Cábula de SQL · MariaDB

Bases · Base de dados · Tabelas · Alterar dados · Ajuda

Holger Sebastião · www.sebastiao.org
Versão 1.0 · 2026-10-25

Conceitos — base de dados relacional

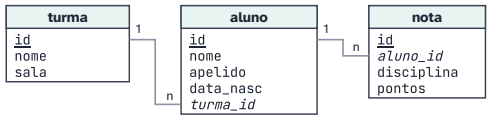
Tabela aluno

id	nome	apelido	data_nasc	turma_id
1	Ana	Silva	2008-03-14	1
2	Rui	Costa	2007-11-02	2
3	Inês	Sousa	2008-06-30	1

chave primária — *amarelo: uma linha = um registo; cada coluna = um atributo (campo)* — *chave estrangeira → turma.id*

Chave primária	torna cada linha única, nunca repetida, nunca vazia
Chave estrangeira	aponta para a chave primária de outra tabela
SGBD	o programa que gere a base de dados: aqui, o MariaDB

A nossa base de dados de exemplo — escola



sublinhado = chave primária, *itálico* = chave estrangeira. Uma turma tem muitos alunos, um aluno tem muitas notas.

Regras de escrita

- Cada instrução termina com `;`
- Palavras-chave em maiúsculas (`SELECT`), nomes em minúsculas: lê-se melhor.
- Texto e datas entre plicas: `'Ana'`, `'2008-03-14'`
- Comentário: `-- até ao fim da linha` (espaço depois de `--`)
- Nomes sem acentos nem espaços: `aluno`, `turma_id`

Onde escrevo SQL?

- **phpMyAdmin** (por exemplo, com XAMPP): escolher a base de dados → separador **SQL** → escrever a instrução → Executar.
- **Linha de comandos:** entrar com `mariadb -u root -p` (no XAMPP: `mysql -u root -p`), sair com `exit`.

Criar a base de dados

```
CREATE DATABASE escola
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_unicode_ci;
USE escola;   -- daqui em diante, em escola
```

`SHOW DATABASES;`	todas as bases de dados
`SHOW TABLES;`	todas as tabelas da base de dados
`DESCRIBE aluno;`	colunas e tipos de dados
`DROP DATABASE escola;`	apaga **tudo**, sem perguntar

`utf8mb4` guarda todos os carateres (ç, ã, emojis). `unicode_ci` ordena segundo as regras Unicode, `ci`: maiúsculas e minúsculas tanto faz.

Tipos de dados

TIPO	PARA · EXEMPLO
`INT`	número inteiro: `42`
`DECIMAL(6,2)`	dinheiro, 6 dígitos, 2 decimais: `1234.50`
`DOUBLE`	número decimal (medição): `3.14`
`VARCHAR(50)`	texto até 50 carateres: `'Ana'`
`CHAR(8)`	texto, sempre exatamente 8: `'1000-001'`
`TEXT`	texto longo
`DATE`	data: `'2026-10-25'`
`TIME`	hora: `'07:45:00'`
`DATETIME`	data e hora: `'2026-10-25 07:45:00'`
`BOOLEAN`	verdadeiro/falso, guardado como 1/0: `TRUE`

! Decimais com **ponto**: `3.5`. Datas sempre `'AAAA-MM-DD'`, não `'25/10/2026'`.

Códigos postais e telefones são texto, não números: senão perde-se o 0 inicial.

NULL quer dizer: sem valor

`NULL` **não** é `0` nem `''` (texto vazio), mas sim «desconhecido». Campos sem `NOT NULL` podem ficar vazios. Testar com `IS NULL` (página 2).

Criar uma tabela — CREATE TABLE

```
CREATE TABLE turma (
  id INT PRIMARY KEY AUTO_INCREMENT,
  nome VARCHAR(10) NOT NULL UNIQUE,
  sala VARCHAR(10)
);

CREATE TABLE aluno (
  id INT PRIMARY KEY AUTO_INCREMENT,
  nome VARCHAR(50) NOT NULL,
  apelido VARCHAR(50) NOT NULL,
  data_nasc DATE,
  turma_id INT,
  FOREIGN KEY (turma_id)
    REFERENCES turma(id)
);
```

`AUTO_INCREMENT`	o MariaDB numera o id sozinho: 1, 2, 3 …
`NOT NULL`	campo obrigatório, não pode ficar vazio
`UNIQUE`	nenhum valor repetido
`DEFAULT 0`	valor por defeito
`FOREIGN KEY`	só ids que existem em `turma`

Ordem: criar primeiro `turma`, depois `aluno`.

Alterar e apagar tabelas

```
-- nova coluna
ALTER TABLE aluno ADD email VARCHAR(100);
-- mudar o tipo de dados
ALTER TABLE aluno MODIFY email TEXT;
-- apagar uma coluna
ALTER TABLE aluno DROP COLUMN email;
-- apagar a tabela com os dados
DROP TABLE nota;
```

Implementar relações

1:n chave estrangeira do lado n. **n:m** tabela intermédia:

```
CREATE TABLE inscricao (
  aluno_id INT,
  clube_id INT,
  PRIMARY KEY (aluno_id, clube_id),
  FOREIGN KEY (aluno_id)
    REFERENCES aluno(id),
  FOREIGN KEY (clube_id) REFERENCES clube(id)
);
```

Inserir dados — INSERT

```
INSERT INTO turma (nome, sala)
VALUES ('11IT1', 'B204'),
       ('12IT1', 'B206');

INSERT INTO aluno (nome, apelido,
  data_nasc, turma_id)
VALUES ('Ana', 'Silva', '2008-03-14', 1);
```

Deixar o `id` de fora: o `AUTO_INCREMENT` atribui-o. Colunas e valores pela **mesma ordem**.

Alterar dados — UPDATE

```
UPDATE aluno
SET turma_id = 2, apelido = 'Santos'
WHERE id = 1;
```

Apagar dados — DELETE

```
DELETE FROM aluno
WHERE id = 3;
```

! **Sem** `WHERE`, o `UPDATE` altera e o `DELETE` apaga **todas** as linhas. Não há como desfazer. Dica: verificar primeiro com `SELECT … WHERE …` que linhas são afetadas.

`DELETE FROM t;`	todas as linhas fora, a tabela fica
`TRUNCATE t;`	como acima, o id recomeça em 1
`DROP TABLE t;`	a tabela e a estrutura desaparecem

Mensagens de erro típicas

`1064`	erro de escrita: verificar vírgula, parêntese, `;` ou `'`. A mensagem mostra o sítio **depois** do erro.
`1452`	a chave estrangeira aponta para um id que não existe
`1451`	a linha ainda é usada noutra tabela: apagar lá primeiro
`1062`	valor repetido em `PRIMARY KEY` ou `UNIQUE`
`1146`	a tabela não existe: verificar o nome ou o `USE`

Cábula de SQL · MariaDB

Consultas · Filtrar · Agregar · Juntar · Ajuda

Holger Sebastião · www.sebastiao.org
Versão 1.0 · 2026-10-25

Estrutura de uma consulta — SELECT

```
SELECT   colunas          -- o quê?
FROM     tabela           -- de onde?
WHERE    condicao         -- que linhas?
GROUP BY coluna           -- formar grupos
HAVING   condicao         -- que grupos?
ORDER BY coluna           -- ordenar
LIMIT    quantidade;      -- quantas?
```

Só `SELECT` e `FROM` são obrigatórios. A **ordem** é fixa. Para memorizar: **S-F-W-G-H-O-L**, de cima para baixo.

Escolher colunas

```
SELECT * FROM aluno;        -- todas

SELECT nome, apelido
FROM aluno;

SELECT DISTINCT turma_id    -- sem repetidos
FROM aluno;

SELECT apelido AS Nome      -- alias
FROM aluno;
```

Ordenar e limitar

```
SELECT nome, apelido
FROM aluno
ORDER BY apelido ASC, nome ASC
LIMIT 10;
```

`ASC` crescente (A–Z, 1–9, por defeito), `DESC` decrescente. Em caso de empate, decide a segunda coluna. `LIMIT 10`: só as primeiras 10 linhas.

Funções úteis

`CONCAT(nome, ' ', apelido)`	juntar texto
`UPPER(nome)` · `LOWER`	maiúsculas · minúsculas
`LENGTH(nome)`	comprimento
`CURDATE()` · `NOW()`	hoje · agora
`YEAR(data_nasc)`	só o ano

```
SELECT nome,
  TIMESTAMPDIFF(YEAR, data_nasc,
                CURDATE()) AS anos
FROM aluno;     -- idade em anos
```

Filtrar linhas — WHERE

```
SELECT nome, apelido
FROM aluno
WHERE turma_id = 1
  AND data_nasc >= '2008-01-01';
```

`= <>`	igual, diferente (também `!=`)
`< > <= >=`	menor, maior …
`AND`	as duas condições têm de ser verdadeiras
`OR`	pelo menos uma tem de ser verdadeira
`NOT`	inverte
`BETWEEN 10 AND 15`	de 10 a 15, os dois limites **incluídos**
`IN ('Matemática', 'Português')`	um da lista
`IS NULL`	o campo está vazio
`IS NOT NULL`	o campo está preenchido

! `= NULL` **nunca** encontra nada. Usa sempre `IS NULL`.

AND antes de OR: com condições misturadas, usa parênteses:

```
WHERE (disciplina = 'Matemática'
       OR disciplina = 'Português')
  AND pontos < 5
```

Procurar texto — LIKE

```
SELECT * FROM aluno
WHERE apelido LIKE 'Sil%';
```

`%`	quantos carateres quiseres, até nenhum
`_`	exatamente um caráter
`'Sil%'`	começa por Sil: Silva, Silveira
`'%eira'`	acaba em eira: Oliveira, Pereira
`'%e%'`	tem um e em qualquer sítio
`'S_usa'`	Sousa, mas não Souza

Contar e calcular

```
SELECT COUNT(*) AS total
FROM aluno
WHERE turma_id = 1;
```

`COUNT(*)`	número de linhas
`COUNT(coluna)`	campos preenchidos (sem NULL)
`SUM(pontos)`	soma
`AVG(pontos)`	média
`MIN(pontos)`	valor mais baixo
`MAX(pontos)`	valor mais alto
`ROUND(AVG(pontos), 1)`	arredondar a 1 casa decimal

Formar grupos — GROUP BY

```
SELECT disciplina, ROUND(AVG(pontos), 1)
         AS media
FROM nota
GROUP BY disciplina
HAVING AVG(pontos) >= 8
ORDER BY media DESC;
```

DISCIPLINA	MEDIA
Informática	11.4
Português	9.2
Inglês	8.0

`GROUP BY disciplina`: uma linha de resultado **por disciplina**, a função calcula por grupo.

WHERE ou HAVING? `WHERE` filtra linhas **antes** de agrupar, `HAVING` filtra grupos **depois**. `COUNT`, `AVG` … só no `HAVING`.

! Cada coluna do `SELECT` vai no `GROUP BY` ou numa função. Senão, o MariaDB mostra um valor qualquer, sem erro.

Subconsulta

```
SELECT nome, apelido
FROM aluno
WHERE id IN (SELECT aluno_id
             FROM nota
             WHERE pontos = 15);
```

A consulta interior corre primeiro e dá uma lista de ids.

Juntar tabelas — JOIN

```
SELECT aluno.nome, turma.nome
FROM aluno
INNER JOIN turma
  ON aluno.turma_id = turma.id;
```

NOME	NOME
Ana	11IT1
Rui	12IT1
Inês	11IT1

Depois de `ON`: **chave estrangeira = chave primária**. `tabela.coluna` indica a tabela.



INNER JOIN: só alunos que têm turma.
LEFT JOIN: **todos** os alunos; sem turma fica `NULL`.

Três tabelas

```
SELECT turma.nome, aluno.apelido,
       nota.disciplina, nota.pontos
FROM nota
INNER JOIN aluno
  ON nota.aluno_id = aluno.id
INNER JOIN turma
  ON aluno.turma_id = turma.id
WHERE turma.nome = '11IT1'
ORDER BY aluno.apelido;
```

Calcular na consulta

```
SELECT disciplina, pontos,
  ROUND((17 - pontos) / 3, 1) AS nota_escolar
FROM nota;
```

Linha a linha, com `+ - * /`. A tabela não muda.

Erros típicos

- Vírgula antes de `FROM`: `SELECT nome, FROM`
- Texto sem plicas: `WHERE disciplina = Inglês`
- `JOIN` sem `ON`: tudo combinado com tudo.
- Coluna `id` em duas tabelas: escrever `tabela.id`.
- Palavras reservadas como nomes: `order`, `key`, `group`.