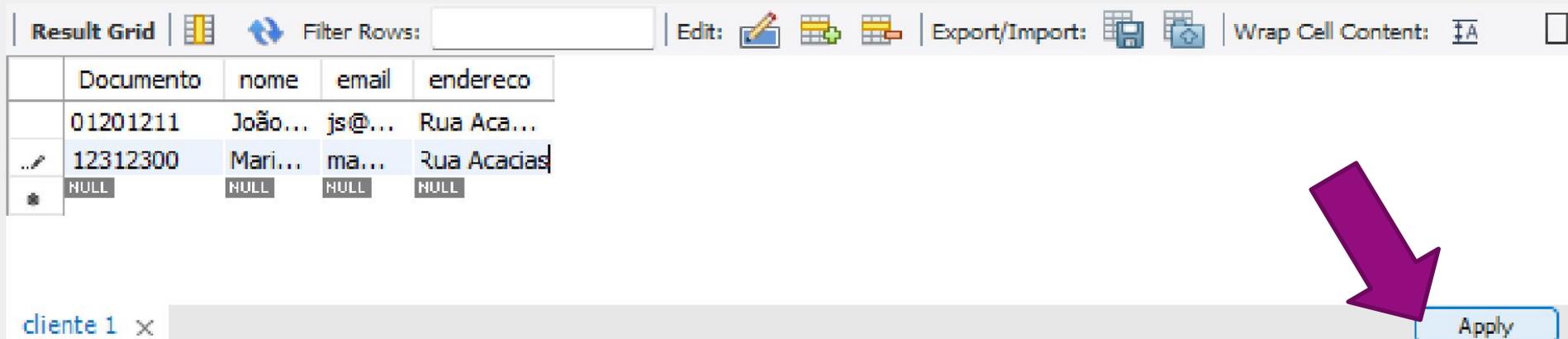


Campos

| Documento  | Nome           | Email               | Endereço    |
|------------|----------------|---------------------|-------------|
| 012.012-11 | João da Silva  | <u>js@gmail.com</u> | Rua Acacias |
| 123.123-00 | Maria da Silva | <u>ma@gmail.com</u> | Rua Acacias |

# Salvando dados

- Preencha os dados conforme o exemplo e clique em Apply



The screenshot shows a data grid interface with a toolbar at the top. The toolbar includes options for 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. Below the toolbar is a table with the following data:

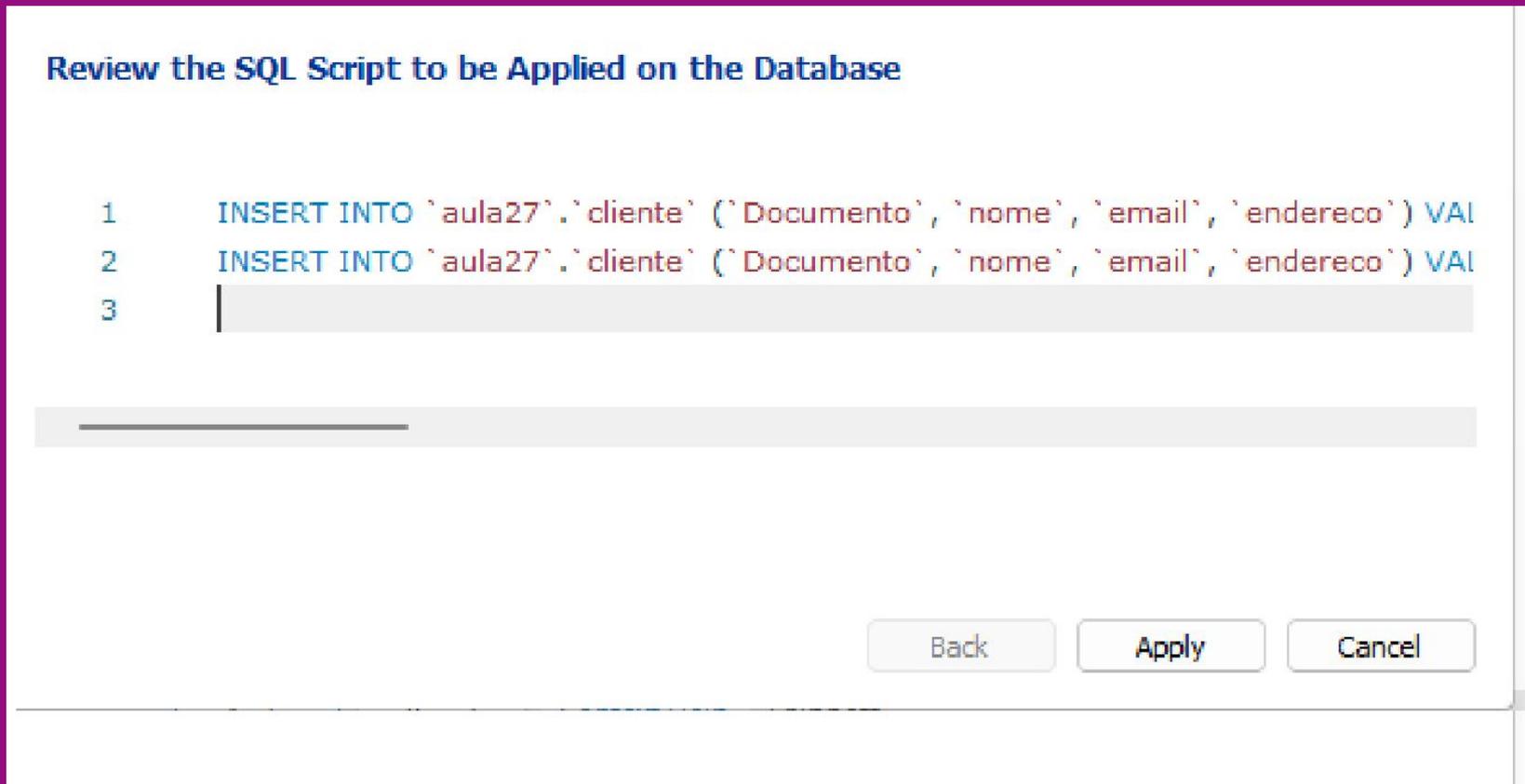
|      | Documento | nome    | email  | endereco    |
|------|-----------|---------|--------|-------------|
|      | 01201211  | João... | js@... | Rua Aca...  |
| .. / | 12312300  | Mari... | ma...  | Rua Acacias |
| *    | NULL      | NULL    | NULL   | NULL        |

At the bottom of the interface, there is a tab labeled 'cliente 1' and an 'Apply' button. A red arrow points to the 'Apply' button.

Output

# Comando INSERT do SQL

- Confirme  Apply

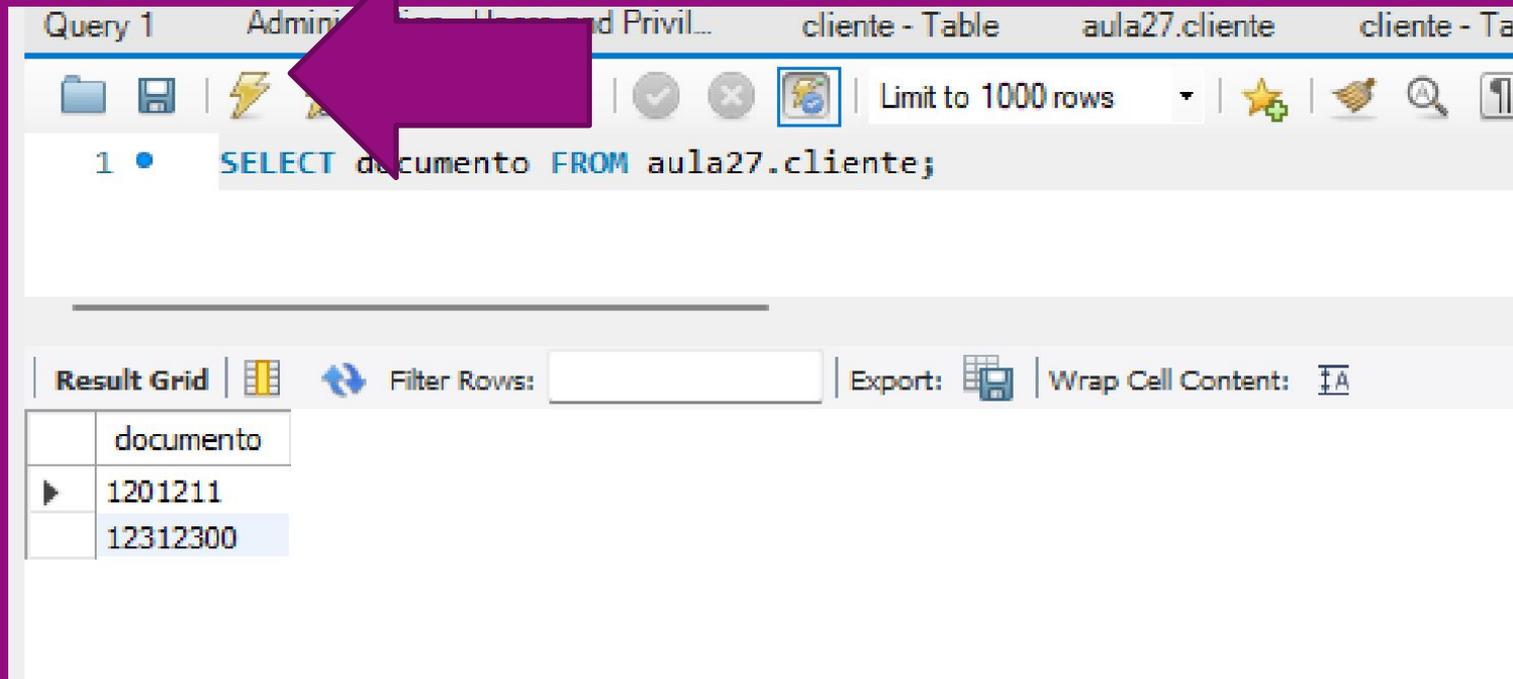


# A Ferramenta gerou os “inserts” no banco

- ```
INSERT INTO `aula27`.`cliente` (`Documento`,  
`nome`, `email`, `endereco`) VALUES  
('01201211', 'João da Silva', 'js@gmail.com',  
'Rua Acacias');
```
- ```
INSERT INTO `aula27`.`cliente` (`Documento`,  
`nome`, `email`, `endereco`) VALUES  
('12312300', 'Maria da Silva',  
'ma@gmail.com', 'Rua Acacias');
```

# Agora podemos brincar

- `SELECT Documento FROM aula27.cliente;`
- Retorna apenas os Dados do campo Documento (Clique no símbolo do raio)



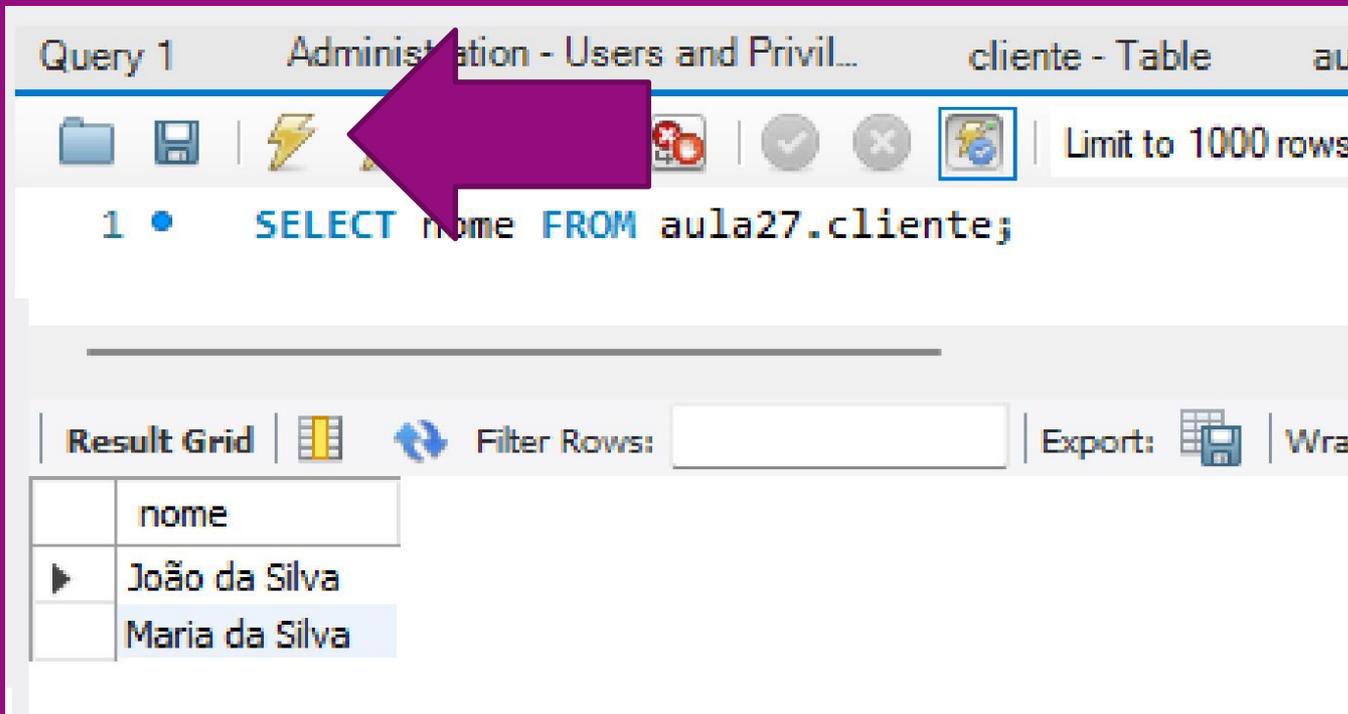
The screenshot shows a SQL query editor window. The query is `SELECT documento FROM aula27.cliente;`. The results are displayed in a table with the following data:

|   | documento |
|---|-----------|
| ▶ | 1201211   |
|   | 12312300  |

A red arrow points to the 'Limit to 1000 rows' dropdown menu in the toolbar.

# Agora podemos brincar

- `SELECT nome FROM aula27.cliente;`
- Retorna apenas os Dados do campo nome  
(Clique no símbolo do raio)



The screenshot shows a SQL query editor window. The title bar includes 'Query 1', 'Administration - Users and Privil...', 'cliente - Table', and 'au'. The toolbar contains icons for folder, save, lightning bolt, a red skull and crossbones, a checkmark, a close button, a globe, and a 'Limit to 1000 rows' option. The query text is: `1 • SELECT nome FROM aula27.cliente;`. Below the query, the 'Result Grid' is visible, showing a table with the following data:

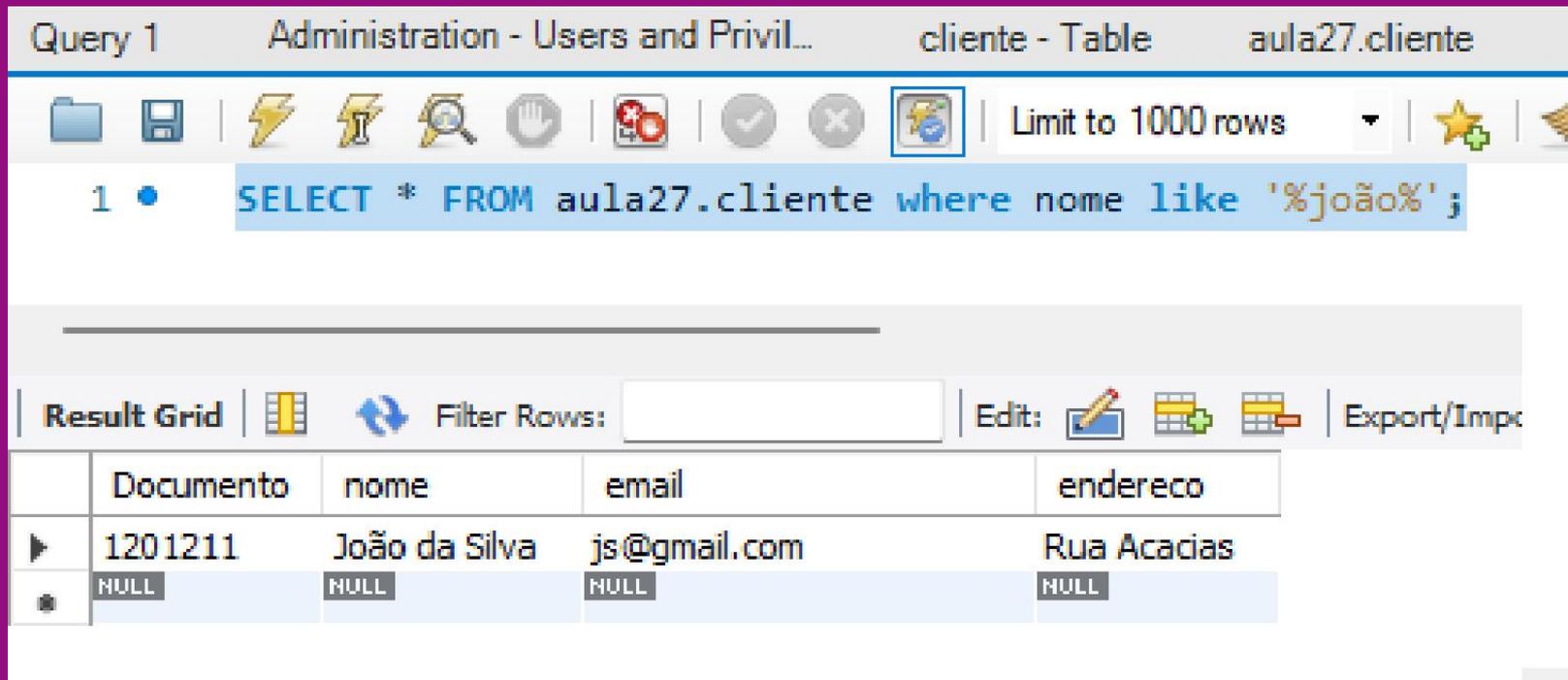
|   | nome           |
|---|----------------|
| ▶ | João da Silva  |
|   | Maria da Silva |

# Exercício 1

- Faça o select para os demais campos
  - Endereco
  - email

# Adiantando um pouco as coisas

- `SELECT * FROM aula27.cliente where nome like '%joão%';`



The screenshot shows a database query tool interface. At the top, there are tabs for 'Query 1', 'Administration - Users and Privil...', 'cliente - Table', and 'aula27.cliente'. Below the tabs is a toolbar with various icons, including a folder, save, lightning bolt, magnifying glass, and a dropdown menu set to 'Limit to 1000 rows'. The main area displays a SQL query: `SELECT * FROM aula27.cliente where nome like '%joão%';`. Below the query is a 'Result Grid' section with a 'Filter Rows' input field and 'Edit' and 'Export/Impo' buttons. The result grid shows a table with the following data:

|   | Documento | nome          | email        | endereco    |
|---|-----------|---------------|--------------|-------------|
| ▶ | 1201211   | João da Silva | js@gmail.com | Rua Acacias |
| • | NULL      | NULL          | NULL         | NULL        |

# Adiantando um pouco as coisas

- `SELECT * FROM aula27.cliente where nome like '%joão%';`
  - Where: Para especificar um parametro de busca
  - Quando for texto, deve estar entre “
  - A palavra “like” neste caso serve para dizer para retornar qualquer registro que contenha a string. Sem o Like teria que passar o nome exatamente como está no registro, por exemplo ‘João da Silva’

# Atividade 1: Crie sua tabela para representar este formulário

| Carga horária                                                                                                                                                                                                   | Semestre | Ano  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|------|
| 160h                                                                                                                                                                                                            | 2º       | 2020 |
| Nome dos professores:                                                                                                                                                                                           |          |      |
|                                                                                                                                                                                                                 |          |      |
| Cursos:                                                                                                                                                                                                         |          |      |
| Bacharelado em Ciência da Computação, Bacharelado em Sistemas de Informação, Análise e Desenvolvimento de Sistemas, Banco de Dados, Jogos Digitais, Sistemas para Internet e Gestão de Tecnologia da Informação |          |      |

# Atividade 2: Crie sua tabela para representar este formulário

Nome Aluno

Prova 1

Prova 2

Prova 3

Freq. (%)

Aproveit. Semestral

Aval. Final

ResultFinal

# Vamos Conferir



# Gabarito

- CREATE TABLE `aula27`.`professores` (  
`idProfessores` INT NOT NULL,  
`cargahoraria` INT NULL, `semestre` INT  
NULL, `ano` INT NULL, `nome\_professores`  
LONGTEXT NULL, `cursos` LONGTEXT NULL,  
PRIMARY KEY (`idProfessores`), UNIQUE  
INDEX `idProfessores\_UNIQUE`  
(`idProfessores` ASC) VISIBLE);

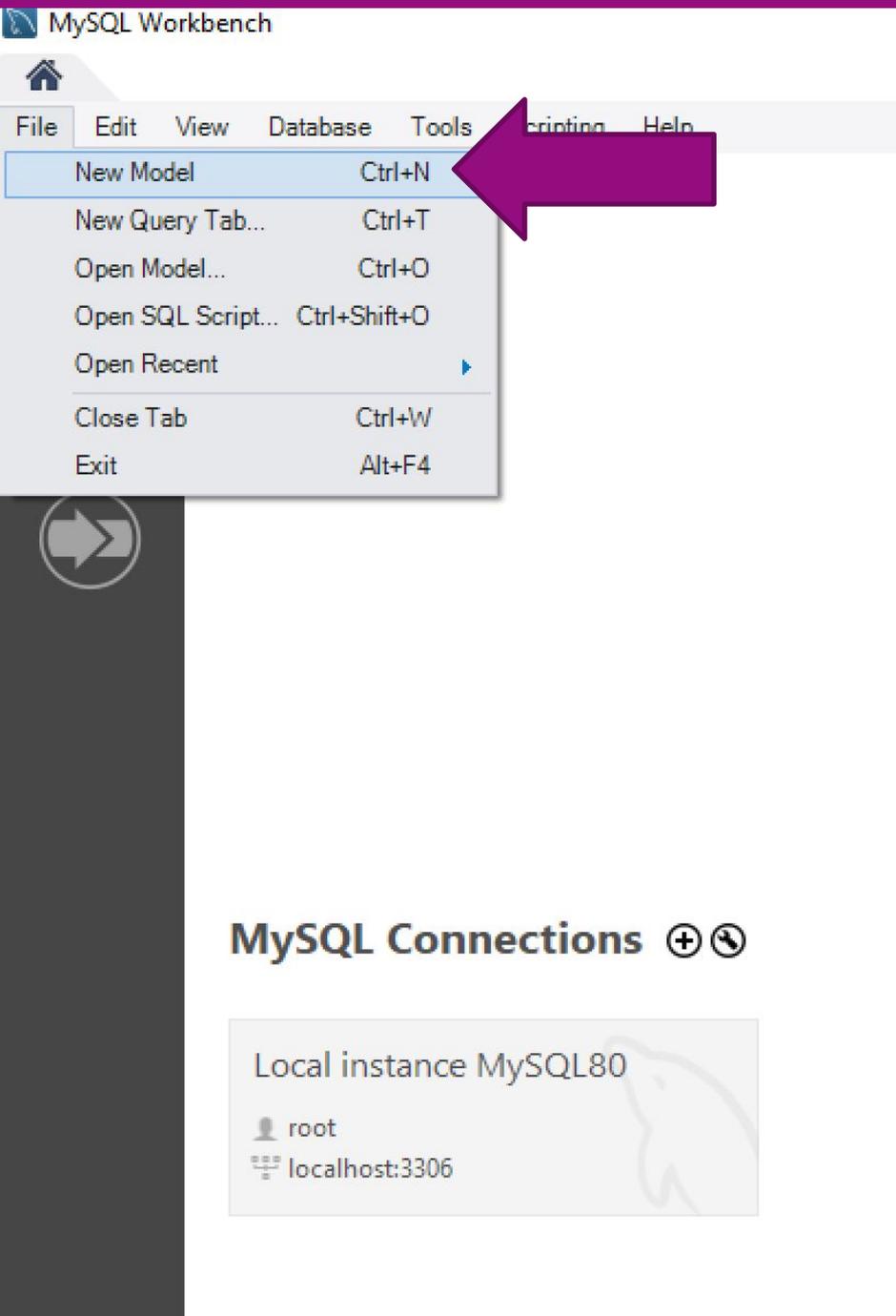
# Gabarito

- CREATE TABLE `aula27`.`aluno` ( `matricula` INT NOT NULL, `prova1` FLOAT NULL, `prova2` FLOAT NULL, `prova3` FLOAT NULL, `frequencia` FLOAT NULL, `aproveitamento\_semestral` FLOAT NULL, `avaliacao\_final` FLOAT NULL, `resultado\_final` VARCHAR(45) NULL, PRIMARY KEY (`matricula`));

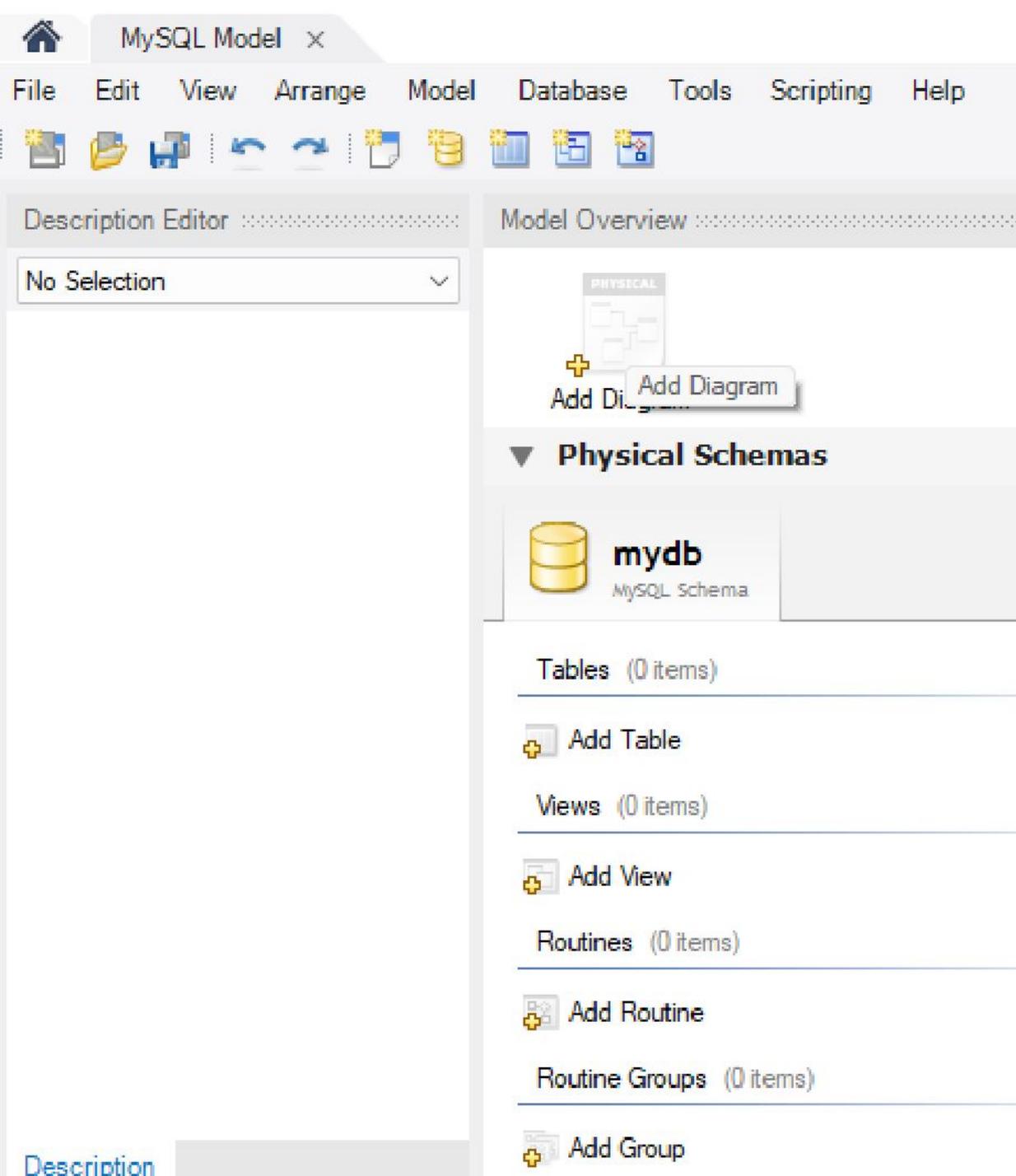
# Busca Ativa

- Leitura do Livro:
  - Barboza, Fabrício Felipe, M. e Pedro Henrique Chagas Freitas. *Modelagem e desenvolvimento de banco de dados*. Disponível em: Minha Biblioteca, Grupo A, 2018





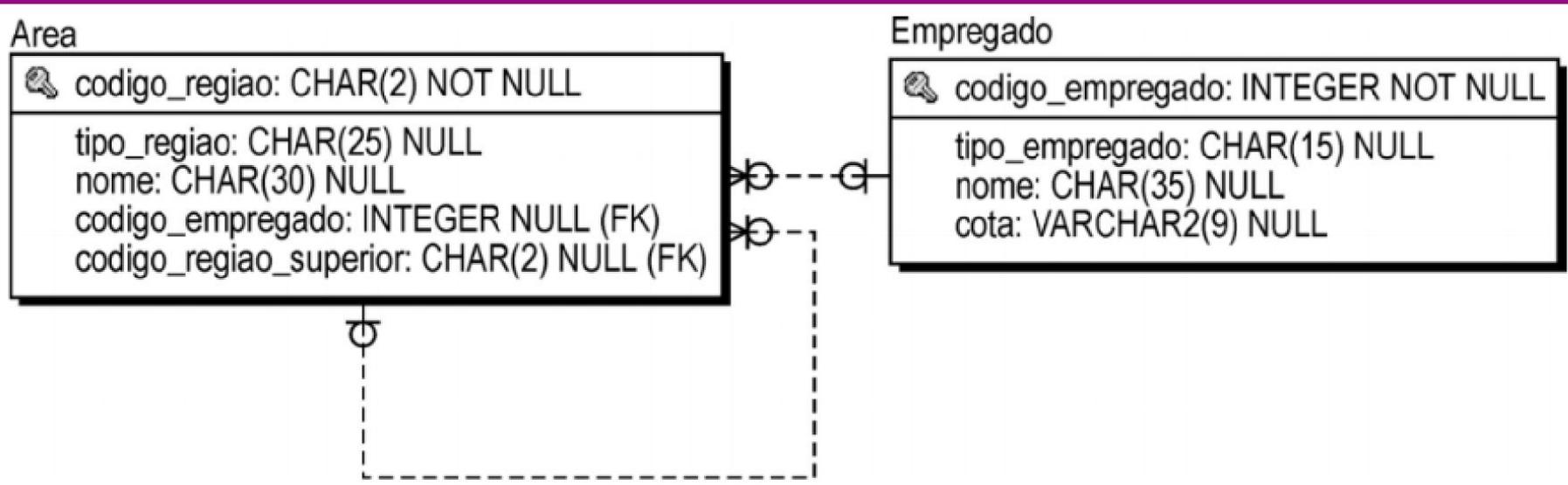
# Agora Vamos Modelar



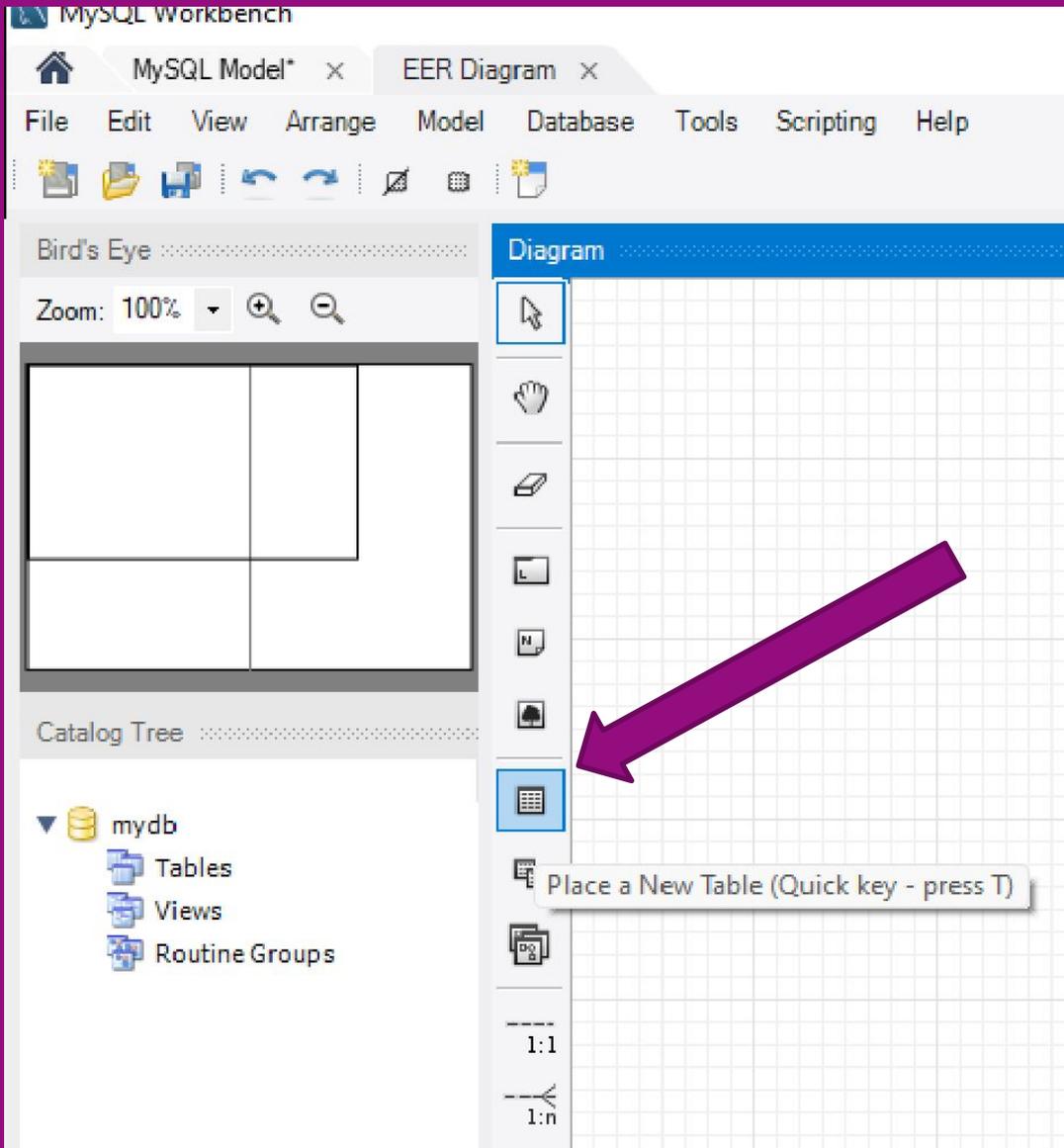
- Adicione um diagrama

# Exemplo Modelo físico

- Página 22 do livro:
  - Machado, Felipe Nery R. *BANCO DE DADOS – PROJETO E IMPLEMENTAÇÃO*. Disponível em: Minha Biblioteca, (4th edição). Editora Saraiva, 2020.



# Vamos Criar o nosso modelo



- Crie uma Tabela

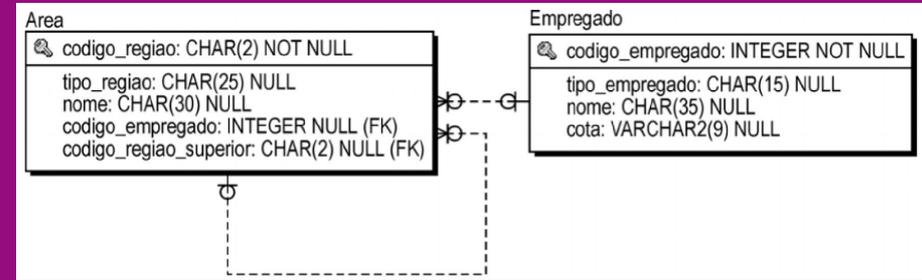




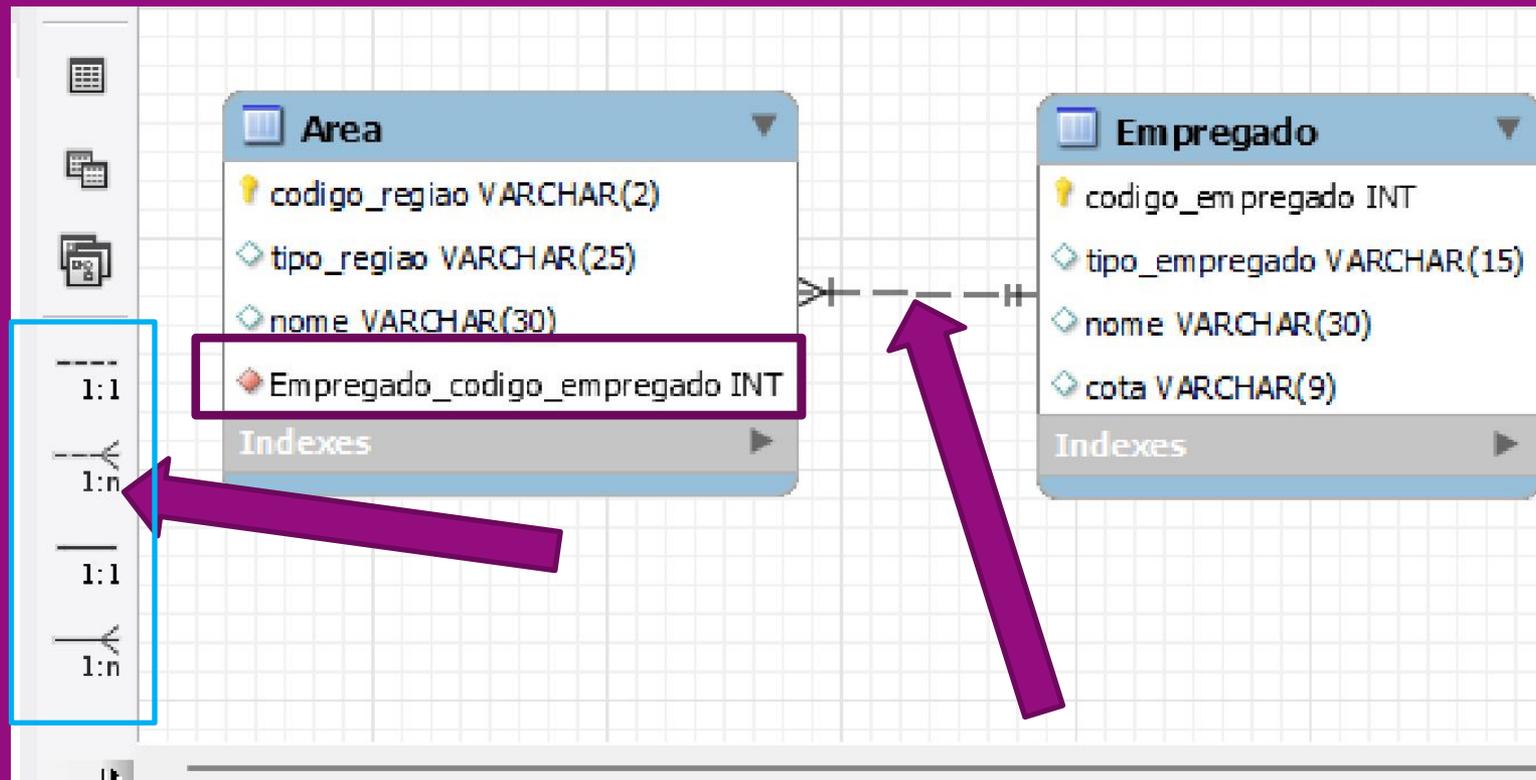




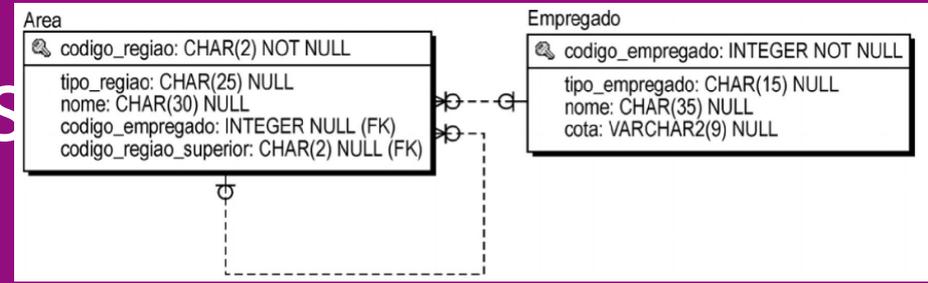
# Relacionamentos



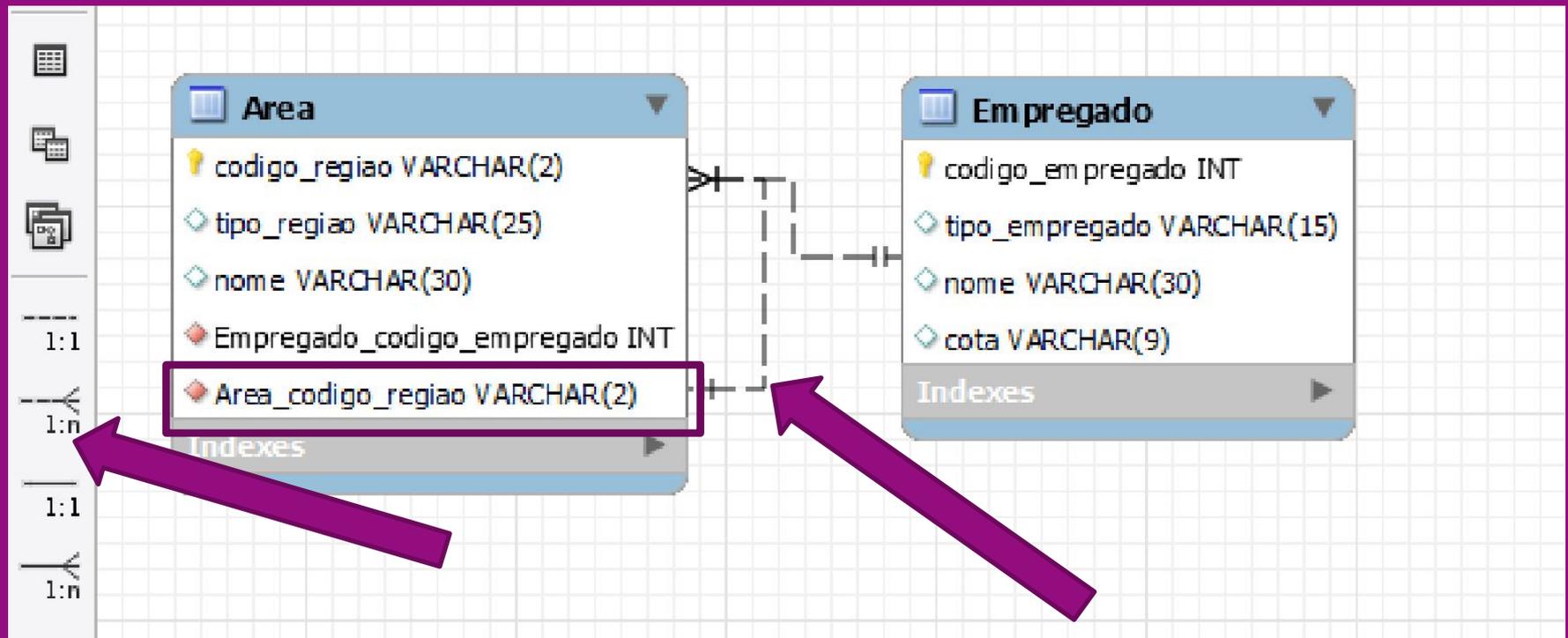
- Note que ao criar o relacionamento, o campo da chave estrangeira é criado automaticamente.



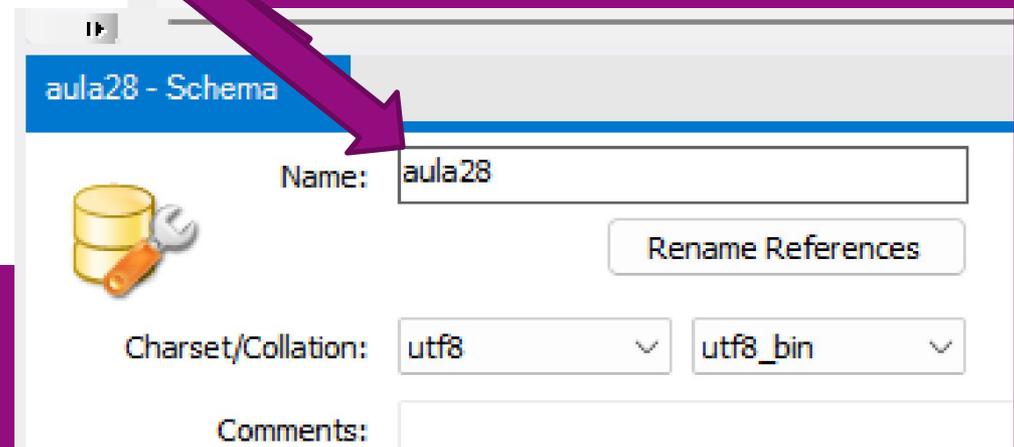
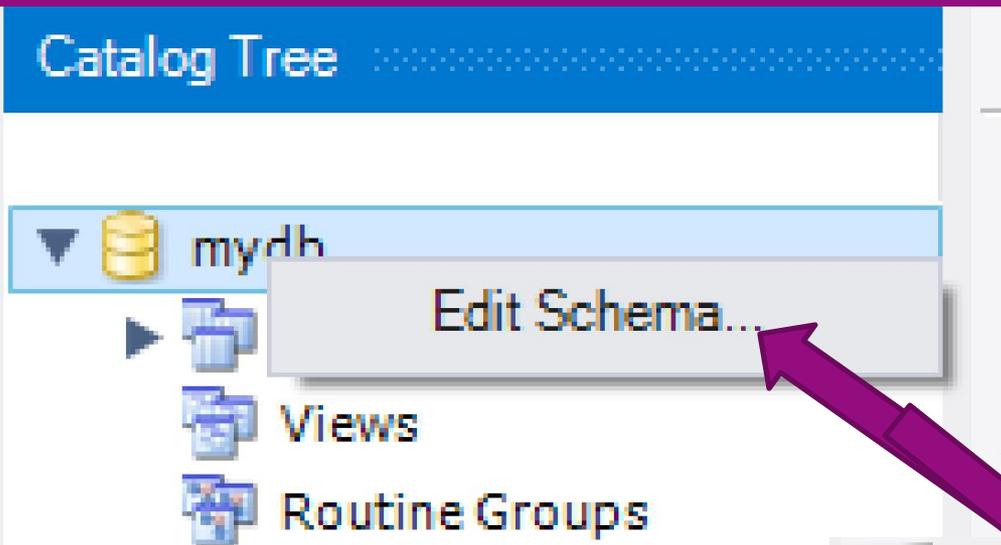
# Relacionamentos



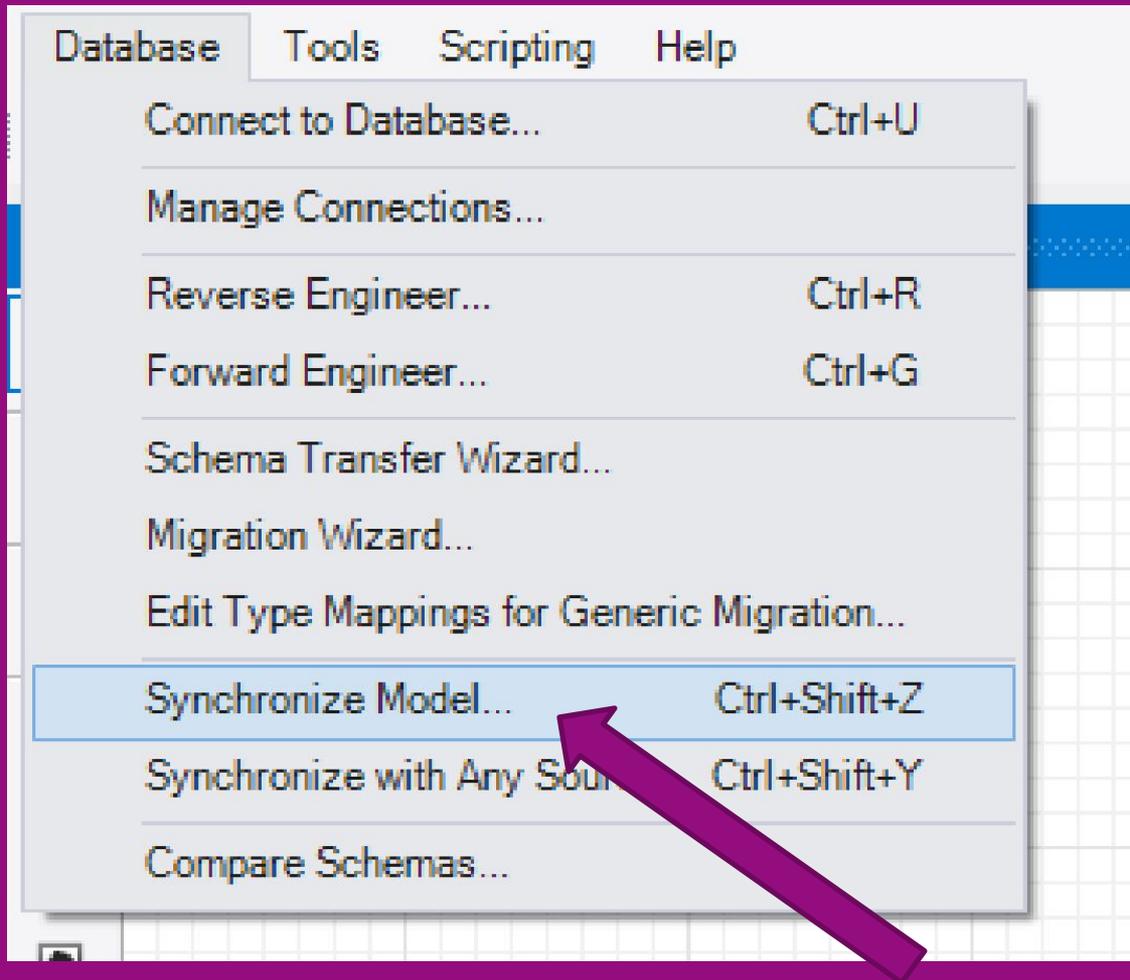
- Crie o segundo relacionamento



# Vamos criar um Schema



# Vamos criar as tabelas no nosso Schema



# Conectar com o *database*

Synchronize Model with Database ✕

**Connection Options**

- Sync Options
- Connect to DBMS
- Select Schemas
- Retrieve Objects
- Select Changes to Apply
- Review DB Changes
- Synchronize Progress

**Set Parameters for Connecting to a DBMS**

Stored Connection:  ▼ Select from saved connection settings

Connection Method:  ▼ Method to use to connect to the RDBMS

Parameters  SSL  Advanced

|                 |                                                   |                                      |                                                               |                                                                      |
|-----------------|---------------------------------------------------|--------------------------------------|---------------------------------------------------------------|----------------------------------------------------------------------|
| Hostname:       | <input type="text" value="localhost"/>            | Port:                                | <input type="text" value="3306"/>                             | Name or IP address of the server host - and TCP/IP port.             |
| Username:       | <input type="text" value="root"/>                 |                                      |                                                               | Name of the user to connect with.                                    |
| Password:       | <input type="button" value="Store in Vault ..."/> | <input type="button" value="Clear"/> | The user's password. Will be requested later if it's not set. |                                                                      |
| Default Schema: | <input type="text"/>                              |                                      |                                                               | The schema to use as default schema. Leave blank to select it later. |

# Seleccionar o schema

Synchronize Model with Database

- Connection Options
- Sync Options
- Connect to DBMS
- Select Schemas**
- Retrieve Objects
- Select Changes to Apply
- Review DB Changes
- Synchronize Progress

### Select the Schemas to be Synchronized

 **Select the Schemata to be Synchronized:**

|                                     | Model Schema                                                                             | RDBMS Schema |                            |
|-------------------------------------|------------------------------------------------------------------------------------------|--------------|----------------------------|
| <input checked="" type="checkbox"/> |  aula28 | aula28       | schema not found in target |

Override target schema to be synchronized with:

The schemata from your model are missing from the target.  
If you are creating them for the first time use the Forward Engineer function.

Connection Options

Sync Options

Connect to DBMS

Select Schemas

Retrieve Objects

Select Changes to Apply

**Review DB Changes**

Synchronize Progress

**Preview Database Changes to be Applied**

```
1 -- MySQL Workbench Synchronization
2 -- Generated: 2022-05-27 09:49
3 -- Model: New Model
4 -- Version: 1.0
5 -- Project: Name of the project
6 -- Author: RIHS
7
8 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
9 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
10 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO
11
12 CREATE SCHEMA IF NOT EXISTS `aula28` DEFAULT CHARACTER SET utf8 ;
13
14 CREATE TABLE IF NOT EXISTS `aula28`.`Area` (
15   `codigo_regiao` VARCHAR(2) NOT NULL,
16   `tipo_regiao` VARCHAR(25) NULL DEFAULT NULL,
17   `nome` VARCHAR(30) NULL DEFAULT NULL,
18   `Empregado_codigo_empregado` INT(11) NOT NULL,
19   `Area_codigo_regiao` VARCHAR(2) NOT NULL,
20   PRIMARY KEY (`codigo_regiao`),
21   INDEX `fk_Area_Empregado_idx` (`Empregado_codigo_empregado` ASC) VISIBLE,
22   INDEX `fk_Area_Area1_idx` (`Area_codigo_regiao` ASC) VISIBLE,
23   CONSTRAINT `fk_Area_Empregado`
24     FOREIGN KEY (`Empregado_codigo_empregado`)
25     REFERENCES `aula28`.`Empregado` (`codigo_empregado`)
26     ON DELETE NO ACTION
```

Save to File...

Copy to Clipboard

 Skip DB changes and update model only

Back

Execute &gt;

Cancel

# Create Table a partir do modelo

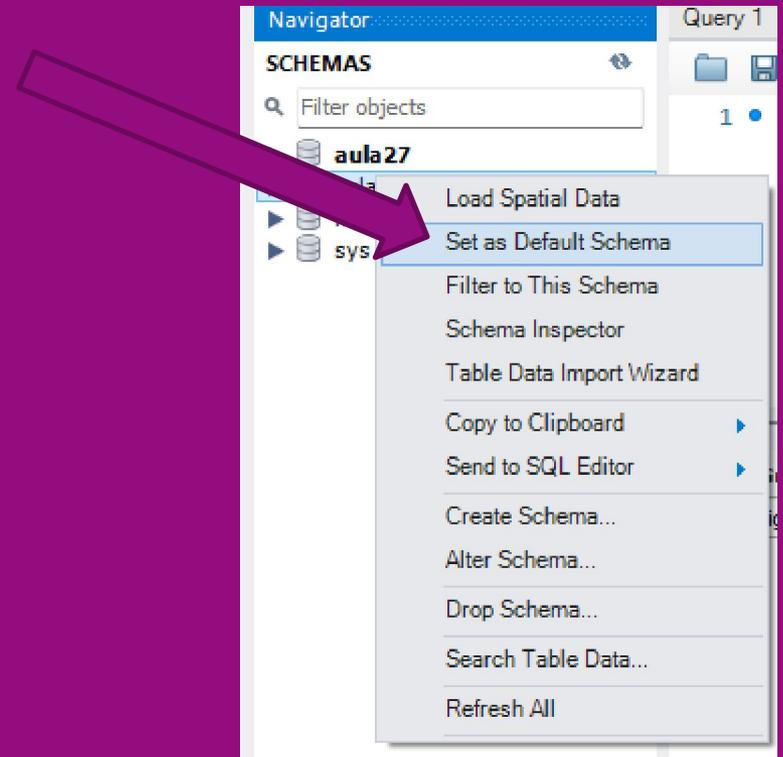
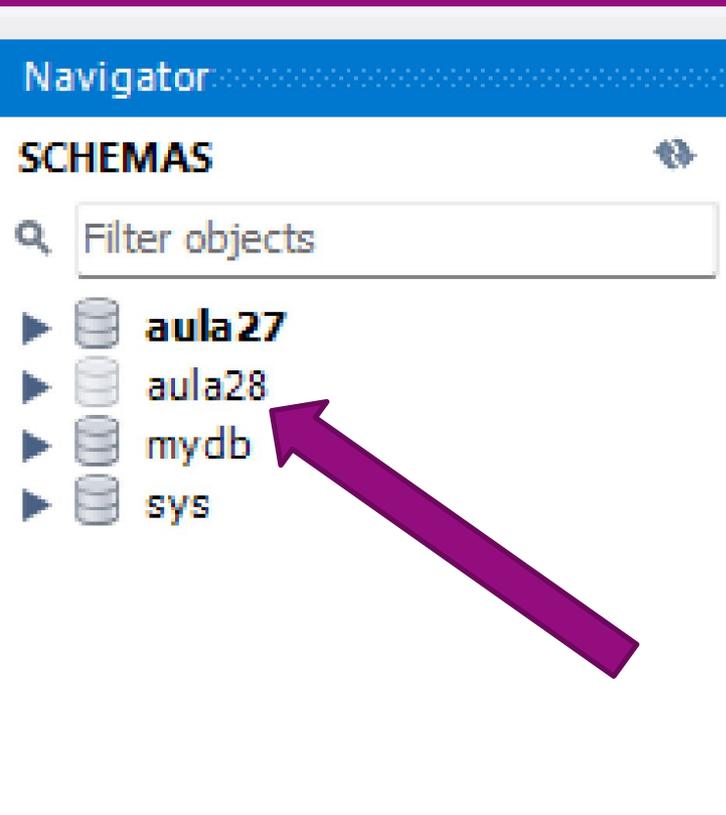
```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';CREATE SCHEMA IF NOT EXISTS `aula28` DEFAULT CHARACTER SET utf8 ;
```

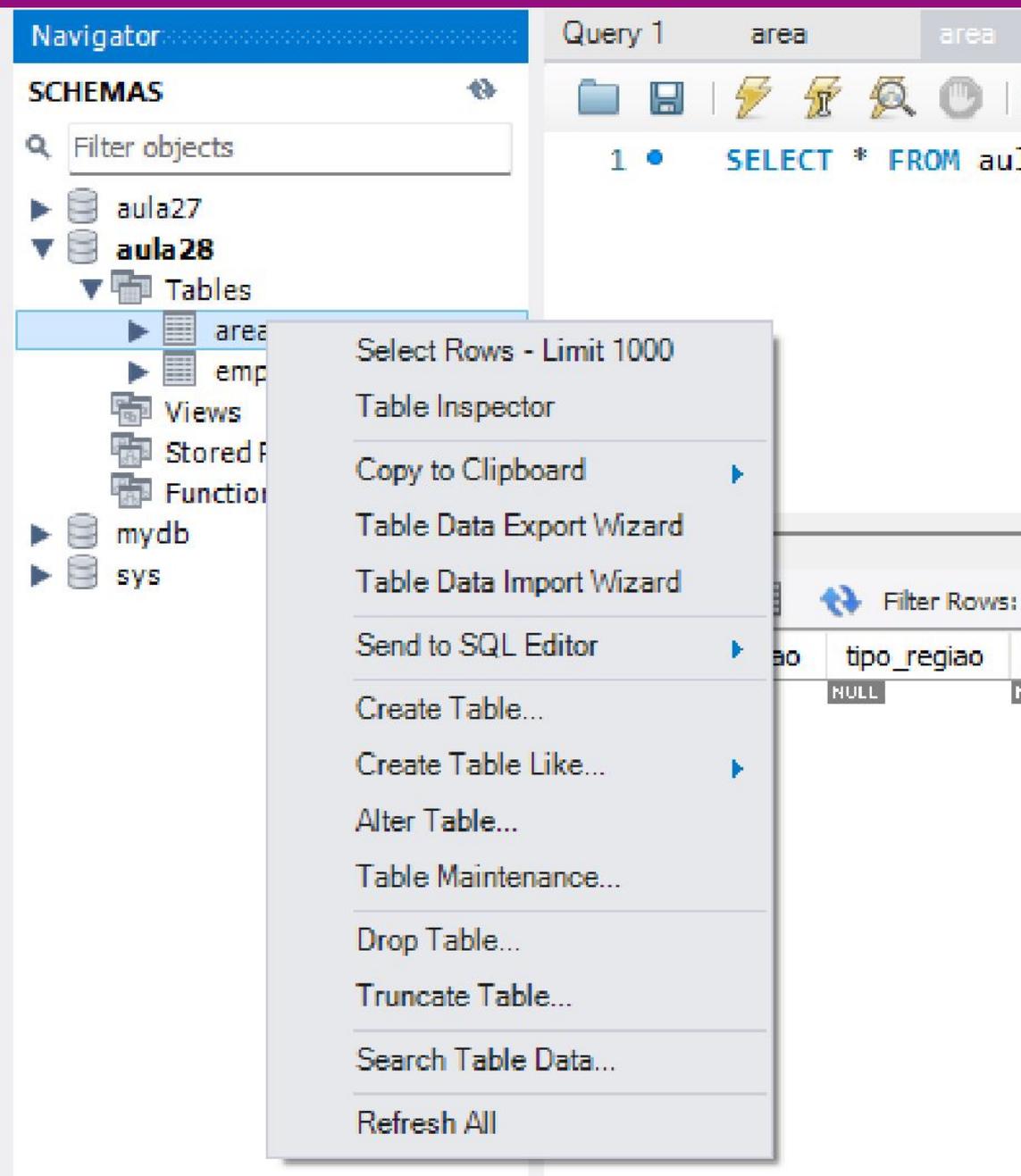
```
CREATE TABLE IF NOT EXISTS `aula28`.`Area` ( `codigo_regiao` VARCHAR(2) NOT NULL, `tipo_regiao` VARCHAR(25) NULL DEFAULT NULL, `nome` VARCHAR(30) NULL DEFAULT NULL, `Empregado_codigo_empregado` INT(11) NOT NULL, `Area_codigo_regiao` VARCHAR(2) NOT NULL, PRIMARY KEY (`codigo_regiao`), INDEX `fk_Area_Empregado_idx` (`Empregado_codigo_empregado` ASC) VISIBLE, INDEX `fk_Area_Area1_idx` (`Area_codigo_regiao` ASC) VISIBLE, CONSTRAINT `fk_Area_Empregado` FOREIGN KEY (`Empregado_codigo_empregado`) REFERENCES `aula28`.`Empregado` (`codigo_empregado`) ON DELETE NO ACTION ON UPDATE NO ACTION, CONSTRAINT `fk_Area_Area1` FOREIGN KEY (`Area_codigo_regiao`) REFERENCES `aula28`.`Area` (`codigo_regiao`) ON DELETE NO ACTION ON UPDATE NO ACTION)ENGINE = InnoDBDEFAULT CHARACTER SET = utf8;
```

```
CREATE TABLE IF NOT EXISTS `aula28`.`Empregado` ( `codigo_empregado` INT(11) NOT NULL, `tipo_empregado` VARCHAR(15) NULL DEFAULT NULL, `nome` VARCHAR(30) NULL DEFAULT NULL, `cota` VARCHAR(9) NULL DEFAULT NULL, PRIMARY KEY (`codigo_empregado`))ENGINE = InnoDBDEFAULT CHARACTER SET = utf8;
```

```
SET SQL_MODE=@OLD_SQL_MODE;SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

# Banco Criado





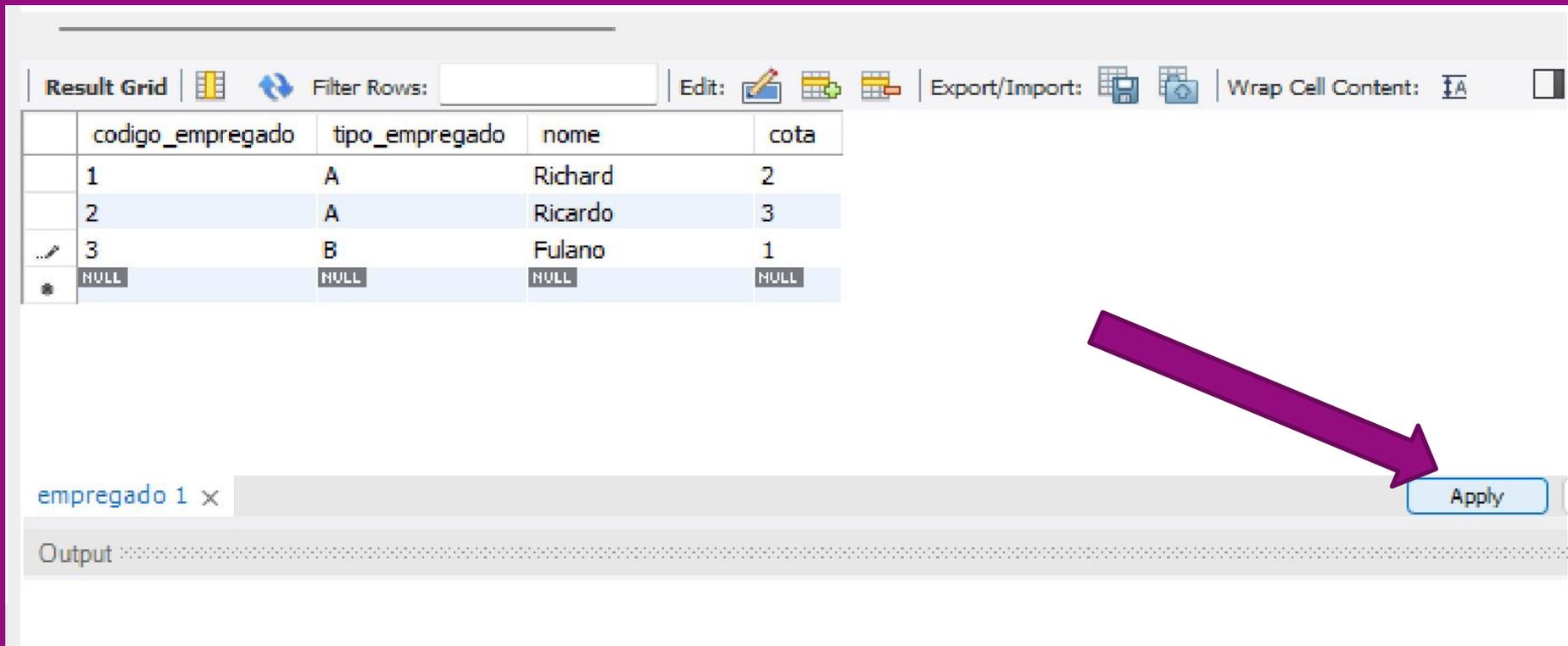
- Selecione uma das tabelas
- E clique em Select Rows

# SQL

## *Structure Query Language*

- ▶ **INSERT INTO** <nome da tabela>  
(<nome da(s) coluna(s)>)  
**VALUES** (<valores>);

# Vamos Preencher os dados



The screenshot shows a data grid interface with a toolbar at the top. The toolbar includes a 'Result Grid' button, a 'Filter Rows' input field, an 'Edit' button, and 'Export/Import' and 'Wrap Cell Content' buttons. The data grid contains the following table:

|  | codigo_empregado | tipo_empregado | nome    | cota |
|--|------------------|----------------|---------|------|
|  | 1                | A              | Richard | 2    |
|  | 2                | A              | Ricardo | 3    |
|  | 3                | B              | Fulano  | 1    |
|  | NULL             | NULL           | NULL    | NULL |

Below the table, there is a tab labeled 'empregado 1 x' and an 'Apply' button. A large purple arrow points to the 'Apply' button. Below the 'Apply' button is an 'Output' section.

# A ferramenta gerou os Inserts

```
INSERT INTO `aula28`.`empregado` (`codigo_empregado`,  
`tipo_empregado`, `nome`, `cota`) VALUES ('1', 'A',  
'Richard', '2');
```

```
INSERT INTO `aula28`.`empregado` (`codigo_empregado`,  
`tipo_empregado`, `nome`, `cota`) VALUES ('2', 'A',  
'Ricardo', '3');
```

```
INSERT INTO `aula28`.`empregado` (`codigo_empregado`,  
`tipo_empregado`, `nome`, `cota`) VALUES ('3', 'B',  
'Fulano', '1');
```



# Alter Table

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

### Review the SQL Script to be Applied on the Database

Online DDL

Algorithm:  Lock Type:

```
1 ALTER TABLE `aula28`.`area`  
2 DROP FOREIGN KEY `fk_Area_Area1`;  
3 ALTER TABLE `aula28`.`area`  
4 CHANGE COLUMN `Area_codigo_regiao` `Area_codigo_regiao` VARCHAR(2) NULL ;  
5 ALTER TABLE `aula28`.`area`  
6 ADD CONSTRAINT `fk_Area_Area1`  
7 FOREIGN KEY (`Area_codigo_regiao`)  
8 REFERENCES `aula28`.`area` (`codigo_regiao`);  
9
```

Back Apply Cancel

# Alter Table

- `ALTER TABLE `aula28`.`area` DROP FOREIGN KEY `fk_Area_Area1`;`
- `ALTER TABLE `aula28`.`area` CHANGE COLUMN `Area_codigo_regiao` `Area_codigo_regiao` VARCHAR(2) NULL ;ALTER TABLE `aula28`.`area` ADD CONSTRAINT `fk_Area_Area1` FOREIGN KEY (`Area_codigo_regiao`) REFERENCES `aula28`.`area` (`codigo_regiao`);`

# Agora Podemos preencher

Result Grid |  Filter Rows:  | Edit:    | Export/Import:   | Wrap Cell Content: 

|                                                                                 | codigo_regiao | tipo_regiao | nome        | Empregado_codigo_empregado | Area_codigo_regiao |
|---------------------------------------------------------------------------------|---------------|-------------|-------------|----------------------------|--------------------|
|                                                                                 | SC            | estadual    | TI          | 1                          | NULL               |
|                                                                                 | FL            | municipal   | Programação | 2                          | SC                 |
|   | PL            | municipal   | Testador    | 3                          | SC                 |
|  | NULL          | NULL        | NULL        | NULL                       | NULL               |

area 1 x  Apply

Output

# INSERT

- `INSERT INTO `aula28`.`area` (`codigo_regiao`, `tipo_regiao`, `nome`, `Empregado_codigo_empregado`) VALUES ('SC', 'estadual', 'TI', '1');`
- `INSERT INTO `aula28`.`area` (`codigo_regiao`, `tipo_regiao`, `nome`, `Empregado_codigo_empregado`, `Area_codigo_regiao`) VALUES ('FL', 'municipal', 'Programação', '2', 'SC');`
- `INSERT INTO `aula28`.`area` (`codigo_regiao`, `tipo_regiao`, `nome`, `Empregado_codigo_empregado`, `Area_codigo_regiao`) VALUES ('PL', 'municipal', 'Testador', '3', 'SC');`

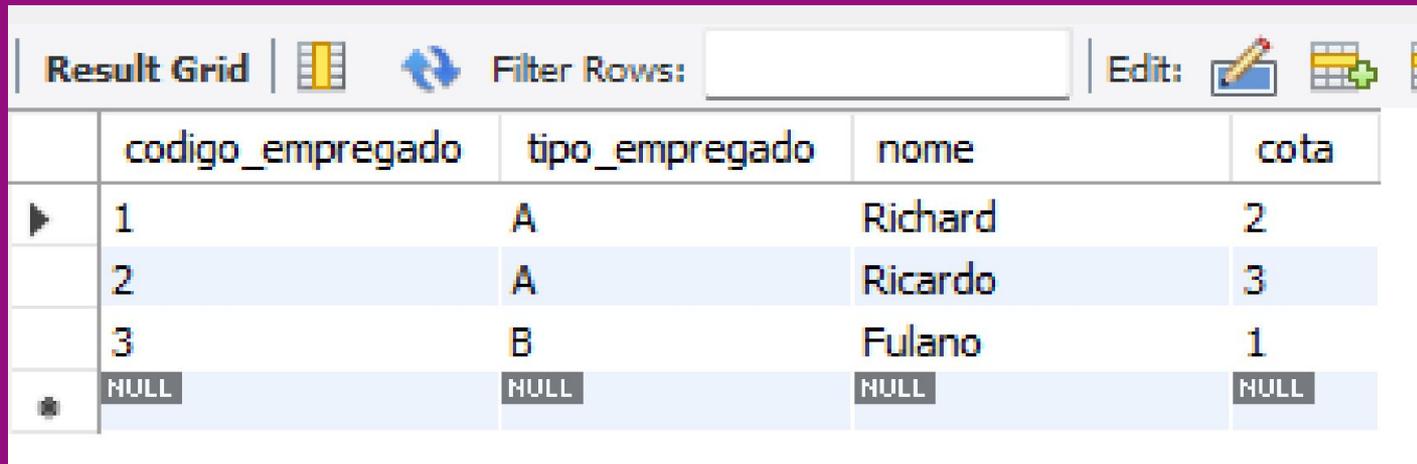
# SQL

## *Structure Query Language*

- ▶ **SELECT**: especifica as colunas da tabela que queremos selecionar.
- ▶ **FROM**: especifica as tabelas.
- ▶ **WHERE**: especifica a condição de seleção das linhas.

# Exemplo

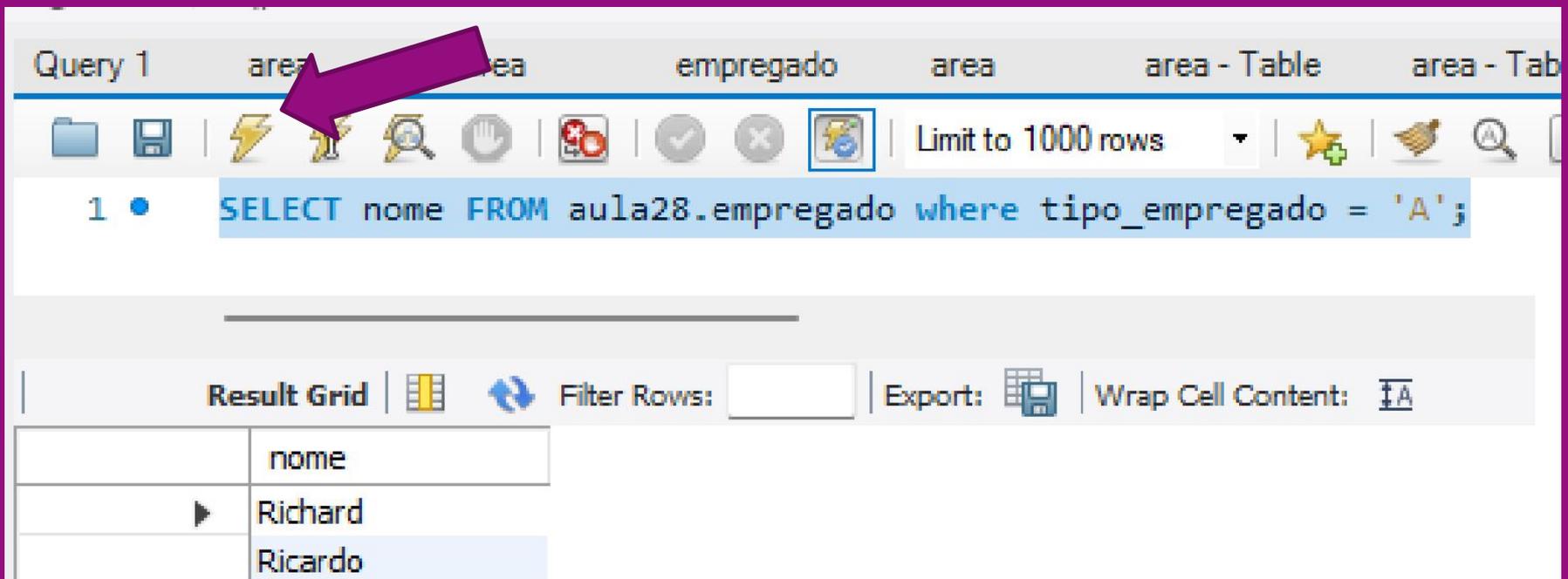
- `SELECT * FROM aula28.empregado;`
- Retorna todos



|   | codigo_empregado | tipo_empregado | nome    | cota |
|---|------------------|----------------|---------|------|
| ▶ | 1                | A              | Richard | 2    |
|   | 2                | A              | Ricardo | 3    |
|   | 3                | B              | Fulano  | 1    |
| ● | NULL             | NULL           | NULL    | NULL |

# Apenas os nomes dos empregados tipo A

- `SELECT` nome `FROM` aula28.empregado `where` tipo\_empregado = 'A';

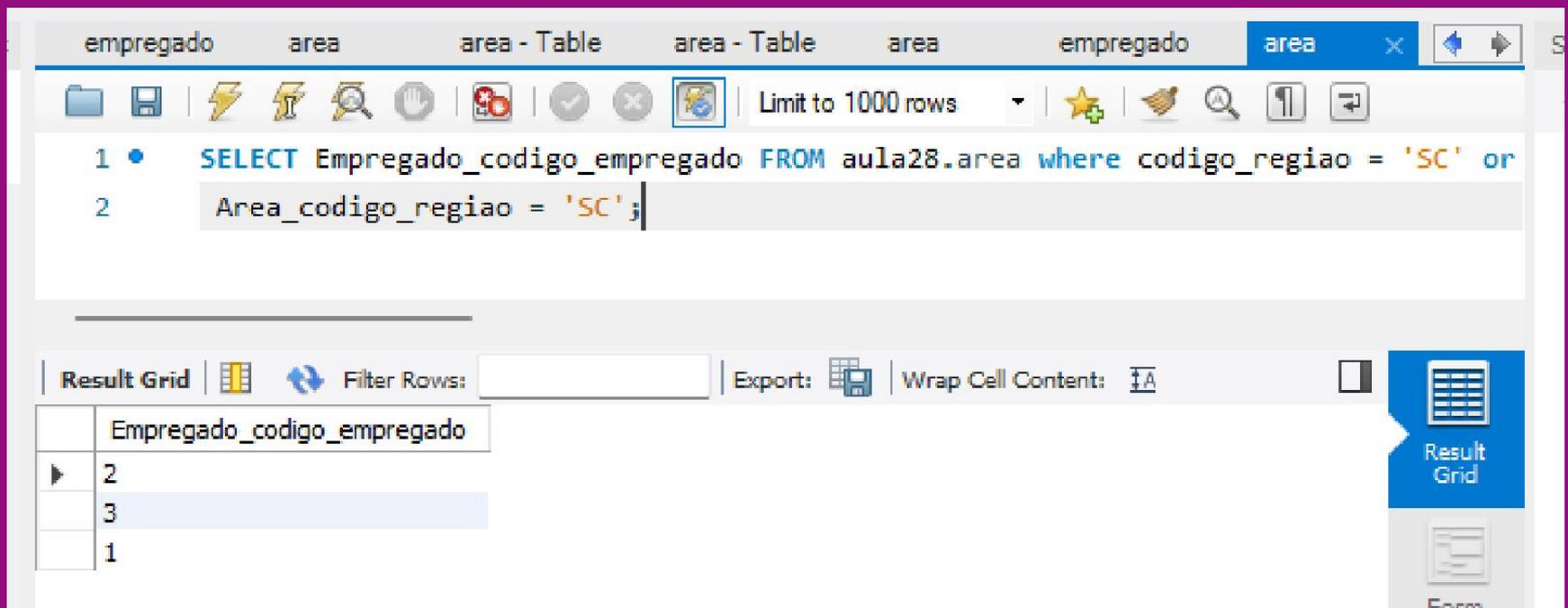


The screenshot shows a database query tool interface. At the top, there are several tabs labeled 'Query 1', 'area', 'empregado', 'area - Table', and 'area - Tab'. Below the tabs is a toolbar with various icons, including a lightning bolt, a magnifying glass, a hand, a checkmark, a close button, and a 'Limit to 1000 rows' dropdown. The main area displays a SQL query: `SELECT nome FROM aula28.empregado where tipo_empregado = 'A';`. Below the query is a 'Result Grid' section with a table containing the following data:

|   | nome    |
|---|---------|
| ▶ | Richard |
|   | Ricardo |

# Códigos dos empregados de SC

- **SELECT** Empregado\_codigo\_empregado  
**FROM** aula28.area **where** codigo\_regiao = 'SC'  
**or** Area\_codigo\_regiao = 'SC';



The screenshot shows a SQL query editor with the following query:

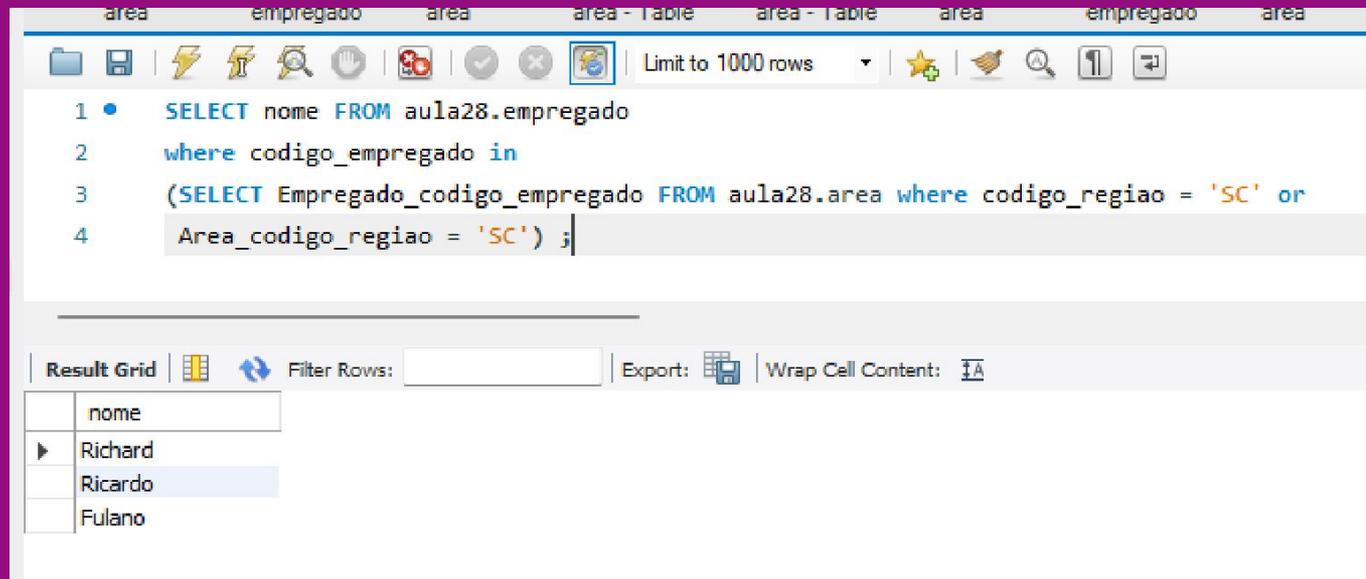
```
1 • SELECT Empregado_codigo_empregado FROM aula28.area where codigo_regiao = 'SC' or  
2 Area_codigo_regiao = 'SC';
```

The result grid below the query shows the following data:

| Empregado_codigo_empregado |
|----------------------------|
| 2                          |
| 3                          |
| 1                          |

# SELECT dentro de SELECT

- **SELECT** nome **FROM** aula28.empregado **where** codigo\_empregado **in** (**SELECT** Empregado\_codigo\_empregado **FROM** aula28.area **where** codigo\_regiao = 'SC' **or** Area\_codigo\_regiao = 'SC') ;



The screenshot shows a database query editor with the following SQL query:

```
1 • SELECT nome FROM aula28.empregado
2   where codigo_empregado in
3   (SELECT Empregado_codigo_empregado FROM aula28.area where codigo_regiao = 'SC' or
4   Area_codigo_regiao = 'SC') ;
```

The results are displayed in a table with the following data:

| nome    |
|---------|
| Richard |
| Ricardo |
| Fulano  |

# Exercício 2

- Faça a consulta dos empregados que contenham a string 'Ric' no nome
- Faça a consulta dos empregados que contenham cotas abaixo ou igual a 2

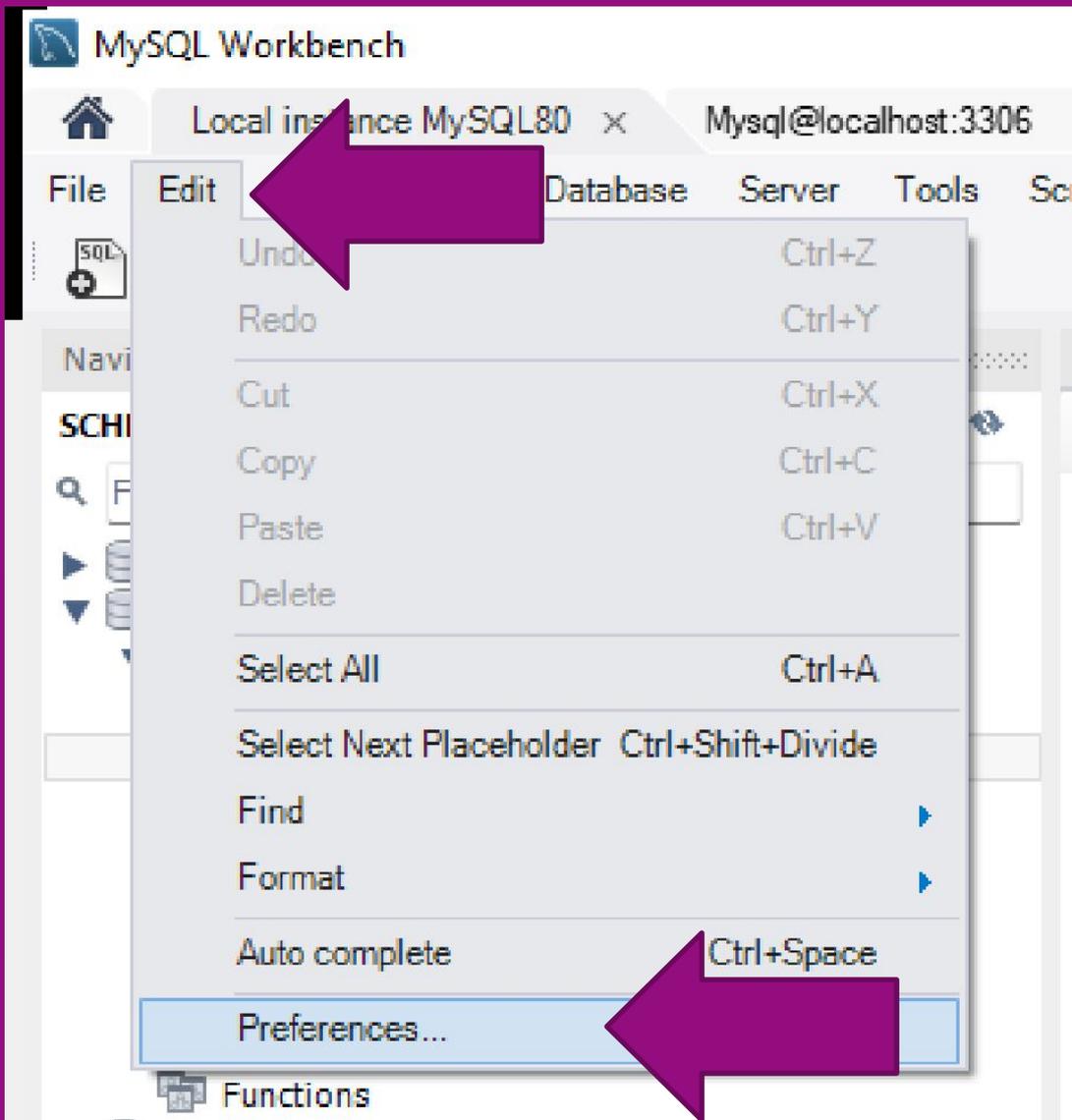
# Exercício 3

- A. Faça a consulta das áreas que contenham a letra T
- B. Faça a consulta das áreas que são municipais

# Alterar

- **UPDATE** <nome da tabela>  
    **SET** <nome da(s) coluna(s)> = valor  
    **WHERE** <condição>

# Mas antes de testar Vamos tirar desmarcar o Safe mode



- General Editors
- SQL Editor
- Query Editor
- Object Editors
- SQL Execution
- Administration
- Modeling
- Defaults
- MySQL
- Diagram
- Appearance
- Fonts & Colors
- SSH
- Others

Auto-save scripts interval:

10 seconds

Interval to perform auto-saving on all open script tabs. The scripts will be restored from the last auto-saved version if Workbench unexpectedly quits.

- Create new tabs as Query tabs instead of File
- Restore expanded state of the active schema objects

- Desmarque a opção  
Safe Updates....

## Sidebar

- Show Schema Contents in Schema Tree
- Show Metadata and Internal Schemas

## MySQL Session

DBMS connection keep-alive interval (in

600

Time interval between sending keep-alive messages to DBMS. Set to 0 to not send keep-alive messages.

DBMS connection read timeout (in seconds):

30

The maximum amount of time the query can take to return data from the DBMS. Set 0 to skip the read timeout.

DBMS connection timeout interval (in seconds):

60

Maximum time to wait before a connection attempt is aborted.

## Other

Internal Workbench Schema:

.mysqlworkbench

This schema will be used by MySQL Workbench to store information required for certain operations.

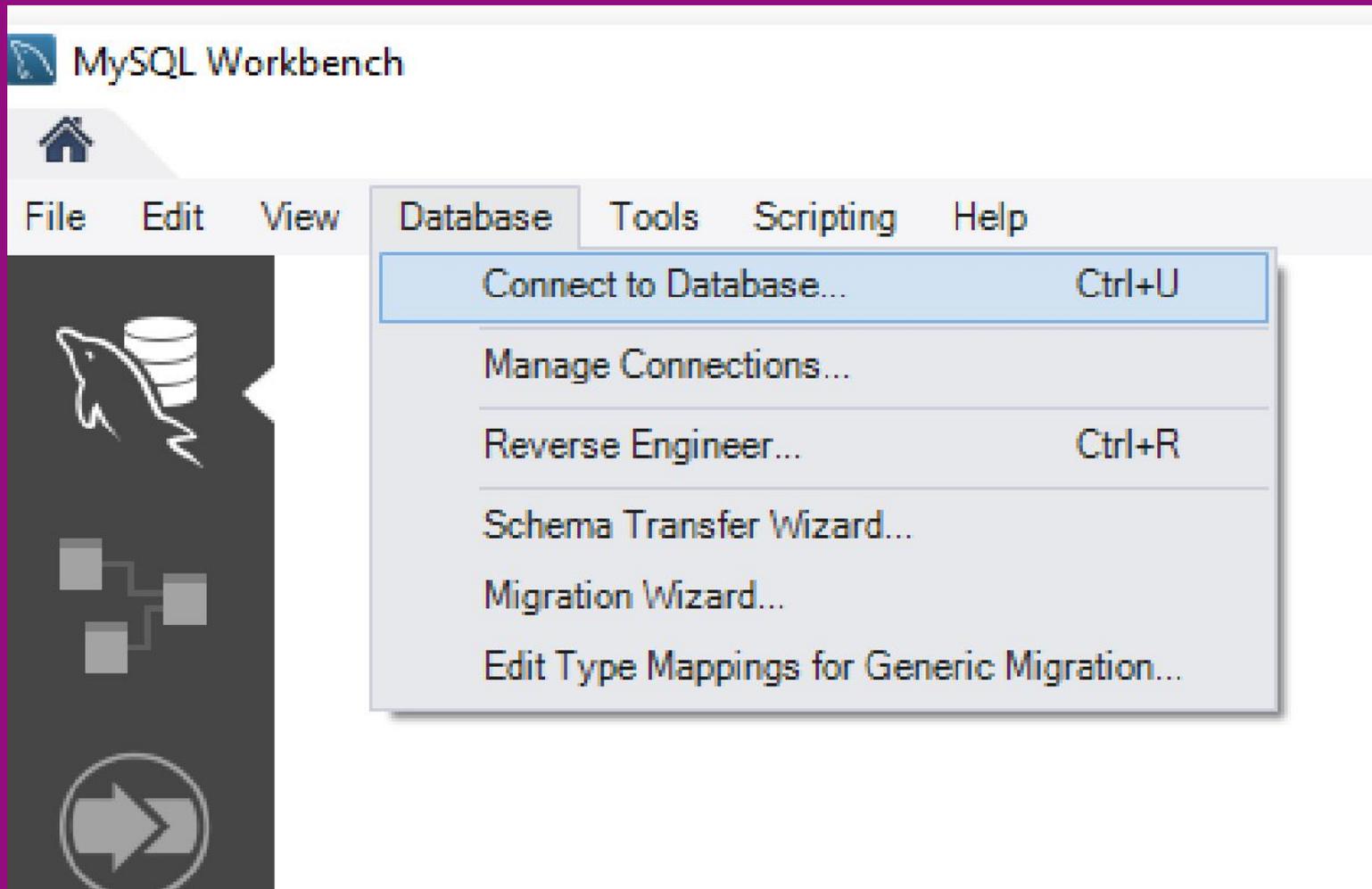
- Safe Updates (rejects UPDATES and DELETES with no restrictions)

statements that do not use a key in the WHERE clause or a LIMIT clause. This makes it possible to catch UPDATE or DELETE statements where keys are not used.

OK

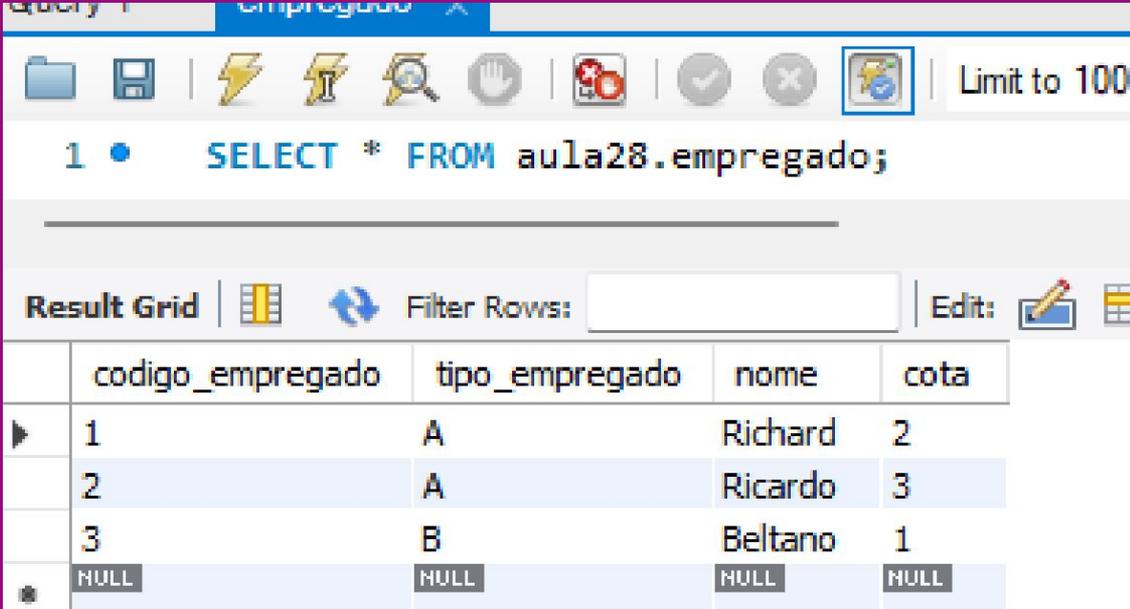
Cancel

# Reconnect ao banco



# UPDATE

- **UPDATE** aula28.empregado **SET** nome = 'Beltano' **where** nome = 'Fulano';



The screenshot shows a database query tool interface. The query editor displays the following SQL statement:

```
1 • SELECT * FROM aula28.empregado;
```

Below the query editor, the results are displayed in a table format. The table has four columns: `codigo_empregado`, `tipo_empregado`, `nome`, and `cota`. The results are as follows:

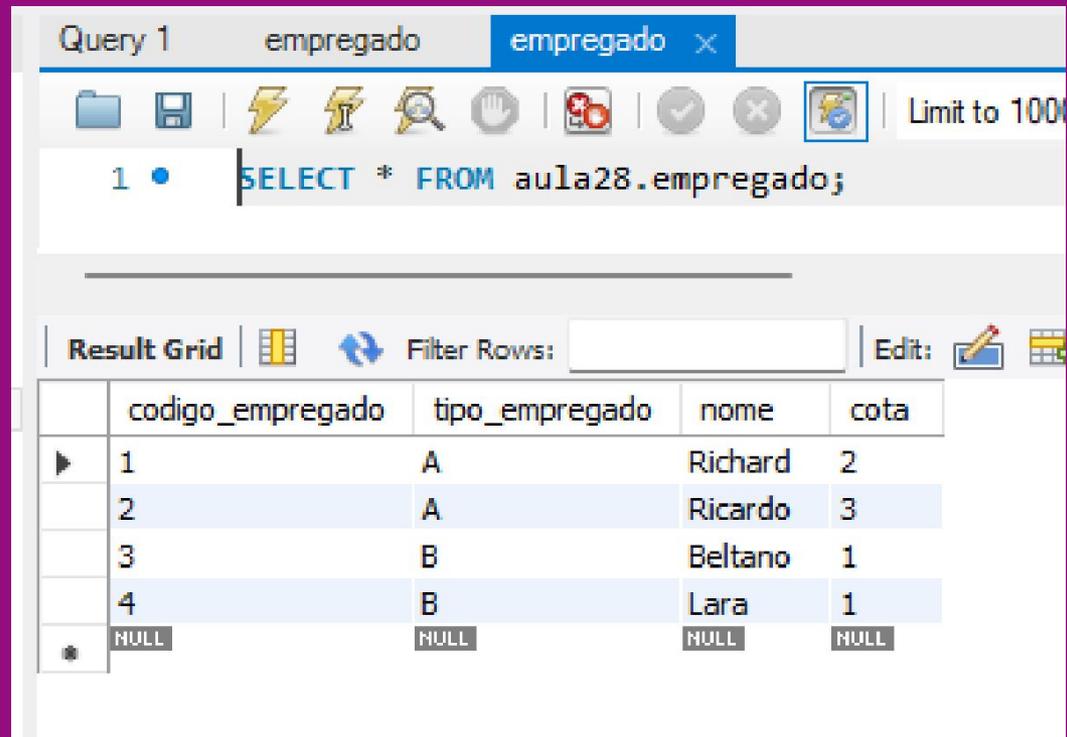
|   | codigo_empregado | tipo_empregado | nome    | cota |
|---|------------------|----------------|---------|------|
| ▶ | 1                | A              | Richard | 2    |
|   | 2                | A              | Ricardo | 3    |
|   | 3                | B              | Beltano | 1    |
| • | NULL             | NULL           | NULL    | NULL |

# Lembrando do INSERT

- **INSERT INTO** <nome da tabela> (<nome da(s) coluna(s)>)  
**VALUES** (<valores>);

# Inserindo novo registro

- **INSERT INTO** aula28.empregado  
(codigo\_empregado, tipo\_empregado , nome  
, cota ) **VALUES** (4, 'B', 'Lara', 1);



Query 1 empregado empregado x

1 • `SELECT * FROM aula28.empregado;`

Result Grid | Filter Rows: | Edit:

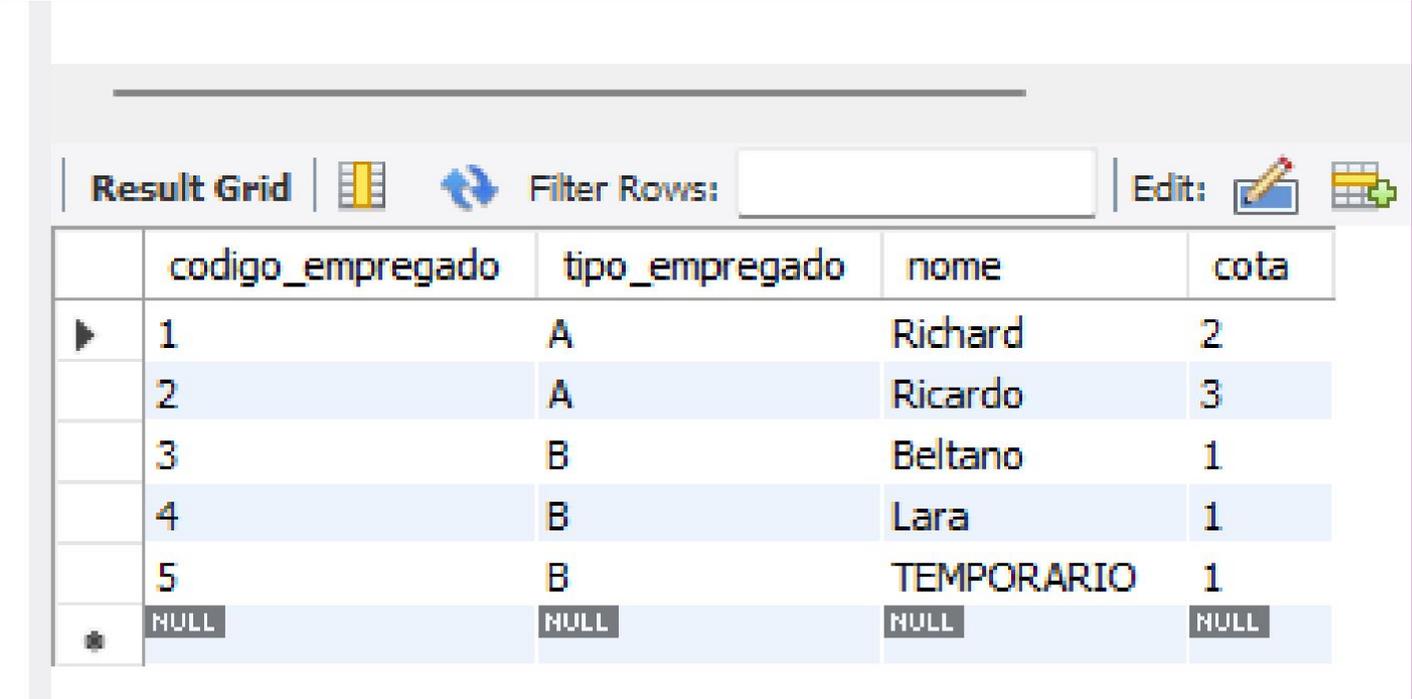
|   | codigo_empregado | tipo_empregado | nome    | cota |
|---|------------------|----------------|---------|------|
| ▶ | 1                | A              | Richard | 2    |
|   | 2                | A              | Ricardo | 3    |
|   | 3                | B              | Beltano | 1    |
|   | 4                | B              | Lara    | 1    |
| • | NULL             | NULL           | NULL    | NULL |

# Exercício 4

- Adicione um Registro na tabela AREA para o código do novo empregado.
- Mude o nome do empregado Beltrano para a o seu nome.

# VAMOS INSERIR +1

- INSERT INTO aula28.empregado  
(codigo\_empregado, tipo\_empregado , nome  
, cota ) VALUES (5, 'B', 'TEMPORARIO', 1);



Result Grid |   Filter Rows:  | Edit:  

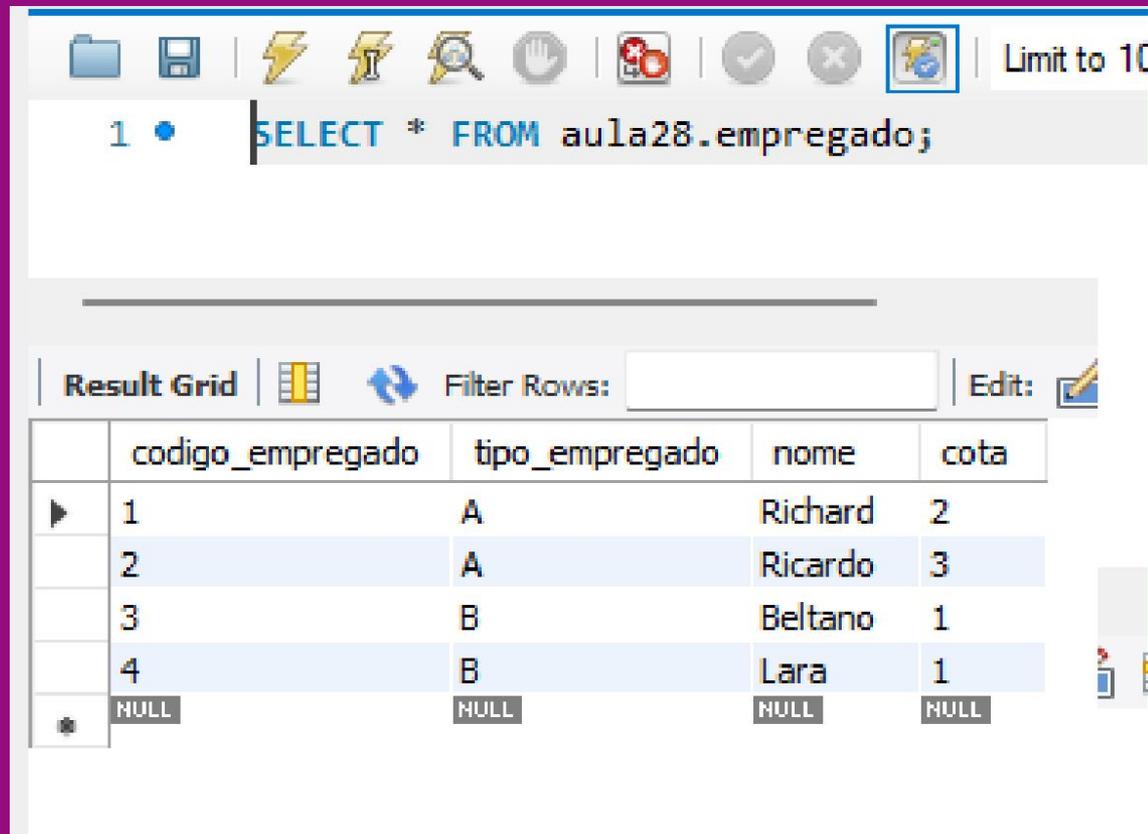
|   | codigo_empregado | tipo_empregado | nome       | cota |
|---|------------------|----------------|------------|------|
| ▶ | 1                | A              | Richard    | 2    |
|   | 2                | A              | Ricardo    | 3    |
|   | 3                | B              | Beltano    | 1    |
|   | 4                | B              | Lara       | 1    |
|   | 5                | B              | TEMPORARIO | 1    |
| ✱ | NULL             | NULL           | NULL       | NULL |

# DELETE

- **DELETE FROM** <nome da tabela> **WHERE** <condição>;

# DELETE

- **DELETE FROM** aula28.empregado **where** codigo\_empregado = 5;



The screenshot shows a database query tool interface. The top toolbar includes icons for file operations, execution, search, and window management. The query editor contains the following SQL statement:

```
1 • SELECT * FROM aula28.empregado;
```

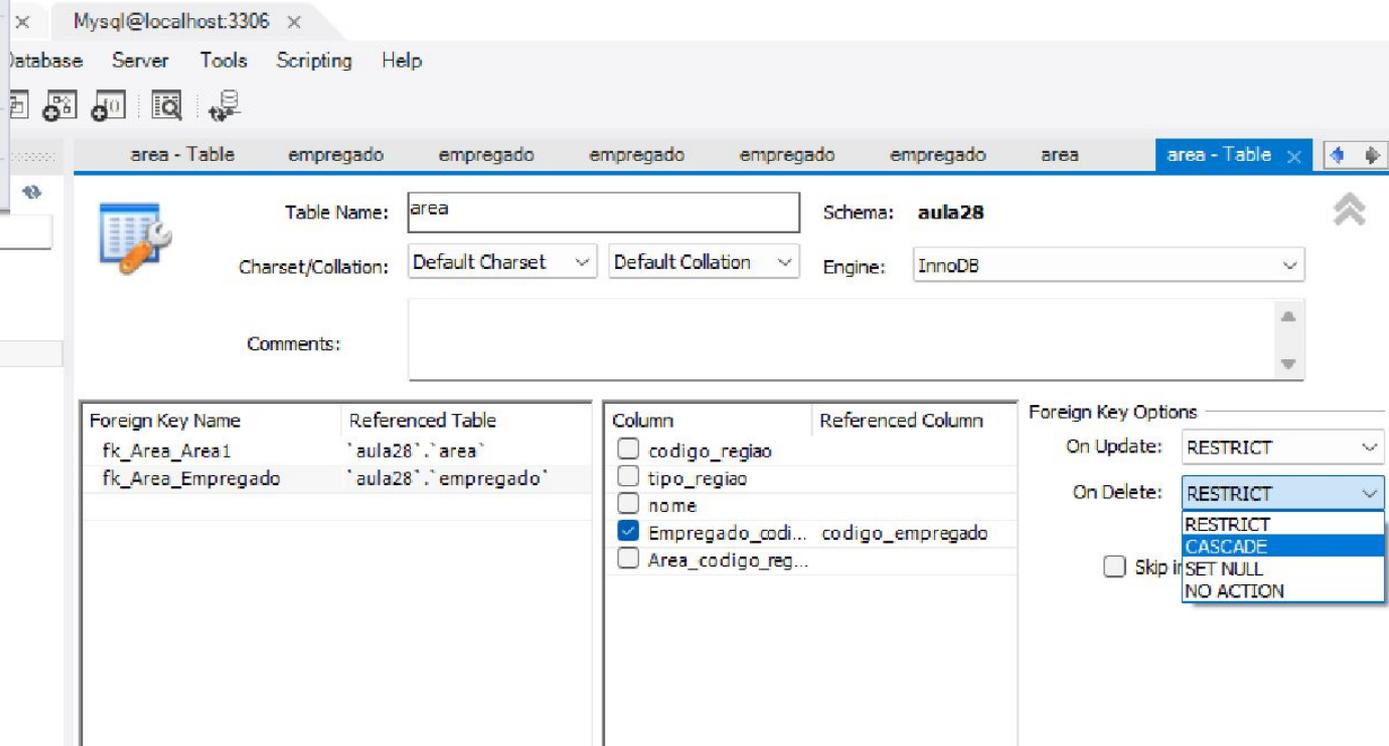
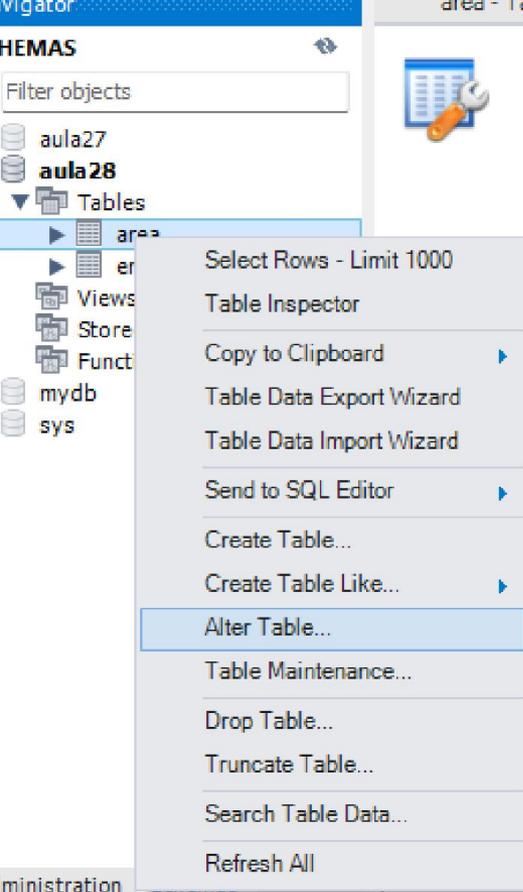
Below the query editor is the "Result Grid" section, which displays the results of the query in a table format. The table has four columns: "codigo\_empregado", "tipo\_empregado", "nome", and "cota". The results are as follows:

|   | codigo_empregado | tipo_empregado | nome    | cota |
|---|------------------|----------------|---------|------|
| ▶ | 1                | A              | Richard | 2    |
|   | 2                | A              | Ricardo | 3    |
|   | 3                | B              | Beltano | 1    |
|   | 4                | B              | Lara    | 1    |
| * | NULL             | NULL           | NULL    | NULL |

# O que acontece ?

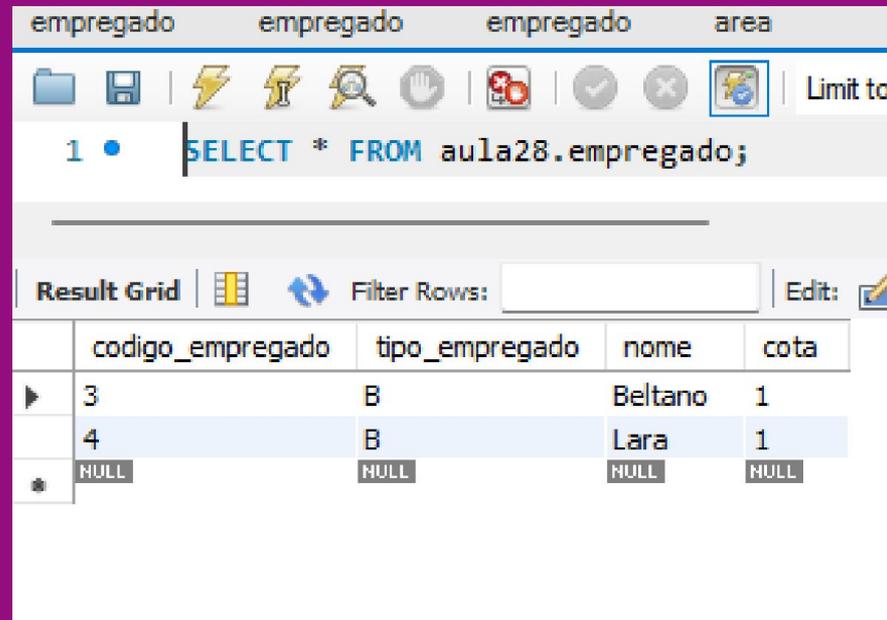
- `DELETE FROM aula28.empregado where codigo_empregado = 2;`
- Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (``aula28`.`area`, CONSTRAINT `fk_Area_Empregado` FOREIGN KEY (`Empregado_codigo_empregado`) REFERENCES `empregado` (`codigo_empregado`)) 0.000 sec`

# Alterar a TABELA



# O que acontece ?

- **DELETE FROM** aula28.empregado **where** codigo\_empregado = 2;
- **DELETE FROM** aula28.empregado **where** codigo\_empregado = 1;



The screenshot shows a database management tool interface. At the top, there are four tabs labeled 'empregado', 'empregado', 'empregado', and 'area'. Below the tabs is a toolbar with various icons. The main area displays a SQL query: `SELECT * FROM aula28.empregado;`. Below the query is a 'Result Grid' section. The grid has a header row with columns: 'codigo\_empregado', 'tipo\_empregado', 'nome', and 'cota'. There are three rows of data: the first row has values 3, B, Beltano, 1; the second row has values 4, B, Lara, 1; and the third row has values NULL, NULL, NULL, NULL. The first two rows are highlighted in light blue.

|   | codigo_empregado | tipo_empregado | nome    | cota |
|---|------------------|----------------|---------|------|
| ▶ | 3                | B              | Beltano | 1    |
|   | 4                | B              | Lara    | 1    |
| * | NULL             | NULL           | NULL    | NULL |

# Exercício 5

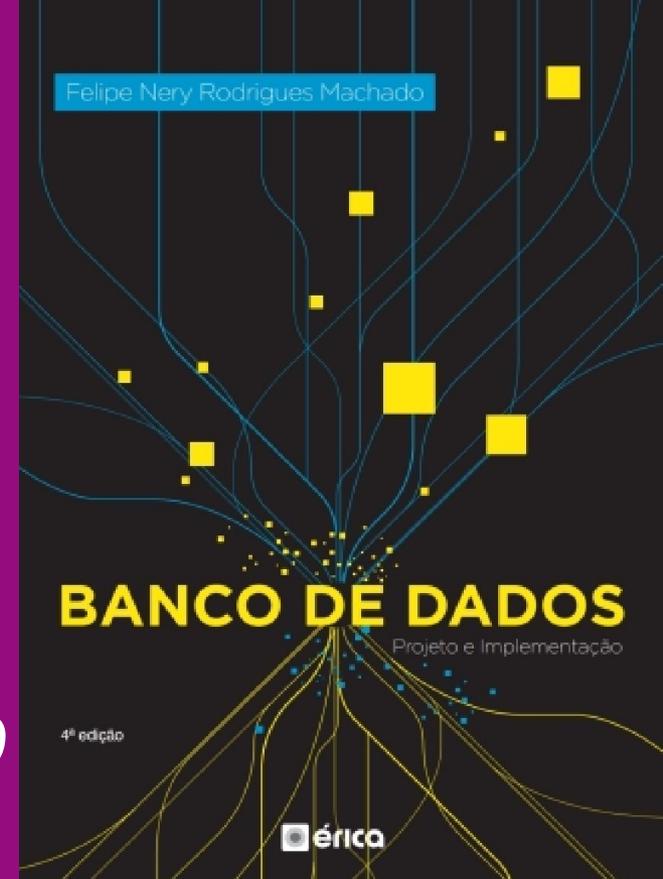
- Apague o registro da Area SC
  - Lembre-se tem que alterar a estrutura da tabela por causa da foreign key

# Presença

- Faça os exercícios e responda o formulário que será passado no chat,
- Prazo: até dia 05/06

# Busca ATIVA

- Leitura e execução dos exemplos livro:
  - Machado, Felipe Nery R. *BANCO DE DADOS – PROJETO E IMPLEMENTAÇÃO*. Disponível em: Minha Biblioteca, (4th edição). Editora Saraiva, 2020.
- Principalmente, antes da A2



# Referências bibliográficas

Machado, Felipe Nery R. BANCO DE DADOS – PROJETO E IMPLEMENTAÇÃO. Disponível em: Minha Biblioteca, (4th edição). Editora Saraiva, 2020.

DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Campus, 2004. 865 p. ISBN 85-352-1273-6.

ELMASRI, Ramez; NAVATHE, Sham. **Sistemas de banco de dados**. 4. ed. São Paulo: Addison-Wesley, 2005. 724 p. ISBN 8588639173.

NAVATHE, Elmasri. **Sistema de Banco de Dados**. 6ª Edição. São Paulo: Person Addison Wesley, 2011.

PEREIRA, Silvio do Lago. **Estruturas de dados fundamentais: conceitos e aplicações**. 8. ed. São Paulo: Érica, 2004. 238 p. ISBN 85-7194-370-2.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de banco de dados**. 3. ed. São Paulo: Makron Books, 2004. 778 p. ISBN 85-346-1073-8.

# CRÉDITOS

## COORDENAÇÃO



Vera Rejane Niedersberg  
Schuhmacher

---

## PROFESSORES



Rafael Lessa  
Daniella Vieira

---