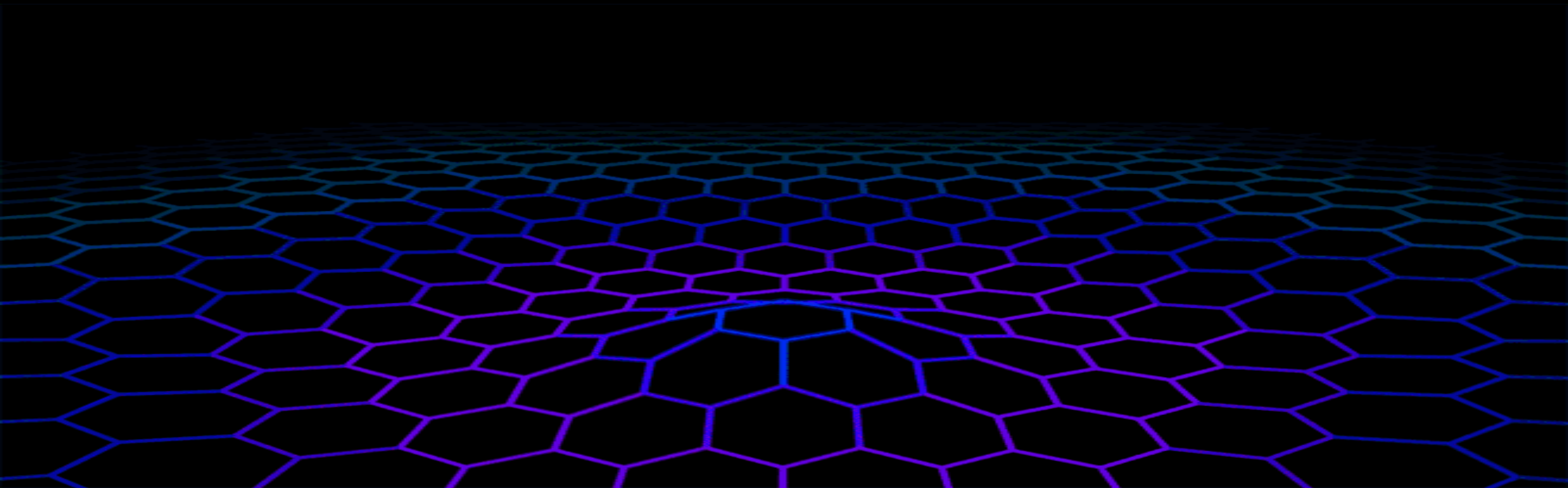


Banco de Dados I



SQL

PostgreSQL



ORACLE®
PL/SQL



Tools

MySQL - <https://dev.mysql.com/downloads>

MariaDB - <https://mariadb.org>

WampServer - <https://www.wampserver.com/en/download-wampserver-64bits>

Xampp - https://www.apachefriends.org/pt_br/index.html

Chocolatey - <https://chocolatey.org>

Atom - <https://atom.io>

Brackets - <http://brackets.io>

DataGrip - <https://www.jetbrains.com/datagrip>

HeidiSQL - <https://www.heidisql.com>

Notepad++ - <https://notepad-plus-plus.org/downloads>

SquirrelL - <http://www.squirrelsql.org>

Sublime Text – <https://www.sublimetext.com/download>

Visual Studio Code - <https://code.visualstudio.com>

DDL – Data Definition Language

Create, Alter e Drop

DML – Data Manipulation Language

Select, Insert, Delete e Update

DCL – Data Control Language (subgrupo da DML)

Grant e Revoke

SQL - Histórico

Surgem vários SGBD baseados em SQL

Problemas - falta de padrões

American National Standard Institute (ANSI)
e

International Standards Organizations (ISO)
definem o padrão SQL.

- SQL-86 (SQL 1)

- SQL-89

- SQL-92 (SQL 2)

- SQL:1999 (SQL 3)

- SQL:2003

- SQL:2008

- SQL:2016



O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Structured Query Language) como interface.

1979 - começou a ser desenvolvido por: David Axmark, Allan Larsson e Michael "Monty" Widenius (empresa **TcX**)

1995 - Primeira versão do MySQL definitivamente lançada.

2008 - A MySQL AB (desenvolvedora do MySQL) foi adquirida pela **Sun Microsystems**.

2009 - A Sun Microsystems (incluindo o MySQL) foi adquirida pela **Oracle** .

Trabalhando com MySQL via console

*Para logar no MySQL, via console, é necessário possuir um usuário com privilégios de **root** para realizar as tarefas de criação, visualização e alteração de bancos de dados.*

*A seguir um exemplo de login utilizando o usuário master do SGBD MySQL, o usuário **root**:*

Acessar a pasta **bin** do MySql

Executar o comando:

```
mysql -u root -p
```

Exemplo: C:\tools\mysql\mysql-8.0.29-winx64\bin>**mysql -u root -p**

Informar a senha: no nosso exemplo a senha é vazia.

Trabalhando com MySQL via console

 Administrador: Windows PowerShell

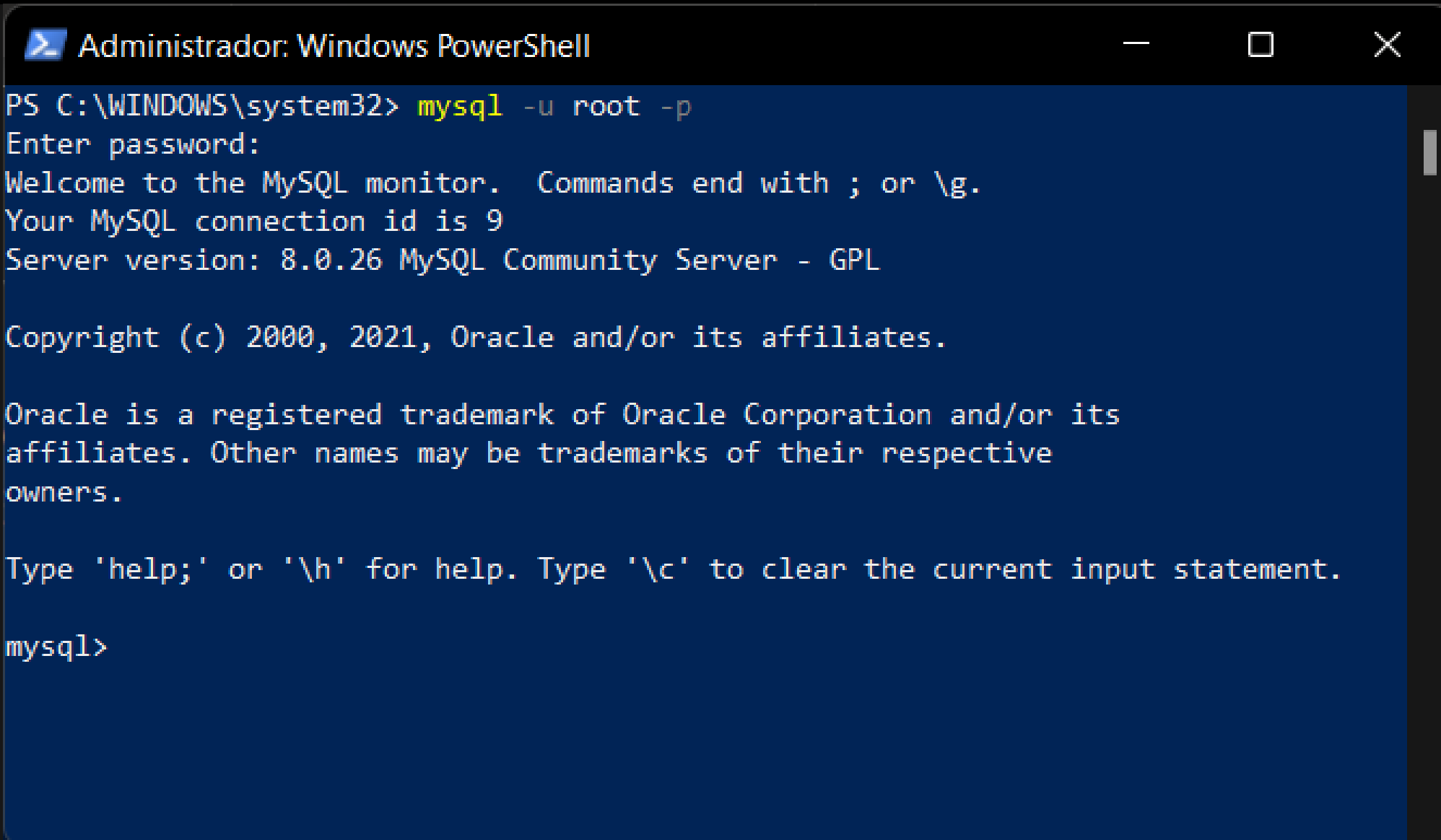
```
PS C:\WINDOWS\system32> mysql -u root -p
```

Trabalhando com MySQL via console

 Administrador: Windows PowerShell

```
PS C:\WINDOWS\system32> mysql -u root -p
Enter password: ■
```

Trabalhando com MySQL via console



```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

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>
```

Trabalhando com MySQL via console

```
Administrador: Windows PowerShell

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.03 sec)

mysql>
```

O MySQL, por exemplo, já vem com alguns bancos de dados criados. Para visualizá-los basta executar o comando **SHOW DATABASES;**

Bancos de Dados

O MySQL, por exemplo, já vem com alguns bancos de dados criados. Para visualizá-los basta executar o comando **SHOW DATABASES;**

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sys            |
+-----+
4 rows in set (0.00 sec)

mysql> _
```

Criação de Banco de Dados

```
Mysql> CREATE DATABASE Exemplo;
```

ou

```
Mysql> CREATE SCHEMA Exemplo;
```

Visualizar Bancos de Dados

```
Mysql> SHOW DATABASES; [Enter]
```

| Databases | |
|--------------------|--|
| information_schema | } criados na instalação do MySQL |
| mysql | |
| performance_schema | |
| sys | |
| Exemplo | |

Databases - Nomenclatura

Máximo 64 caracteres;

Permitido: letras, números, traços, *underlines*;

Proibido: barras e pontos;

Evitar: acentos e cedilhas.

Databases

Não podemos criar bancos de dados com nomes iguais.

Se tentarmos, será exibida a mensagem:

Can't create databases 'Exemplo'. Database exists

Para evitar esse erro usamos a cláusula IF NOT EXISTS

```
MySQL>CREATE DATABASE IF NOT EXISTS Exemplo;
```

Excluindo banco de dados

Ao apagar um banco de dados, todas as tabelas e dados também serão excluídos.

```
Mysql> DROP DATABASE Exemplo;
```

ou

```
Mysql> DROP SCHEMA Exemplo;
```

Com DROP também podemos usar a cláusula IF EXISTS:

```
Mysql> DROP DATABASE IF EXISTS Exemplo;
```

Selecioneando um banco de dados

Podemos ter vários bancos de dados, mas só podemos manipular um por vez.

```
Mysql> USE Exemplo;
```

Tabelas

Para criar Tabelas precisamos definir sua estrutura = seus campos

Exemplo:

Tabela: **aluno**

Campos: **mat, nome, email**

Obs.: Também precisamos definir tipos dos Campos

Tabelas

SQL oferece três instruções para definição do esquema da base de dados:

CREATE TABLE

- define a estrutura da tabela
- cria a tabela vazia

DROP TABLE

- Elimina a tabela da base de dados

ALTER TABLE

- Permite modificar a definição da tabela

Tipos de Dados

No MySQL os tipos de dados são divididos em três grupos:

Tipos Numéricos

Tipos de Data

Tipo de Cadeia

Tipos Numéricos

Exatos:

NUMERIC

DECIMAL

INTEGER

SMALLINT

Aproximados:

FLOAT

REAL ou DOUBLE

DECIMAL ou DEC.

Tipos Numéricos

| Tipo | Bytes | De | Até |
|-----------------------|-------|-----------------------------|---------------------|
| TINYINT | 1 | -128 | 127 |
| SMALLINT | 2 | -32768 | 32767 |
| MEDIUMINT | 3 | -8388608 | 8388607 |
| INT ou Integer | 4 | -2147483648 | 2.147.483.647 |
| BIGINT | 8 | -9223372036854775808 | 9223372036854775807 |
| Bit ou Bool | 1 | Inteiro que pode ser 0 ou 1 | |

Tipos Numéricos

NUMERIC e DECIMAL

- implementados como o mesmo tipo.
- preserva a exatidão (Ex. dados monetários).

Exemplo:

Precisão: número de dígitos
que serão armazenados

salario DECIMAL(5,2)

Escala: número de
dígitos que serão
armazenados após o
ponto decimal

Tipos de Data e Hora

DATE

- para apenas do valor da data, sem a parte da hora.
- formato 'YYYY-MM-DD'.
- faixa de '1001-01-01' até '9999-12-31'.

DATETIME

- para valores que contém data e a hora.
- formato 'YYYY-MM-DD HH:MM:SS'.
- faixa de '1001-01-01 00:00:00' até '9999-12-31 23:59:59'.

Tipos de Data e Hora

TIME

- formato 'HH:MM:SS'
- faixa '-838:59:59' até '838:59:59'.
- Pode ser usado para intervalos de tempo entre dois eventos e não somente para hora do dia (que seria até 24)

Tipos de Data e Hora

TIMESTAMP

- para valores que contém data e a hora.
- Margem vai desde 1 de janeiro de 1970 ao ano 2037

YEAR

- formato com 2 ou 4 algarismos.
- faixa de 1901 até 2155.

Tipos de Cadeia de Caracteres (texto)

CHAR(N)

- Caracteres alfanuméricos
- Tamanho é fixo e instaurado ao ser criado (mesmo que não usado, aquele espaço em disco é alocado)
- Pode ter de 0 a 255 caracteres.

Exemplo: `endereco CHAR(30);`

Observe que não há acentos nem cedilhas no nome do campo, pois muitos servidores não acentuam, e sua tabela teria difícil acesso.

Tipos de Cadeia de Caracteres (texto)

VARCHAR(N)

- Aloca apenas o espaço necessário para gravação
- Trocamos espaço por velocidade, pois este campo é 50% mais lento que o CHAR.
- Exemplo: `endereco VARCHAR(30);`
 - Define um campo chamado `endereco` que pode conter até 30 letras. Se você preencher apenas duas, o campo não ocupará todos os 30 bytes, mas apenas 2.

Tipos de Cadeia de Caracteres (texto)

TEXT/BLOB

- Para guardar grandes quantidades de caracteres.
- Pode conter de 0 a 65535 bytes,
- TEXT **não** é sensível a letras maiúsculas e minúscula quando uma comparação é realizada
- BLOB é sensível a letras maiúsculas e minúscula quando uma comparação é realizada

Tipos de Cadeia de Caracteres (texto)

MediumTEXT / MediumBLOB

- Máximo de 16.777.215 caracteres.

LongTEXT / LongBLOB

- Máximo de 4.294.967.295 caracteres.

SET

SET

Permite que o usuário faça uma escolha dado determinado número de opções. Cada campo pode conter até, 64 opções.

Exemplo:

```
curso SET ("informatica", "geomatica") NOT NULL;
```

Neste exemplo este campo pode conter apenas os seguintes itens:

"informatica"

"geomatica"

"informatica, geomatica"

ENUM

ENUM

Indicado para:

- conjunto de opções fixas
- menu *dropdown* de formulário

Semelhante ao SET, com a diferença que apenas um valor pode ser escolhido.

Exemplo:

```
sexo ENUM("masculino", "feminino") NOT NULL;
```

Neste exemplo este campo pode conter os seguintes valores:

"masculino"

"feminino"

SET e ENUM

Diferença entre o *datatype* **ENUM** e o *datatype* **SET**:

ENUM pode ter um (e apenas 1) valor escolhido de uma lista de opções.
Esta lista pode ter até 65535 elementos.

SET pode ter zero a "n" valores escolhidos da lista de opções.
Esta lista apenas pode ter 64 elementos.

Definição de Base de Dados

SQL oferece três instruções para definição do esquema da base de dados:

CREATE TABLE

- define a estrutura da tabela
- cria a tabela vazia

DROP TABLE

- Elimina a tabela da base de dados

ALTER TABLE

- Permite modificar a definição da tabela

Criação de Tabela



```
CREATE TABLE teste(
```

Nome da
tabela

```
    codigo INT(11),
```

```
    nome CHAR(15),
```

```
    email VARCHAR(25)
```

Tipos dos
campos

```
);
```

Nome dos
campos

Restrições

Uma maneira de limitar os dados que podem ser inseridos em uma tabela é a definição de tipo.

Podemos desejar definir outras restrições como:

- Dados de um coluna em relação a outras colunas ou linhas.
- Valores aceitáveis.

Não Nulo (**NOT NULL**):

- Define que uma coluna não pode conter valor nulo.
- A cláusula **NOT NULL** especifica que uma coluna não admite valor vazio.
- Exemplo: `nome CHAR(15) NOT NULL;`
 - O campo nome não pode ser vazio, é de preenchimento obrigatório.

Unicidade (**UNIQUE**):

- Define que os dados contidos na coluna (ou grupo de colunas) sejam únicos (não se repitam) em relação a todas as outras linhas da tabela.
- A cláusula **UNIQUE** especifica que os dados não se repetem.

Restrições



```
CREATE TABLE alunos (  
    mat    char(5) UNIQUE,  
    nome   char(50),  
    nasc   date  
);
```

-- OU

```
CREATE TABLE alunos (  
    mat    char(5),  
    nome   char(50),  
    nasc   date,  
    UNIQUE (mat)  
);
```

Restrições

Se uma **restrição de unicidade** faz referencia a um grupo de colunas, elas são separadas por vírgula:



```
CREATE TABLE exemplo(  
    a    char(5),  
    b    char(50),  
    c    date,  
    UNIQUE (a, c)  
);
```

Restrições

Chave primária:

- A chave primária indica que a coluna, ou grupo de colunas, pode ser utilizado como identificador único para as linhas da tabela
- É a junção de restrição de unicidade com a restrição de não nulo. Apenas a unicidade não garante identificador único pois não exclui os valores nulos.
- Uma tabela pode ter no máximo uma chave primária, mas pode ter várias restrições de unicidade e de não nulo.

Restrições

```
CREATE TABLE alunos (  
    mat    char(5) UNIQUE NOT NULL,  
    nome   char(50),  
    nasc   date  
);
```

```
CREATE TABLE alunos (  
    mat    char(5) PRIMARY KEY,  
    nome   char(50),  
    nasc   date  
);
```

```
CREATE TABLE exemplo (  
    a      char(5),  
    b      char(50),  
    c      date,  
    PRIMARY KEY (a, c)  
);
```

Auto incremento:

- Este recurso, faz com que conforme novos registros são criados, automaticamente estes obtém valores que correspondem ao valor deste mesmo campo no registro anterior, somado a 1.
- Exemplo: `codigo INT AUTO_INCREMENT;`
 - Soma um a cada registro automaticamente neste campo. Começando de 1, com inserção subsequente.



```
DROP DATABASE IF EXISTS aula09;  
CREATE DATABASE IF NOT EXISTS aula09;  
USE aula09;
```

```
CREATE TABLE teste(  
    codigo    INTEGER AUTO_INCREMENT NOT NULL,  
    nome      VARCHAR(195) NOT NULL,  
    email     VARCHAR(200),  
    telefone  CHAR(15),  
    PRIMARY KEY(codigo)  
);
```

Comandos relativos as tabelas:

Mostrar tabelas - Lista todas as tabelas existentes no banco de dados atual.

```
mysql>SHOW tables;
```

Exibir colunas - Exibe as colunas da tabela.

```
mysql>SHOW COLUMNS FROM teste;
```

Mostrar estrutura - Mostra a estrutura da tabela.

```
mysql>DESCRIBE teste;
```

```
mysql>DESC teste;
```

Vamos praticar
novamente?

```
USE aula09;
```

```
CREATE TABLE departamento(  
    codigo INT(11) NOT NULL,  
    nome VARCHAR(15) NOT NULL,  
    PRIMARY KEY(codigo),  
    UNIQUE (nome)  
);
```

```
CREATE TABLE empregado(  
    matricula CHAR(9) NOT NULL,  
    nome VARCHAR(15) NOT NULL,  
    dataNasc DATE,  
    endereco VARCHAR(30),  
    sexo CHAR(1),  
    salario DECIMAL(10,2),  
    codDep INT(11) NOT NULL,  
    PRIMARY KEY (matricula),  
    FOREIGN KEY (codDep) REFERENCES departamento(codigo)  
);
```


ATIVIDADE - Crie a modelagem lógica e física:

Criar um banco chamado aula09b

Criar as tabelas:

EMPREGADO (matricula, nome, endereco, salario, matSuperv, codDep)

DEPARTAMENTO (codigo, nome, telefone)

PROJETO (codigo, nome, local, codDep)

ALOCACAO (matEmp, codProj, horas)

DEPENDENTE (id, matEmp, nome, dataN)

Solução



```
DROP DATABASE IF EXISTS aula09b;  
CREATE DATABASE IF NOT EXISTS aula09b;  
USE aula09b;
```

```
CREATE TABLE departamento(  
    codigo    INT(11),  
    nome      VARCHAR(45) NOT NULL,  
    telefone  VARCHAR(15),  
    PRIMARY KEY(codigo)  
);
```

```
CREATE TABLE empregado(  
    matricula    INT(11),  
    nome          VARCHAR(45) NOT NULL,  
    endereco      VARCHAR(100),  
    salario       DECIMAL(10,2),  
    matSupervisor INT(11) NOT NULL,  
    codDep        INT(11) NOT NULL,  
    PRIMARY KEY (matricula),  
    FOREIGN KEY (codDep) REFERENCES departamento(codigo),  
    FOREIGN KEY (matSupervisor) REFERENCES  
empregado(matricula)  
);
```

```
CREATE TABLE projeto(  
    codigo    INT(11),  
    nome      VARCHAR(45) NOT NULL,  
    local     VARCHAR(100),  
    codDep    INT(11) NOT NULL,  
    PRIMARY KEY (codigo),  
    FOREIGN KEY (codDep) REFERENCES departamento(codigo)  
);
```

```
CREATE TABLE alocao(  
    matEmp    INT(11),  
    codProj   INT(11),  
    horas     INT(11),  
    PRIMARY KEY (matEmp, codProj),  
    FOREIGN KEY (matEmp) REFERENCES empregado(matricula),  
    FOREIGN KEY (codProj) REFERENCES projeto(codigo)  
);
```

```
CREATE TABLE dependente(  
    id    INT(11),  
    matEmp INT(11),  
    nome  VARCHAR(45) NOT NULL,  
    dataN DATE,  
    PRIMARY KEY (id),  
    FOREIGN KEY (matEmp) REFERENCES empregado(matricula)  
);
```

Mais um...

Preparando o terreno

Crie a tabela **clientes**

```
CREATE TABLE clientes(  
    codigo CHAR(6),  
    nome VARCHAR(20),  
    dataN DATE,  
    valor DECIMAL(10,2),  
    cidade VARCHAR(20),  
    PRIMARY KEY(codigo)  
);
```

SQL - DML

Inserir dados

INSERT INTO e VALUES.

Obs.: *strings* devem ser incluídas em pares de apóstrofos (aspas simples).

Números Inteiros ou Flutuantes não necessitam de apóstrofos.

Sintaxe:

```
INSERT INTO <nome_tabela> (<campo1>, ..., <campoN>)  
VALUES (<valorCampo1>, ..., < valorCampoN>);
```

Sintaxe alternativa:

```
INSERT INTO <nome_tabela>  
VALUES (<valorCampo1>, ..., < valorCampoN>);
```

Exemplo

Inserindo registros na tabela **clientes**

```
INSERT INTO clientes (codigo, nome, dataN, valor, cidade) VALUES ('B5200X', 'Bartolomeu', '1982-12-23', 100.4, 'Pelotas');
INSERT INTO clientes (codigo, nome, dataN, valor, cidade) VALUES ('A73111', 'Orpiliano', '2001-07-19', 98.6, 'Canoas');
INSERT INTO clientes (codigo, nome, dataN, valor, cidade) VALUES ('A77222', 'Dorvalina', '1999-11-21', 1219.48, 'Pelotas');
INSERT INTO clientes (codigo, nome, dataN, valor, cidade) VALUES ('B79321', 'Luz Artemina', '1969-01-24', 31.29, 'Erechim');
INSERT INTO clientes VALUES ('X9843Z', 'Josisvaldo', '1967-04-01', 93.45, 'Blumenau');
INSERT INTO clientes (codigo, nome) VALUES ('S9983W', 'Sarafina');
```


Recuperar dados

SELECT e FROM.

Sintaxe:

```
SELECT * FROM nome_tabela;
```

Sintaxe alternativa:

```
SELECT <coluna1>, ..., <colunaN> FROM nome_tabela;
```

Exemplo

Recuperar todas as colunas da tabela **clientes**.

Recuperar apenas as colunas **nome** e **valor** da tabela **clientes**.

Exemplo

Recuperar todas as colunas da tabela **clientes**.

```
SELECT * FROM clientes
```

Recuperar apenas as colunas **nome** e **valor** da tabela **clientes**.

Exemplo

Recuperar todas as colunas da tabela **clientes**.

```
SELECT * FROM clientes
```

Recuperar apenas as colunas **nome** e **valor** da tabela **clientes**.

```
SELECT nome, valor FROM clientes
```

Recuperar dados específicos

WHERE, OR e AND.

Sintaxe

```
SELECT * FROM nome_tabela WHERE <condição>;
```

Sintaxe utilizando **AND**

Todas as condições envolvidas devem ser verdadeiras.

```
SELECT * FROM nome_tabela WHERE <condição> AND <condição>;
```

Sintaxe utilizando **OR**

Pelo menos uma condição envolvida deve ser verdadeira.

```
SELECT * FROM nome_tabela WHERE <condição> OR <condição>;
```

Operadores de comparação para WHERE

| Nome | Operador | Exemplo | Descrição |
|----------------|----------|----------------------|--|
| Igualdade | = | valor_coluna = 15 | Verificar se os dois valores são iguais. |
| Maior que | > | valor_coluna > 25 | Verificar se o valor da esquerda é maior que o da direita. |
| Menor que | < | valor_coluna < 36 | Verificar se o valor da esquerda é menor que o da direita. |
| Maior ou igual | >= | valor_coluna >= 28 | Verificar se o valor da esquerda é maior ou igual ao da direita. |
| Menor ou igual | <= | valor_coluna <= 89 | Verificar se o valor da esquerda é menor ou igual ao da direita. |
| Desigualdade | != ou <> | valor_coluna != 2020 | Verificar se os dois valores são diferentes. |

Exemplo

Recuperar apenas os registros de clientes da cidade **Pelotas**

Recuperar apenas os registros de clientes da cidade **Pelotas** com valor menor do que 100

Recuperar apenas os registros de clientes da cidade **Pelotas** ou da cidade **Erechim**

Recuperar apenas clientes com valor **acima** de **100**.

Recuperar apenas clientes que **não** são de **Pelotas**

Exemplo

Recuperar apenas os registros de clientes da cidade **Pelotas**

```
SELECT * FROM clientes WHERE cidade = 'Pelotas';
```

Recuperar apenas os registros de clientes da cidade **Pelotas** com valor menor do que 100

Recuperar apenas os registros de clientes da cidade **Pelotas** ou da cidade **Erechim**

Recuperar apenas clientes com valor **acima** de **100**.

Recuperar apenas clientes que **não** são de **Pelotas**

Exemplo

Recuperar apenas os registros de clientes da cidade **Pelotas**

```
SELECT * FROM clientes WHERE cidade = 'Pelotas';
```

Recuperar apenas os registros de clientes da cidade **Pelotas** com valor menor do que 100

```
SELECT * FROM clientes WHERE cidade = 'Pelotas' AND valor < 100;
```

Recuperar apenas os registros de clientes da cidade **Pelotas** ou da cidade **Erechim**

Recuperar apenas clientes com valor **acima** de **100**.

Recuperar apenas clientes que **não** são de **Pelotas**

Exemplo

Recuperar apenas os registros de clientes da cidade **Pelotas**

```
SELECT * FROM clientes WHERE cidade = 'Pelotas';
```

Recuperar apenas os registros de clientes da cidade **Pelotas** com valor menor do que 100

```
SELECT * FROM clientes WHERE cidade = 'Pelotas' AND valor < 100;
```

Recuperar apenas os registros de clientes da cidade **Pelotas** ou da cidade **Erechim**

```
SELECT * FROM clientes WHERE cidade = 'Pelotas' OR cidade = 'Erechim';
```

Recuperar apenas clientes com valor **acima** de **100**.

Recuperar apenas clientes que **não** são de **Pelotas**

Exemplo

Recuperar apenas os registros de clientes da cidade **Pelotas**

```
SELECT * FROM clientes WHERE cidade = 'Pelotas';
```

Recuperar apenas os registros de clientes da cidade **Pelotas** com valor menor do que 100

```
SELECT * FROM clientes WHERE cidade = 'Pelotas' AND valor < 100;
```

Recuperar apenas os registros de clientes da cidade **Pelotas** ou da cidade **Erechim**

```
SELECT * FROM clientes WHERE cidade = 'Pelotas' OR cidade = 'Erechim';
```

Recuperar apenas clientes com valor **acima** de **100**.

```
SELECT * FROM clientes WHERE valor > 100;
```

Recuperar apenas clientes que **não** são de **Pelotas**

Exemplo

Recuperar apenas os registros de clientes da cidade **Pelotas**

```
SELECT * FROM clientes WHERE cidade = 'Pelotas';
```

Recuperar apenas os registros de clientes da cidade **Pelotas** com valor menor do que 100

```
SELECT * FROM clientes WHERE cidade = 'Pelotas' AND valor < 100;
```

Recuperar apenas os registros de clientes da cidade **Pelotas** ou da cidade **Erechim**

```
SELECT * FROM clientes WHERE cidade = 'Pelotas' OR cidade = 'Erechim';
```

Recuperar apenas clientes com valor **acima** de 100.

```
SELECT * FROM clientes WHERE valor > 100;
```

Recuperar apenas clientes que **não** são de **Pelotas**

```
SELECT * FROM clientes WHERE cidade != 'Pelotas';
```

Alterar dados

UPDATE e SET.

Sintaxe:

```
UPDATE <Nome da Tabela>  
    SET <Coluna1> = 'Valor Coluna 1', <Coluna2> = 'Valor Coluna 2'  
WHERE <Condição>;
```

Sintaxe alternativa:

```
UPDATE <Nome da Tabela>  
    SET <Coluna1> = 'Valor Coluna 1', <Coluna2> = 'Valor Coluna 2';
```

Exemplo

Alterar o nome do cliente **Bartolomeu** para **Bartoloteu**

Alterar a cidade do cliente de código 'A77222' para Satolep

Alterar para 90 o valor dos clientes da cidade 'Pelotas' que estejam com valor acima de 1000.

Alterar a dataN de todos os clientes para 1º de abril de 2000

Exemplo

Alterar o nome do cliente **Bartolomeu** para **Bartoloteu**

```
UPDATE clientes  
  SET nome = 'Bartoloteu'  
 WHERE nome = 'Bartolomeu';
```

Alterar a cidade do cliente de código 'A77222' para Satolep

Alterar para 90 o valor dos clientes da cidade 'Pelotas' que estejam com valor acima de 1000.

Alterar a dataN de todos os clientes para 1º de abril de 2000

Exemplo

Alterar o nome do cliente **Bartolomeu** para **Bartoloteu**

```
UPDATE clientes  
  SET nome = 'Bartoloteu'  
 WHERE nome = 'Bartolomeu';
```

Alterar a cidade do cliente de código 'A77222' para Satolep

```
UPDATE clientes  
  SET cidade = 'Satolep'  
 WHERE codigo = 'A77222';
```

Alterar para 90 o valor dos clientes da cidade 'Pelotas' que estejam com valor acima de 1000.

Alterar a dataN de todos os clientes para 1º de abril de 2000

Exemplo

Alterar o nome do cliente **Bartolomeu** para **Bartoloteu**

```
UPDATE clientes  
  SET nome = 'Bartoloteu'  
 WHERE nome = 'Bartolomeu';
```

Alterar a cidade do cliente de código 'A77222' para Satolep

```
UPDATE clientes  
  SET cidade = 'Satolep'  
 WHERE codigo = 'A77222';
```

Alterar para 90 o valor dos clientes da cidade 'Pelotas' que estejam com valor acima de 1000.

```
UPDATE clientes  
  SET valor = 90  
 WHERE cidade = 'Pelotas' AND valor > 1000;
```

Alterar a dataN de todos os clientes para 1º de abril de 2000

Exemplo

Alterar o nome do cliente **Bartolomeu** para **Bartoloteu**

```
UPDATE clientes  
  SET nome = 'Bartoloteu'  
 WHERE nome = 'Bartolomeu';
```

Alterar a cidade do cliente de código 'A77222' para Satolep

```
UPDATE clientes  
  SET cidade = 'Satolep'  
 WHERE codigo = 'A77222';
```

Alterar para 90 o valor dos clientes da cidade 'Pelotas' que estejam com valor acima de 1000.

```
UPDATE clientes  
  SET valor = 90  
 WHERE cidade = 'Pelotas' AND valor > 1000;
```

Alterar a dataN de todos os clientes para 1º de abril de 2000

```
UPDATE clientes  
  SET dataN = '2000-04-01';
```

Lembrete

Campos texto (CHAR, VARCHAR, BLOB, TEXT, etc.) têm de ficar entre apóstrofos (aspas simples).

Em campos do tipo número, não se usam apóstrofos.

Revisando

As pesquisas no MySQL são feitas através do comando **SELECT**.

Exemplo:

```
SELECT * FROM clientes;
```

Resultado:

Lista todos os campos(*) de todos os registros da tabela **clientes**.

Ainda revisando

Para visualizar somente alguns campos da tabela, devemos especificar os nomes das colunas desejadas, separados por virgulas.

Exemplo:

```
SELECT nome, valor FROM clientes;
```

Resultado:

Lista os campos **nome** e **valor** de todos registros da tabela **clientes**.

ORDENAÇÃO

ORDER BY

ORDER BY, ou **ORDENAR POR**, simplesmente lista os registros, colocando-os em ordem de acordo com o campo solicitado.

```
SELECT * FROM clientes WHERE valor > 100 ORDER BY nome ASC;
```

ORDENAÇÃO

ORDER BY – ASC – DESC

ASC e **DESC** especificam o tipo de classificação e são, respectivamente, abreviações das palavras em inglês **ascending** e **descending**, ou seja, classificação em ordem crescente ou decrescente.

Quando não especificamos nenhum, o padrão é ascendente

```
SELECT * FROM clientes ORDER BY nome;  
SELECT * FROM clientes ORDER BY nome ASC;  
SELECT * FROM clientes ORDER BY nome DESC;  
SELECT matricula, nome, valor FROM clientes ORDER BY valor DESC;  
SELECT nome FROM clientes ORDER BY valor DESC;
```

Excluir registro

DELETE

Sintaxe:

```
DELETE FROM <Nome da Tabela> WHERE <Condição>;
```

Sintaxe alternativa:

```
DELETE FROM <Nome da Tabela>;
```


Exemplo

Excluir apenas o cliente que tenha o código 'A73111'.

Excluir todos os registros de clientes.

Exemplo

Excluir apenas o cliente que tenha o código 'A73111'.

```
DELETE FROM cliente  
WHERE codigo = 'A73111';
```

Excluir todos os registros de clientes.

Exemplo

Excluir apenas o cliente que tenha o código 'A73111'.

```
DELETE FROM cliente  
WHERE codigo = 'A73111';
```

Excluir todos os registros de clientes.

```
DELETE FROM cliente;
```

ATIVIDADE - Crie a modelagem lógica e física:

Crie as tabelas:

EMPREGADO (matricula, nome, endereco, salario, matSuperv, codDep)

DEPARTAMENTO (codigo, nome, telefone)

PROJETO (codigo, nome, local, codDep)

ALOCACAO (matEmp, codProj, horas)

DEPENDENTE (id, matEmp, nome, dataN)

Crie um script de inserção de dados para as tabelas (seja criativo. Use dados fictícios)

POPULANDO

Crie um script de inserção de dados para as tabelas (seja criativo. Use dados fictícios)

- Cadastre empregados com salario menor do que 5.000, igual a 5.000 e maior do que 5.000;
- Cadastre pelo menos três funcionários com salário entre 1.700 e 2.800
- Cadastre pelo menos 10 departamentos (com códigos de 1 até 10)
- Cadastre dois funcionários no departamento 1;
- Cadastre pelo menos três funcionários no departamento 2
- Cadastre 3 funcionários no departamento 5;
- Cadastre alguns dependentes com data de nascimento igual a 27/10/2002;
- Cadastre alguns dependentes com data de nascimento posterior a 27/10/2002;

Exercícios - Escreva a instrução SQL para:

1. Apresentar a listagem completa dos registros da tabela empregado;
2. Apresentar uma listagem dos nomes e salários dos empregados com salario maior do que 5.000;
3. Listar os nomes e os salários dos empregados em ordem alfabética, decrescente, de nome;
4. Listar os nomes dos empregados do departamento 4 ordenados pelo endereço;
5. Alterar para 2500,50 o salário dos empregados do departamento 2;
6. Aumentar em 20% o salário de todos os empregados;
7. Excluir todos os empregados do departamento 1;
8. Apresente a listagem dos dependentes que nasceram em 27/10/2002;
9. Apresente a listagem dos dependentes que nasceram após 27/10/2002;
10. Listar os empregados do departamento 5;
11. Listar empregados com salário até 5000,00;
12. Listar empregados com salário entre 1700,00 e 2800,00