

Banco de Dados II

Material de Aula

MySql

Prof. Enildo

E-mail: professor.enildo@hotmail.com

Sumário

Introdução à Linguagem SQL e ao Ambiente MySQL	3
O que é SQL?	3
Subconjuntos da linguagem SQL	3
O que é um Banco de Dados Relacional?	4
SQL é igual em todos os bancos de dados?	4
Conceito de SGBD	4
Ambientes para Executar Comandos MySQL	4
Referências para estudos	5
Implementação de Banco de Dados no MySQL	6
Criação de Banco de Dados no MySQL	7
Create Database	7
show databases	7
DROP DATABASE	7
DROP DATABASE IF EXISTS	8
Tipos de Dados Padrão ANSI	9
Tipos de Dados MySql	10
Sub-Linguagem (DDL) – Data Definition Language	12
CREATE	12
Create Table	12
Show Tables	13
Describe	13
ALTER	14
SQL Constraints (Restrições)	17
NOT NULL	17
UNIQUE	17
DEFAULT	18
CHECK	18
PRIMARY KEY	18

FOREIGN KEY	19
AUTO_INCREMENT	20
DROP TABLE	15
TRUNCATE	15
RENAME	16
INDEX.....	16
FUNÇÕES ÚTEIS NO MYSQL	21

Introdução à Linguagem SQL e ao Ambiente MySQL

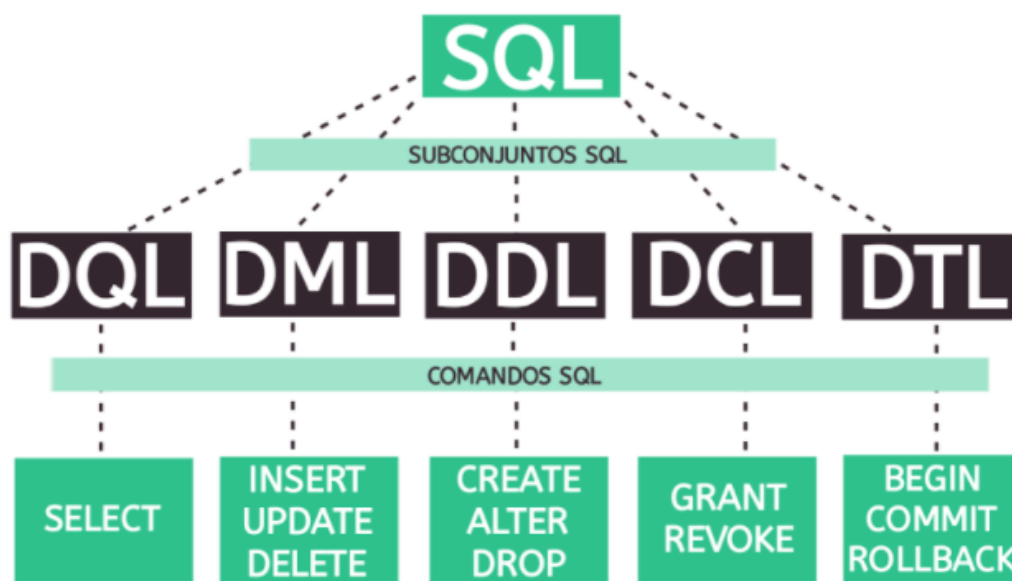
Este material foi elaborado com o objetivo de auxiliar nas aulas de Banco de Dados, oferecendo uma abordagem prática e acessível para quem está iniciando seus estudos na área. Ao longo das próximas seções, serão apresentados comandos da linguagem SQL padrão ANSI, utilizando como base o Sistema Gerenciador de Banco de Dados (SGBD) MySQL, operado preferencialmente pela interface gráfica MySQL Workbench.

⚠ Atenção: alguns comandos demonstrados nesta apostila podem ser específicos do MySQL. Caso esteja utilizando outro SGBD (como Oracle, SQL Server ou PostgreSQL), consulte a documentação oficial para verificar a sintaxe equivalente. A estrutura lógica será a mesma, mas pode haver variações de comandos e funções específicas.

O que é SQL?

A SQL (Structured Query Language), em português Linguagem de Consulta Estruturada, é a linguagem padrão utilizada para interagir com bancos de dados relacionais. A SQL permite criar estruturas, armazenar informações, consultar dados, modificar registros e controlar o acesso ao banco.

Subconjuntos da linguagem SQL



É uma linguagem essencial para profissionais da área de tecnologia, como desenvolvedores, técnicos, analistas e administradores de banco de dados.

A SQL é padronizada pelo ANSI (American National Standards Institute), garantindo que sua base sintática funcione em diversos SGBDs. No entanto, algumas empresas, como Oracle e Microsoft, criaram extensões próprias, como:

- PL/SQL – Oracle
- Transact-SQL (T-SQL) – Microsoft SQL Server

O que é um Banco de Dados Relacional?

Um banco de dados relacional é um sistema que organiza as informações em tabelas que se relacionam entre si. Essas relações permitem maior consistência e integridade dos dados.

Um conceito fundamental nesse modelo é a integridade referencial, que garante que os dados em diferentes tabelas estejam sempre sincronizados. Por exemplo, uma chave primária (PK) de uma tabela pode ser usada como chave estrangeira (FK) em outra, estabelecendo vínculos lógicos entre registros.

SQL é igual em todos os bancos de dados?

Sim e não. A estrutura básica da SQL é a mesma para todos os SGBDs compatíveis com o padrão ANSI. Comandos como SELECT, INSERT, UPDATE, DELETE, CREATE TABLE e DROP funcionam de forma muito semelhante.

Porém, cada fornecedor pode adicionar extensões proprietárias à linguagem. Por isso, ao migrar ou trabalhar com diferentes sistemas (MySQL, SQL Server, Oracle, PostgreSQL), é importante entender que a lógica se mantém, mas a sintaxe pode variar em alguns pontos.

Conceito de SGBD

O SGBD (Sistema Gerenciador de Banco de Dados) é o software responsável por gerenciar o armazenamento, a segurança, a recuperação e a integridade dos dados. Ele também garante o acesso simultâneo por múltiplos usuários.

Neste material, utilizaremos o MySQL, um dos SGBDs mais populares e gratuitos do mercado, amplamente utilizado em:

- Sites e blogs
- Sistemas web corporativos
- Aplicativos de e-commerce
- Ambientes acadêmicos e profissionais

Ambientes para Executar Comandos MySQL

Você pode praticar os comandos SQL do MySQL utilizando diferentes ambientes e ferramentas, conforme sua preferência:

Ambientes gráficos:

- MySQL Workbench – Interface oficial e recomendada para diagramas, scripts e execução de queries.
- DBeaver – Ferramenta multiplataforma que suporta diversos SGBDs.
- phpMyAdmin – Interface web, especialmente útil em servidores locais com XAMPP.

Linha de comando:

- Terminal/Console – Para quem prefere executar comandos diretamente com maior agilidade e controle.

Referências para estudos

1. SQL e seus subconjuntos (ANSI SQL, PL/SQL, T-SQL)

- Elmasri, R., & Navathe, S. B. (2011). *Sistemas de Banco de Dados* (6ª ed.). São Paulo: Pearson.
- Referência clássica sobre SQL padrão, subconjuntos da linguagem, SGBDs relacionais e integridade referencial.
- Date, C. J. (2004). *Introdução a Sistemas de Bancos de Dados* (8ª ed.). Rio de Janeiro: Campus.
- Um dos principais autores em teoria de bancos de dados relacionais e SQL ANSI.
- Oracle Corporation. (2024). *PL/SQL Language Reference*. Recuperado de: <https://docs.oracle.com/en/database/oracle/oracle-database/>
- Microsoft. (2024). *Transact-SQL Reference (T-SQL)*. Recuperado de: <https://learn.microsoft.com/en-us/sql/t-sql/>

2. Conceito de Banco de Dados Relacional e Integridade Referencial

- Korth, H. F., & Silberschatz, A. (2010). *Sistemas de Banco de Dados* (6ª ed.). São Paulo: Pearson.
- Explica com profundidade o modelo relacional, chaves, normalização, integridade e tipos de relacionamentos.
- Garcia-Molina, H., Ullman, J. D., & Widom, J. (2009). *Database Systems: The Complete Book* (2nd ed.). Pearson.

3. SGBD e ambiente MySQL

- Oracle Corporation. (2024). *MySQL 8.0 Reference Manual*. Recuperado de: <https://dev.mysql.com/doc/refman/8.0/en/>
- Documentação oficial do MySQL, abrangendo conceitos de SGBD, comandos SQL, Workbench, segurança, etc.
- Welling, L., & Thomson, L. (2009). *PHP and MySQL Web Development* (4ª ed.). Addison-Wesley.
- Excelente para compreender a aplicação prática do MySQL no desenvolvimento web e uso de interfaces como o phpMyAdmin.

4. Ferramentas gráficas (MySQL Workbench, DBeaver, phpMyAdmin)

- Oracle Corporation. (2024). *MySQL Workbench Manual*. Recuperado de: <https://dev.mysql.com/doc/workbench/en/>
- DBeaver Community. (2024). *User Guide*. Recuperado de: <https://dbeaver.io/docs/>
- phpMyAdmin Team. (2024). *phpMyAdmin Documentation*. Recuperado de: <https://docs.phpmyadmin.net/>

5. Materiais complementares online (úteis para alunos)

- W3Schools. (2024). *SQL Tutorial*. Recuperado de: <https://www.w3schools.com/sql/>
- Khan Academy. (2023). *Intro to SQL: Querying and Managing Data*. Recuperado de: <https://www.khanacademy.org/computing/computer-programming/sql>

Implementação de Banco de Dados no MySQL

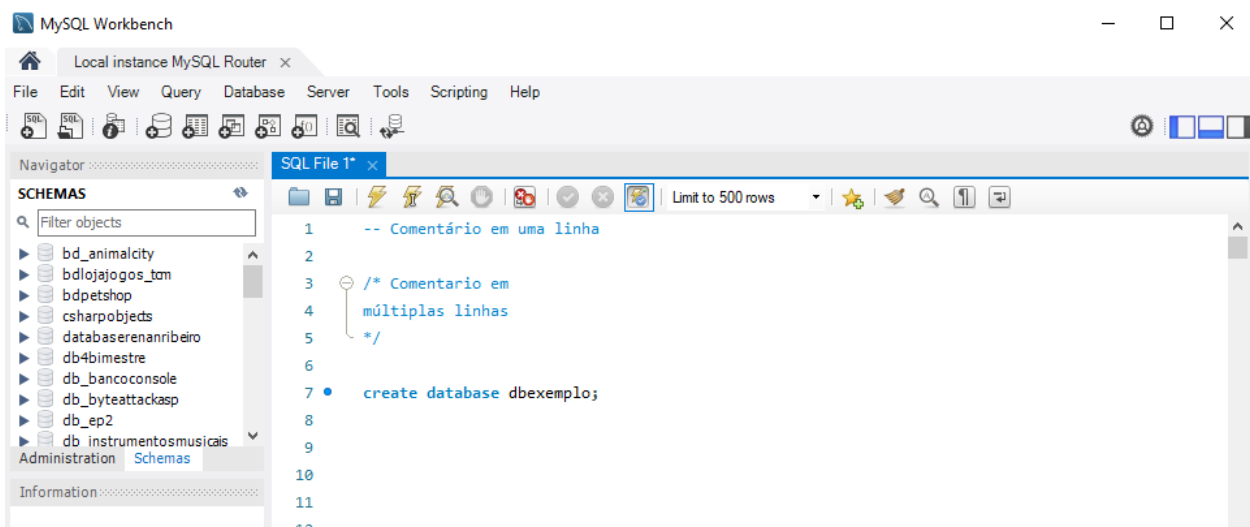
Comentários em Arquivos SQL

Antes de iniciarmos a execução de comandos SQL, é importante aprender como adicionar comentários em seus scripts .sql. Comentários são linhas ignoradas durante a execução, usadas para documentar trechos de código ou desativar comandos temporariamente.

Tipos de comentário:

-- Comentário em uma linha

/* Comentário em
múltiplas linhas
*/



Comentários ajudam na organização, revisão e explicação de trechos de código.

Símbolos em Sintaxes SQL

Para entender melhor a estrutura dos comandos SQL, é comum usar símbolos na documentação da sintaxe:

- < > substituir por nome
- [] opcional
- | separador de opções

Criação de Banco de Dados no MySQL

Create Database

Sintaxe para criar uma base de dados:

`CREATE DATABASE <Nome da Base> [;]`

- **CREATE**: é um comando SQL para criar objetos.
- **DATABASE**: é o objeto do tipo base de dados.
- **<Nome da Base>**: é o nome que damos a base de dados.
- **[;]**: ponto e vírgula indica o término do comando.

Exemplo: `CREATE DATABASE Exemplo_db;`

Comando com personalização (charset e collation)

É recomendável definir o **conjunto de caracteres (charset)** e a **ordenação textual (collation)** ao criar o banco, principalmente para sistemas multilíngues ou que usam acentuação.

```
CREATE DATABASE escola
CHARACTER SET utf8mb4
COLLATE utf8mb4_general_ci;
```

- **CHARACTER SET** define o conjunto de caracteres (ex: UTF-8);
- **COLLATE** define como o banco ordena e compara os dados textual.
- **utf8mb4** permite o uso completo do alfabeto Unicode, incluindo emojis.
- **utf8mb4_general_ci** define uma ordenação que ignora maiúsculas, minúsculas e acentos.

show databases

(Verificando os bancos existentes)

`show databases;`

Exibe todos os bancos de dados existentes no servidor MySQL.

⚠ Este comando **não é parte do padrão SQL ANSI**, mas é amplamente usado no MySQL.

DROP DATABASE

Comando básico para excluir uma base de dados

Sintaxe para excluir uma base de dados:

`DROP DATABASE <Nome da Base> [;]`

Exemplo:

DROP DATABASE escola_db;

⚠ Este comando **apaga permanentemente** o banco de dados e todos os seus dados.

DROP DATABASE IF EXISTS

Exclusão segura com verificação

DROP DATABASE IF EXISTS escola_db;

Evita erros caso o banco escola **não exista** no momento da execução.

Cuidados importantes

- ✓ Nunca exclua um banco de produção sem **backup**.
- ✓ Certifique-se de que o banco **não está em uso** no momento da exclusão.
- ✓ Em interfaces gráficas como o **MySQL Workbench**, é possível excluir um banco clicando com o botão direito no nome da base.

Dica: Certifique-se de ter privilégios de administrador antes de criar qualquer banco de dados. Uma vez que um banco de dados é criado, você pode verificá-lo na lista de bancos de dados com o seguinte comando SQL: SHOW DATABASES;

Referências Técnicas

- ✓ Oracle. (2024). *MySQL 8.0 Reference Manual*. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/>
- ✓ W3Schools. (2024). *SQL CREATE DATABASE Statement*. https://www.w3schools.com/sql/sql_create_db.asp
- ✓ TutorialsPoint. (2024). *SQL - Create Database*. <https://www.tutorialspoint.com/sql/sql-create-database.htm>
- ✓ Boson Treinamentos. (2022). *MySQL – Comandos SHOW, DESCRIBE e MYSQLSHOW*. <http://www.bosontreinamentos.com.br/mysql/mysql-comandos-show-describe-e-mysqlshow-39/>
- ✓ Korth, H. F., & Silberschatz, A. (2010). *Sistemas de Banco de Dados*. Pearson.
- ✓ https://www.w3schools.com/sql/sql_create_db.asp
- ✓ <https://www.tutorialspoint.com/sql/sql-create-database.htm>
- ✓ <http://www.bosontreinamentos.com.br/mysql/mysql-comandos-show-describe-e-mysqlshow-39/>

Tipos de Dados Padrão ANSI

O padrão **ANSI SQL** (também conhecido como **SQL padrão** ou **SQL ANSI/ISO**) define um **conjunto básico e padronizado de tipos de dados** que os bancos relacionais **devem** ou **podem** suportar. Ele é mais **genérico e portátil** do que os tipos específicos de cada SGBD (como MySQL, SQL Server, PostgreSQL etc.).

Tipos de dados definidos pelo padrão ANSI SQL

NUMÉRICOS

Tipo ANSI	Descrição
INTEGER	Número inteiro
SMALLINT	Inteiro pequeno
BIGINT	Inteiro grande (padrão recente)
DECIMAL(p,s)	Número exato com precisão e escala
NUMERIC(p,s)	Idêntico ao DECIMAL (mesma função)
FLOAT(p)	Número de ponto flutuante de precisão aproximada
REAL	Ponto flutuante de precisão simples
DOUBLE PRECISION	Ponto flutuante de precisão dupla

CARACTER (TEXTO)

Tipo ANSI	Descrição
CHAR(n)	Cadeia de caracteres de comprimento fixo
VARCHAR(n)	Cadeia de caracteres de comprimento variável
CHARACTER(n)	Sinônimo de CHAR(n)
CHARACTER VARYING(n)	Sinônimo de VARCHAR(n)

BINÁRIOS (menos usados, mas padronizados)

Tipo ANSI	Descrição
BINARY(n)	Dados binários de comprimento fixo
VARBINARY(n)	Dados binários de comprimento variável

DATA E HORA

Tipo ANSI	Descrição
DATE	Data (ano, mês, dia)
TIME	Hora (hora, minuto, segundo)
TIMESTAMP	Combinação de data e hora
INTERVAL	Intervalos de tempo (ex: INTERVAL DAY TO SECOND)

BOOLEANO

Tipo ANSI	Descrição
BOOLEAN	Lógico: TRUE, FALSE, UNKNOWN

Outros (avançados no padrão SQL mais moderno)

Tipo ANSI	Descrição
ARRAY	Vetores (não suportado por todos os SGBDs)
MULTISET	Conjunto de valores (tipo set)
ROW	Estruturas tipo registro ou objeto
XML	Dados em formato XML
JSON	Suporte parcial em SQL:2016 em diante

RESUMO

O padrão ANSI SQL define **tipos de dados genéricos e portáveis**, como:

- **Numéricos:** INTEGER, DECIMAL, FLOAT
- **Texto:** CHAR, VARCHAR
- **Data/Hora:** DATE, TIME, TIMESTAMP
- **Booleano:** BOOLEAN
- **Binário:** BINARY, VARBINARY

Tipos de Dados MySql

Números Inteiros (exatos)

Tipo	Bytes	Descrição
TINYINT	1	Pequenos inteiros (−128 a 127 ou 0 a 255)
SMALLINT	2	Inteiros pequenos
MEDIUMINT	3	Inteiros médios
INT ou INTEGER	4	Inteiro padrão
BIGINT	8	Inteiros grandes

Modificadores possíveis: UNSIGNED, ZEROFILL

Números de Ponto Flutuante e Precisão Fixa

Tipo	Descrição
FLOAT(p)	Ponto flutuante de precisão simples (32 bits)
DOUBLE ou DOUBLE PRECISION	Ponto flutuante duplo (64 bits)
DECIMAL(M,D) ou NUMERIC(M,D)	Número exato com precisão definida

Ex: DECIMAL(10,2) = 10 dígitos no total, 2 após a vírgula

Texto com comprimento fixo e variável

Tipo	Máximo de caracteres	Descrição
CHAR(n)	Até 255	Texto de tamanho fixo
VARCHAR(n)	Até 65.535 (ver observação)	Texto de tamanho variável

O limite do VARCHAR depende do charset (por ex., utf8mb4 pode ocupar até 4 bytes por caractere).

Campos de Texto Longo (TEXT)

Tipo	Tamanho máximo
TINYTEXT	255 bytes
TEXT	65.535 bytes (64 KB)
MEDIUMTEXT	16.777.215 bytes
LONGTEXT	4.294.967.295 bytes

Tipos ENUM e SET

Tipo	Descrição
ENUM('a','b',...)	Lista predefinida de valores únicos
SET('a','b',...)	Conjunto de valores múltiplos de uma lista fixa

TIPOS DE DATA E HORA

Tipo	Descrição	Faixa de valores
DATE	Data (YYYY-MM-DD)	1000-01-01 a 9999-12-31
TIME	Hora (HH:MM:SS)	-838:59:59 a 838:59:59
DATETIME	Data e hora	1000-01-01 00:00:00 a 9999...
TIMESTAMP	Data/hora com fuso e UTC	1970-01-01 00:00:01 UTC em diante
YEAR	Apenas o ano (YYYY)	1901 a 2155

Todos podem aceitar frações de segundos (ex: DATETIME(3) = milissegundos)

TIPO LÓGICO

Tipo	Descrição
BOOLEAN	Alias de TINYINT(1) (0 ou 1)
BOOL	Alias de TINYINT(1)

Dados binários fixos ou variáveis

Tipo	Descrição
BINARY(n)	Dados binários de comprimento fixo
VARBINARY(n)	Dados binários de comprimento variável

Objetos binários grandes (BLOBs)

Tipo	Tamanho máximo
TINYBLOB	255 bytes
BLOB	65.535 bytes
MEDIUMBLOB	16.777.215 bytes
LONGBLOB	4.294.967.295 bytes

EXTRAS E OBSERVAÇÕES

- ✓ NVARCHAR não é um tipo documentado oficialmente no MySQL, mas é aceito como sinônimo de VARCHAR CHARACTER SET utf8` em algumas distribuições.
- ✓ Charset recomendado: utf8mb4 (Unicode completo, incluindo emojis).
- ✓ Collation recomendado: utf8mb4_general_ci ou utf8mb4_unicode_ci.

Sub-Linguagem (DDL) – Data Definition Language

São comandos usados para **criar**, **alterar** e **remover estruturas** do banco de dados, como tabelas e bancos em si.

Os principais comando desta linguagem são:

- **CREATE** - comando para criar objeto
- **DROP** - comando para apagar objeto
- **ALTER** - comando para modificar objeto
- **TRUNCATE** - apaga todos os registros de uma tabela

CREATE

Comando CREATE é utilizado para criar objeto.

Sintaxe: **create** <tipo_objeto> <nome_objeto> [;]

Vamos relembrar o comando para criar uma base de dados:

- **create database** dbNome;

Create Table

Uma base de dados não requer parâmetros obrigatórios, diferentes da criação de uma tabela.

Sintaxe: **create table** <nome_tabela> (<parâmetro1>, <parâmetro1>,...) [;]

- **Create** - comando para criar objeto
- **Table** - objeto do tipo tabela
- <nome_tabela> - aqui colocamos o nome da tabela
- () – entre os parênteses colocamos os parâmetros da tabela
- <parâmetro> - os parâmetros de uma tabela são os campos “colunas/atributos”
- ; - o ponto e vírgula marca o término do comando, no MySql, são obrigatórios

Falando sobre parâmetros, um campo em uma tabela, vejamos como declarar um atributo:

Sintaxe: <nome_campo> <tipo> <restrições>

Onde:

- <nome_campo> - é o nome do campo propriamente dito, exemplo: “Id, Nome, ...”
- <tipo> - são os tipos de dados que serão armazenados neste campo, exemplo: “int, numeric, varchar”

- <restrições> - são as regras para inserir um dado ou se ele pode ficar sem preenchimento, exemplo: “null, not null, ...”

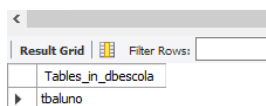
Exemplo:

```
create table tbAluno(  
AlunoID int not null,  
Nome varchar(50) null,  
Endereco varchar(50) null,  
Estado char(2) not null  
);
```

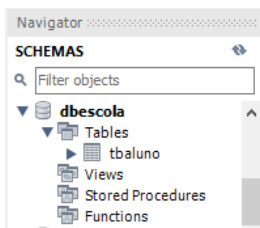
Show Tables

Comando para visualizar as tabelas existente na base de dados em uso é o comando show, veja exemplo a seguir:

```
show tables;
```



No MySql Workbench podemos visualizar na janela navigator, veja exemplo a seguir:



Describe

Retorna as informações básicas de metadados de uma tabela. As informações de metadados incluem o nome da coluna, o tipo de coluna e o comentário da coluna. Opcionalmente, você pode especificar uma especificação de partição ou um nome de coluna para retornar os metadados pertencentes a uma partição ou coluna, respectivamente.

Sintaxe:

```
describe tbAluno;
```

	Field	Type	Null	Key	Default	Extra
▶	AlunoID	int	NO		NULL	
	Nome	varchar(50)	YES		NULL	
	Endereco	varchar(50)	YES		NULL	
	Estado	char(2)	YES		NULL	

ALTER

Comandos da sub-linguagem SQL DDL, o comando ALTER é utilizado para alterar/modificar objeto.

Sintaxe: `alter table` <nome_tabela> <tipo_alteração> [`column`] <coluna_p_alteração> [<tipo> <restrições>] [;]

- **<nome_tabela>**: aqui colocamos o nome da tabela a qual deve ser alterada, não coloque os sinais de < e >.
 - **<tipo_alteração>** [`DROP` | `ADD` | `MODIFY`]
 - [`column`]: comando que indica que é uma coluna.
 - **<coluna_p_alteração>**: nome da coluna que será alterada.
 - **<tipo>**: são os tipos de dados que serão armazenados neste campo, exemplo: "int, numeric, varchar"
 - **<restrições>**: são as regras para inserir um dado ou se ele pode ficar sem preenchimento, exemplo: "null, not null, ..."
 - [;]: determina o termino do comando sql.
-
- `DROP` : apaga um objeto/atributo
 - `ADD` : adiciona um objeto/atributo
 - `MODIFY`: altera um objeto/atributo

Exemplo excluindo uma coluna existente na tabela:

```
alter table tbAluno drop Estado;
```

Exemplo incluindo uma coluna na tabela:

```
alter table tbAluno add Estado char (2) null;
```

Exemplo (1) alterando uma coluna na tabela:

```
alter table tbAluno modify Estado int;
```

Exemplo (2) alterando uma coluna na tabela:

```
alter table tbAluno modify column Estado char (2) null;
```

DROP TABLE

O que faz DROP TABLE?

- ✓ **Apaga a tabela** do banco de dados.
- ✓ **Remove todos os dados**, a **estrutura** e os **índices** associados.
- ✓ **Não pode ser desfeito** (a menos que você tenha backup).

Sintaxe:

```
DROP TABLE <nome_da_tabela> [;]
```

Exemplo:

```
DROP TABLE alunos_tb;
```

Esse comando vai apagar a tabela alunos do banco.

Com verificação de existência (opcional):

Para evitar erro caso a tabela não exista:

```
DROP TABLE IF EXISTS alunos_tb;
```

Isso é útil para scripts que podem ser executados várias vezes sem gerar erro.

Remover várias tabelas ao mesmo tempo:

```
DROP TABLE IF EXISTS tabela1, tabela2, tabela3;
```

Atenção:

O DROP apaga **definitivamente**. Se quiser apenas **limpar os dados**, use o truncate.

TRUNCATE

O que TRUNCATE faz?

- ✓ Remove todos os registros da tabela de forma rápida e eficiente.
- ✓ Mantém a estrutura da tabela (colunas, tipos, índices etc.).
- ✓ Reinicia o contador AUTO_INCREMENT, se houver.
- ✓ Não registra linha por linha no log de transações (por isso é mais rápido que DELETE).

Sintaxe:

```
TRUNCATE TABLE <nome_da_tabela> [;]
```

Exemplo:

```
TRUNCATE TABLE alunos_tb;
```

Isso apaga todos os alunos cadastrados, mas a tabela continua existindo.

Diferença entre DELETE, TRUNCATE e DROP

Comando	Apaga dados?	Apaga estrutura?	Pode ser desfeito?	Reinicia AUTO_INCREMENT?
DELETE	Sim (especificamente ou todos)	Não	Sim (se houver transações)	Não
TRUNCATE	Sim (todos os registros)	Não	Não	Sim
DROP	Sim	Sim	Não	Sim (porque apaga tudo)

Cuidados:

- ✓ TRUNCATE **não pode ter cláusula WHERE**. Ele **apaga tudo de uma vez só**.
- ✓ **Não dispara triggers ON DELETE** (em algumas versões do MySQL).
- ✓ **Mais rápido** que DELETE FROM tabela, especialmente para tabelas grandes.

RENAME

RENAME – Renomeando Tabelas e Colunas no MySQL

O comando RENAME no MySQL é utilizado para **alterar o nome de uma tabela** existente. A sintaxe é direta e de fácil aplicação.

Sintaxe renomear uma tabela

RENAME TABLE <nome_atual_da_tabela> **TO** <nome_novo_da_tabela> [;]

Sintaxe renomear uma coluna de uma tabela

Para renomear **colunas**, utiliza-se o comando ALTER TABLE com a cláusula RENAME COLUMN. A sintaxe é:

ALTER TABLE <nome_da_tabela> **RENAME COLUMN** <nome_antigo> **TO** <nome_novo> [;]

Exemplo:

ALTER TABLE Pessoa_tb **RENAME COLUMN** ID **TO** PessoaID;

Esse comando altera o nome da coluna ID para PessoaID na tabela Pessoa_tb.

INDEX

Os **índices** são estruturas utilizadas para **acelerar a recuperação de dados** em tabelas. Eles funcionam como atalhos que permitem que o banco de dados localize registros com maior rapidez durante buscas ou consultas.

Embora os usuários **não visualizem diretamente os índices**, sua presença pode **aumentar significativamente o desempenho** de operações SELECT, especialmente em tabelas com grandes volumes de dados.

Atenção: A inserção, atualização e exclusão de dados em tabelas indexadas pode ser **mais lenta**, pois o MySQL precisa atualizar os índices associados. Por isso, é recomendado criar índices **apenas em colunas frequentemente utilizadas em filtros (WHERE), ordenações (ORDER BY) ou junções (JOIN)**.

Sintaxe para criar um índice

CREATE INDEX <nome_do_indice> **ON** <nome_da_tabela> (<nome_da_coluna>) [;]

Exemplo:

CREATE INDEX idx_Nome **ON** Pessoa_tb (Name);

Neste exemplo, estamos criando um índice chamado idx_Nome na coluna Name da tabela Pessoa.

Índices podem ser:

- ✓ **Simple:** aplicados a uma única coluna;
- ✓ **Compostos:** aplicados a múltiplas colunas (ex: **CREATE INDEX** idx_multi **ON** tabela (col1, col2)).

SQL Constraints (Restrições)

As restrições são regras aplicadas nas colunas de uma tabela, usadas para limitar os tipos de dados que são inseridos.

Podem ser especificadas no momento de criação da tabela (**CREATE**) ou após a tabela ter sido criada (**ALTER**).

As principais constraints são as seguintes:

- **NOT NULL**
- **UNIQUE**
- **DEFAULT**
- **CHECK**
- **PRIMARY KEY**
- **FOREIGN KEY**

NOT NULL

A constraint **NOT NULL** impõe a uma coluna a **NÃO** aceitar valores **NULL**, ou seja, obriga um campo a sempre possuir um valor. Deste modo, não é possível inserir um registro (ou atualizar) sem entrar com um valor neste campo.

Exemplo de sintaxe utilizando a restrição not null em uma coluna:

```
create table tbconstraint(  
NomeRestricao varchar (60) not null  
);
```



UNIQUE


A restrição **UNIQUE** identifica de forma única cada registro em uma tabela de um banco de dados. As constraints **UNIQUE** e **PRIMARY KEY** garantem a unicidade em uma coluna ou conjunto de colunas.

Uma constraint **PRIMARY KEY** automaticamente possui uma restrição **UNIQUE** definida, portanto não é necessário especificar essa constraint neste caso.

É possível termos várias constraints UNIQUE em uma mesma tabela, mas apenas uma Chave Primária por tabela (lembrando que uma PK pode ser composta, ou seja, constituída por mais de uma coluna – mas ainda assim, será uma única chave primária).

Exemplo de sintaxe utilizando a restrição unique em uma coluna:

```
create table tbconstraint(  
NomeRestricao varchar (60) not null  
Cnpj bigint unique  
);
```




DEFAULT

A restrição DEFAULT é usada para inserir um valor padrão especificado em uma coluna. O valor padrão será adicionado a todos os novos registros caso nenhum outro valor seja especificado na hora de inserir dados.

Exemplo de sintaxe utilizando a restrição default em uma coluna:

```
CREATE TABLE Pessoa (  
Cidade varchar(255) DEFAULT 'São Paulo'  
);
```




CHECK

A CHECK restrição limita o valor que pode ser colocado em uma coluna.

Exemplo de sintaxe utilizando a restrição check em uma coluna:

```
CREATE TABLE Pessoa (  
Idade int,  
CHECK (Idade>=18)  
);
```



PRIMARY KEY

A restrição PRIMARY KEY (Chave Primária) identifica de forma única cada registro em uma tabela de banco de dados.

As Chaves Primárias devem sempre conter valores únicos.

Uma coluna de chave primária não pode conter valores NULL

Cada tabela deve ter uma chave primária e apenas uma chave primária.

Exemplo de sintaxe utilizando a restrição primary key em uma coluna:

```
CREATE TABLE Pessoa (  
  ID int PRIMARY KEY  
);
```



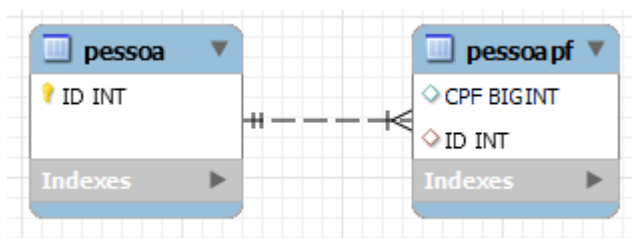
FOREIGN KEY

Uma FOREIGN KEY (Chave Estrangeira) em uma tabela é um campo que aponta para uma chave primária em outra tabela. Desta forma, é usada para criar os relacionamentos entre as tabelas no banco de dados.¹

Exemplo de sintaxe utilizando a restrição foreign key em uma coluna:

```
CREATE TABLE Pessoa (  
  ID int PRIMARY KEY  
);
```

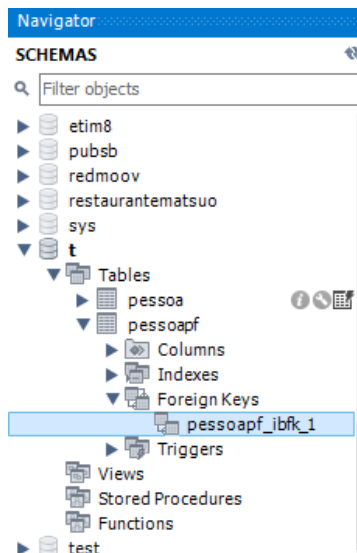
```
CREATE TABLE PessoaPF (  
  CPF bigint,  
  ID int,  
  foreign key (ID) references Pessoa(ID)  
);
```



Vídeo sobre Restrições:

<https://www.youtube.com/watch?v=10Gu4XZBMdo&t=3s>

¹ [http://www.bosontreinamentos.com.br/mysql/mysql-constraints-restricoes-primary-key-fk-default-etc-06/#:~:text=SQL%20Constraints%20\(Restri%C3%A7%C3%B5es\)%20no%20MySQL,de%20dados%20que%20s%C3%A3o%20inseridos.](http://www.bosontreinamentos.com.br/mysql/mysql-constraints-restricoes-primary-key-fk-default-etc-06/#:~:text=SQL%20Constraints%20(Restri%C3%A7%C3%B5es)%20no%20MySQL,de%20dados%20que%20s%C3%A3o%20inseridos.)

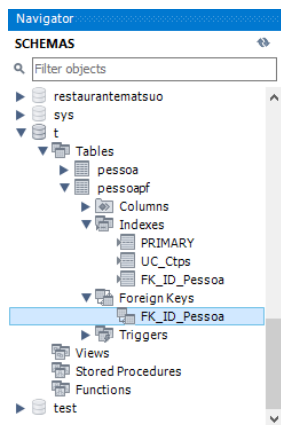


Como podemos ver na aba de navegação em schemas, que as restrições são objetos da tabela nomeados automaticamente!

Mas é possível nomeá-los? Sim, é possível!

Para permitir a nomeação de uma restrição podemos utilizar o comando **CONSTRAINT**.

Exemplo de sintaxe utilizando restrições nomeadas:



```
CREATE TABLE PessoaPF (
    ID int not null,
    CPF bigint,
    CTPS int,
    Idade smallint,
    CONSTRAINT CHK_Idade CHECK (Idade>=18),
    CONSTRAINT UC_Ctps UNIQUE (CTPS),
    CONSTRAINT PK_PessoaPF PRIMARY KEY (CPF),
    CONSTRAINT FK_ID_Pessoa foreign key (ID) references Pessoa(ID)
);
```

Veja um exemplo apagando uma FOREIGN KEY:

```
ALTER TABLE PessoaPF DROP FOREIGN KEY FK_ID_Pessoa;
```

AUTO_INCREMENT

O incremento automático “**AUTO_INCREMENT**” permite que um número único seja gerado automaticamente quando um novo registro é inserido em uma tabela.

Muitas vezes este é o campo de chave primária que gostaríamos que fosse criado automaticamente toda vez que um novo registro fosse inserido.

Exemplo de sintaxe utilizando a restrição primary key e o auto_increment:

```
CREATE TABLE Pessoa (  
    PessoaId int NOT NULL AUTO_INCREMENT,  
    primary key (PessoaId)  
);
```



FUNÇÕES ÚTEIS NO MYSQL

O MySQL oferece diversas **funções internas** que podem ser utilizadas para manipular e obter informações do sistema, como data, hora, usuário atual, entre outras.

Abaixo, destacamos algumas das mais utilizadas:

Funções de Data e Hora

Função	Descrição
CURRENT_DATE()	Retorna a data atual (formato: YYYY-MM-DD)
CURRENT_TIME()	Retorna a hora atual (formato: HH:MM:SS)
CURRENT_TIMESTAMP()	Retorna a data e hora atuais (timestamp completo)

Função de Usuário

Função	Descrição
CURRENT_USER()	Retorna o usuário atual logado no sistema MySQL

Exemplo prático com CURRENT_TIMESTAMP():

```
77 • select current_timestamp();  
78
```

The screenshot shows a MySQL query result grid. The query is 'select current_timestamp();'. The result is a single row with the value '2022-03-29 22:03:53'.

Essas funções são muito úteis em registros automáticos de logs, criação de campos created_at, auditorias e controle de sessões.

Sub-Linguagem (DML) – Data Manipulation Language

A manipulação de dados significa:

- a busca da informação armazenada no BD;
- a inserção de novas informações nos BD;
- a eliminação de informações no BD;
- a modificação de dados armazenados no BD.

Linguagem de Manipulação de Dados (DML, de Data Manipulation Language) é uma família de linguagens de computador utilizada para a recuperação, inclusão, remoção e modificação de informações em bancos de dados. Pode ser procedural, que especifica como os dados devem ser obtidos do banco; pode também ser declarativa (não procedural), em que os usuários não necessitam especificar o caminho de acesso, isto é, como os dados serão obtidos. O padrão SQL é não procedural. DMLs foram utilizadas inicialmente apenas por programas de computador, porém (com o surgimento da SQL) também têm sido utilizadas por pessoas.

Principais comandos

As DMLs têm sua capacidade funcional organizada pela palavra inicial em uma declaração, a qual é quase sempre um verbo. No caso da SQL, estes verbos são:

- Insert
- Update
- Delete

Cada declaração SQL é um comando declarativo. As declarações individuais da SQL são declarativas, em oposição às imperativas, na qual descrevem o que o programa deveria realizar, em vez de descrever como ele deveria realizar. Muitas implementações de banco de dados SQL estendem suas capacidades SQL fornecendo linguagens imperativas, isto é, procedurais. Exemplos destas implementações são o PL/SQL, da Oracle, e o SQL PL, da DB2. As linguagens de manipulação de dados tendem a ter muitos tipos diferentes e capacidades entre distribuidores de banco de dados. Há um padrão estabelecido para a SQL pela ANSI, porém os distribuidores ainda fornecem suas próprias extensões ao padrão enquanto não implementam o padrão por completo.²

INSERT

A instrução SQL INSERT INTO é usada para inserir novos registros em uma tabela.

É possível escrever a INSERT INTO declaração de duas maneiras:

1. Especifique os nomes das colunas e os valores a serem inseridos:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

2. Se você estiver adicionando valores para todas as colunas da tabela, não precisará especificar os nomes das colunas na consulta SQL. No entanto, certifique-se de que a ordem dos valores esteja na mesma ordem das colunas na tabela.

² https://pt.wikipedia.org/wiki/Linguagem_de_manipula%C3%A7%C3%A3o_de_dados

A sintaxe seria a seguinte:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

DELETE

A instrução DELETE é usada para excluir registros existentes em uma tabela.

Sintaxe:

```
DELETE FROM table_name WHERE condição;
```

Nota: Tenha cuidado ao excluir registros em uma tabela! Observe a WHERE cláusula na declaração DELETE. A WHERE cláusula especifica quais registros devem ser excluídos. Se você omitir a cláusula WHERE, todos os registros da tabela serão excluídos!

Excluir todos os registros

É possível excluir todas as linhas de uma tabela sem excluir a tabela. Isso significa que a estrutura, os atributos e os índices da tabela estarão intactos:

Sintaxe:

```
DELETE FROM table_name;
```

Nota: “No Mysql” caso utilize a declaração DELETE sem a cláusula WHERE, ocorrerá um erro por conta do modo de atualização segura, código de ERRO MySQL 1175, veja a seguir como funciona o sql_safe_updates

SQL_SAFE_UPDATES


O MySQL ERROR code 1175 é acionado quando você tenta atualizar ou excluir dados de uma tabela sem usar uma WHERE cláusula.

O MySQL tem um modo de atualização seguro para evitar que os administradores emitam uma instrução UPDATE ou DELETE sem uma cláusula WHERE.

Você pode ver se o modo de atualização segura está habilitado em seu servidor MySQL verificando a variável global sql_safe_updates.

Verifique a variável global usando a SHOW VARIABLES instrução da seguinte forma:

```
13 • SHOW VARIABLES LIKE "sql_safe_updates";
14
```



Variable_name	Value
sql_safe_updates	ON

O exemplo acima mostra que `sql_safe_updates` é ON, então uma instrução UPDATE ou DELETE sem a cláusula WHERE causará o erro 1175.

Para corrigir o erro, você pode desabilitar o modo de atualização segura ou seguir a descrição do erro adicionando uma cláusula WHERE que usa uma coluna KEY.

Você pode usar a SET instrução para desabilitar a atualização segura conforme mostrado abaixo:

```
set sql_safe_updates = 0;
```

Agora você deve ser capaz de executar uma instrução UPDATE ou DELETE sem uma cláusula WHERE.³

Se você quiser ativar o modo de atualização segura novamente, você pode SET usar a variável global como mostrado abaixo:

```
set sql_safe_updates = 1;
```

UPDATE

A instrução UPDATE é usada para modificar os registros existentes em uma tabela.

Sintaxe

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condição;
```

Nota: Tenha cuidado ao atualizar registros em uma tabela! Observe a cláusula WHERE na declaração UPDATE. A cláusula WHERE especifica quais registros devem ser atualizados. Se você omitir a cláusula WHERE, todos os registros da tabela serão atualizados!

³ <https://sebbastian.com/mysql-error-code-1175/>

Sub-Linguagem (DQL) - Data Query Language

Structured Query Language, ou Linguagem de Consulta Estruturada ou SQL, é a linguagem de pesquisa declarativa padrão para banco de dados relacional (base de dados relacional). Muitas das características originais do SQL foram inspiradas na álgebra relacional.

SELECT

A instrução SELECT do MySQL é usada para selecionar dados de um banco de dados.

Os dados retornados são armazenados em uma tabela de resultados, chamada de conjunto de resultados.

O SELECT **não altera dados**, apenas os **lê e apresenta** com base nos critérios da consulta.

Sintaxe SELECT

```
SELECT column1, column2, ...  
FROM table_name;
```

Aqui, coluna1, coluna2, ... são os nomes dos campos da tabela da qual você deseja selecionar os dados. Se você deseja selecionar todos os campos disponíveis na tabela, use a seguinte sintaxe: ⁴

```
SELECT * FROM table_name;
```

Cláusula WHERE – Filtrando dados

Permite selecionar **apenas os registros que atendem a uma condição**.

Exemplo:

```
SELECT * FROM alunos_tb WHERE idade > 18;
```

⁴ MySQL Documentation: <https://dev.mysql.com/doc/>
SILBERSCHATZ, A.; KORTH, H. *Sistemas de Banco de Dados*.
W3Schools SQL: <https://www.w3schools.com/sql/>

Operadores

Operador	Significado
=	Igual
>, <	Maior, menor
>=, <=	Maior/menor igual
<> ou !=	Diferente
LIKE	Pesquisa textual (com % ou _)
IN	Dentro de um conjunto

ORDER BY

Exibe os dados em uma **ordem específica** (**[ASC]** crescente ou **[DESC]** decrescente).

Exemplos:

```
SELECT nome, idade FROM alunos ORDER BY idade ASC;  
SELECT nome FROM alunos ORDER BY nome DESC;
```

GROUP BY

Funções Agregadas (com GROUP BY)

Permitem **resumir dados numéricos**. As principais são:

Função	Descrição
COUNT()	Conta registros
SUM()	Soma valores
AVG()	Média
MAX()	Maior valor
MIN()	Menor valor

Exemplo:

```
SELECT curso, COUNT(*) AS total FROM alunos_tb GROUP BY curso;
```

Exemplos Prática no MySQL

Supondo que temos a tabela `alunos_tb`:

```
CREATE TABLE alunos_tb (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  nome VARCHAR(100),  
  idade INT,  
  curso VARCHAR(100)  
);
```

```
INSERT INTO alunos_tb (nome, idade, curso)
VALUES ('Carlos', 18, 'Informática'),
       ('Ana', 19, 'Administração'),
       ('Mariana', 20, 'Informática'),
       ('João', 17, 'Enfermagem'),
       ('Letícia', 18, 'Administração');
```

Consultas práticas:

- Todos os dados:

```
SELECT * FROM alunos_tb;
```

- Apenas nomes e idades:

```
SELECT nome, idade FROM alunos_tb;
```

;

- Alunos de Informática:

```
SELECT nome FROM alunos_tb WHERE curso = 'Informática';
```

- Alunos com idade acima de 18:

```
SELECT nome, idade FROM alunos_tb WHERE idade > 18;
```

- Contar quantos alunos há por curso:

```
SELECT curso, COUNT(*) AS total FROM GROUP BY curso;
```

- Listar alunos por idade decrescente:

```
SELECT nome, idade FROM alunos_tb ORDER BY idade DESC;
```

Alias

Alias (Apelidos). Você pode **renomear colunas e funções** para deixar os resultados mais claros.

```
SELECT nome AS "Nome do Aluno", idade AS "Idade (anos)"
FROM alunos_tb;
```

Stored Procedures

<Em construção/ atualização> !...aguarde.....