for_code[ ]

[Fique conectado]

Acesse para treinar

db-fiddle.com/

Material da Capacitação

andradasdev.github.io/sql/

Siga a for_code no Instagram

instagram.com/forcodeufrj

for_code[ ]

[Capacitação de MySQL]

Fundamentos de banco de dados relacional

Ronivaldo D. Andrade - Diretor de Projetos – 27 de agosto de 2026

[Apresentação]

Ronivaldo D. Andrade

Estagiário de Análise e Desenvolvimento de Sistemas na DGOM — Marinha do Brasil

Diretor de Projetos da for_code

Matemática Aplicada — UFRJ

ronidomingues.ard@gmail.com

linkedin.com/in/ronidomingues · github.com/ronidomingues

[O trato de hoje]

Você vai sair daqui sabendo

- Criar um banco e uma tabela
- Inserir, consultar, alterar e apagar dados
- Filtrar com `WHERE`
- Ligar duas tabelas com `JOIN`

O que NÃO cabe em 2 horas

- Normalização formal
- Índices e performance
- Transações e backup

Em 2 horas ninguém vira DBA.

Mas todo mundo sai capaz de montar o banco do primeiro projeto.

[Como as 2 horas vão passar]

Bloco	Assunto	Início	Tempo
0	Abertura	00:00	5 min
1	Conceitos: dado, SGBD, relacional	00:05	15 min
2	Ambiente: todo mundo conectado	00:20	15 min
3	Criando banco e tabela	00:35	20 min
–	Pausa	00:55	5 min
4	Inserindo e consultando	01:00	25 min
5	Alterando e apagando	01:25	10 min
6	Ligando tabelas com `JOIN`	01:35	20 min
7	Erros comuns e encerramento	01:55	5 min

Dos 120 minutos, cerca de 75 são de mão na massa.

BLOCO 1

Do que estamos falando

00:05 · 15 min

[Dado, informação e banco]

Dado

Valor bruto. Sozinho não diz nada.

```
'Ana'
2024-03-11
7
```

Informação

Dado dentro de um contexto.

"Ana ingressou na liga em 11/03/2024 e está no 7º período."

Banco de dados

Coleção organizada de dados, guardada de forma persistente e estruturada.

[SGBD: quem realmente cuida dos dados]

O SGBD é o software servidor que gerencia o banco. Em inglês, DBMS.

- Grava e lê os dados em disco
- **Garante a integridade** — não deixa entrar dado que viola suas regras
- Controla o acesso concorrente — várias pessoas escrevendo sem corromper nada
- Controla as permissões de cada usuário

MySQL · PostgreSQL · SQL Server · Oracle · SQLite

[Cuidado com esta frase]

``A única forma de falar com um SGBD é por SQL.''

Isso é falso.

- A conversa real acontece por um **protocolo de rede binário** — o MySQL escuta na porta 3306
- O SQL é a linguagem que **trafega dentro** desse protocolo
- O MySQL também expõe o **X DevAPI**, uma API de CRUD sem SQL
- Quando você usa um **ORM** (Prisma, Eloquent, Hibernate), escreve objetos — o ORM traduz para SQL antes de enviar

SQL é a interface **padrão e dominante** — não a única.

[Modelo relacional]

Edgar F. Codd, 1970. Os dados vivem em tabelas.

```
TABELA membro
+----+--------------+--------------------+---------+
| id | nome         | email              | periodo |
+----+--------------+--------------------+---------+
|  1 | Ana Souza    | ana@liga.edu.br    |       7 |
|  2 | Bruno Lima   | bruno@liga.edu.br  |       3 |
+----+--------------+--------------------+---------+
```

- **Linha** (registro): uma ocorrência do mundo real
- **Coluna** (campo): uma característica dela, com **tipo fixo**
- **Chave**: o que liga uma tabela a outra — daí "relacional"

[SQL não é um bloco só]

Sigla	Para quê	Comandos
DDL	Define a **estrutura**	`CREATE ALTER DROP`
DML	Manipula os **dados**	`INSERT UPDATE DELETE`
DQL	**Consulta** os dados	`SELECT`
DCL	Controla **permissões**	`GRANT REVOKE`
TCL	Controla **transações**	`COMMIT ROLLBACK`

Hoje: DDL (uma vez) + DML e DQL (o tempo todo).

BLOCO 2

Todo mundo conectado

00:20 · 15 min

[Você não vai instalar nada]

O MySQL e o phpMyAdmin rodam na máquina do instrutor. Você acessa por uma URL, no navegador.

URL	_____
Usuário	`aluno`
Senha	`aluno`

Na primeira visita aparece uma tela de aviso do túnel.
Clique em **Visit Site** e siga.

[phpMyAdmin: onde vamos digitar]

Um cliente SQL é o programa por onde você conversa com o servidor.

- Ele abre a conexão
- Envia os comandos
- Mostra o resultado em tabela

O phpMyAdmin é uma **aplicação web** — não é programa instalado. Por isso basta o navegador.

Outros clientes

- MySQL Workbench
- DBeaver
- HeidiSQL
- `mysql` no terminal

[Mão na massa #1: crie o seu banco]

Abra a aba SQL e execute:

```
SELECT VERSION(), NOW();
```

Voltou versão e data? Está conectado. Agora crie o seu banco — troque `SEUNOME` pelo seu primeiro nome:

```
CREATE DATABASE liga_SEUNOME CHARACTER SET utf8mb4;
USE liga_SEUNOME;
```

`liga_ana` `liga_joao` `liga_carla`

[Por que cada um tem o seu banco]

Um servidor só, a turma inteira dentro.

- Se todos usassem o mesmo banco, o segundo `CREATE TABLE` já falharia
- Um `DELETE` de qualquer um apagaria o dado de todos
- O login `aluno` só tem permissão em bancos `liga_*`

✓ `CREATE DATABASE liga_ana`

× `USE cap` (banco do instrutor)

× `CREATE DATABASE bagunca`

[*]

BLOCO 3

Criando a estrutura

00:35 · 20 min

[A ordem nunca muda]

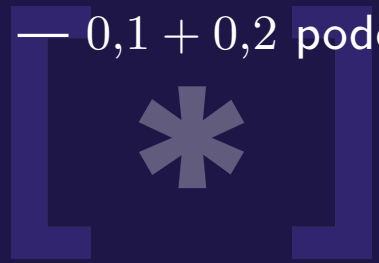
```
1. CREATE DATABASE  ->  criar o banco        | DDL
2. USE              ->  selecionar o banco   | feito UMA vez
3. CREATE TABLE     ->  criar a tabela       |
                    v
4. INSERT           ->  inserir dados        | DML/DQL
5. SELECT           ->  consultar            | o tempo todo
6. UPDATE           ->  alterar              |
7. DELETE           ->  remover              |
```

Sempre existe estrutura antes do dado.

[Tipos de dados]

Tipo	Guarda	Use para
`INT`	Inteiro	IDs, contadores
`TINYINT`	Inteiro pequeno	Período, idade
`VARCHAR(n)`	Texto até *n*	Nome, e-mail
`TEXT`	Texto longo	Descrições
`DATE`	`AAAA-MM-DD`	Data de ingresso
`DATETIME`	Data e hora	Agendamentos
`TIMESTAMP`	Data e hora com fuso	`criado_em`
`BOOLEAN`	Verdadeiro ou falso	Sinalizadores
`DECIMAL(p,s)`	Número exato	**Dinheiro**
`FLOAT`	Número aproximado	Nunca dinheiro

Regra de ouro: dinheiro em `DECIMAL`. `FLOAT` é aproximado — $0{,}1 + 0{,}2$ pode não dar $0{,}3$.



[Mão na massa #2: as duas tabelas]

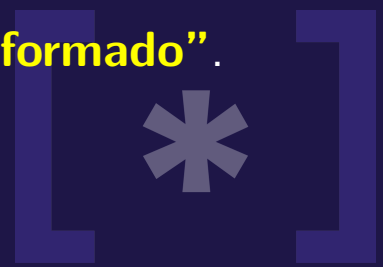
```sql
CREATE TABLE curso (
  id   INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(80) NOT NULL UNIQUE
) ENGINE=InnoDB;
```

```sql
CREATE TABLE membro (
  id             INT          AUTO_INCREMENT PRIMARY KEY,
  nome           VARCHAR(120) NOT NULL,
  email          VARCHAR(160) NOT NULL UNIQUE,
  periodo        TINYINT      NOT NULL,
  data_ingresso  DATE         NOT NULL,
  ativo          BOOLEAN      NOT NULL DEFAULT TRUE,
  curso_id       INT          NULL
) ENGINE=InnoDB;
```

[As restrições são a qualidade do dado]

`PRIMARY KEY`	Identifica cada linha. Não repete, não aceita `NULL`. **Uma por tabela**
`AUTO_INCREMENT`	O MySQL gera o número sozinho
`NOT NULL`	Campo obrigatório
`UNIQUE`	Não pode repetir. Pode haver várias por tabela
`DEFAULT x`	Valor assumido se o `INSERT` não informar
`FOREIGN KEY`	Liga esta tabela a outra (bloco 6)

NULL não é zero e não é texto vazio — é **"não informado"**.



[Conferindo o que você criou]

```
SHOW TABLES;
DESCRIBE membro;
SHOW CREATE TABLE membro;
```

Confira que as duas tabelas apareceram
antes de irmos para a pausa.

Pausa

5 minutos

Deixe o phpMyAdmin aberto.

BLOCO 4

Inserindo e consultando

01:00 · 25 min

[CRUD: as quatro operações]

Create	`INSERT`	DML
Read	`SELECT`	DQL
Update	`UPDATE`	DML
Delete	`DELETE`	DML

Todo sistema que você escrever na vida faz essas quatro coisas.

[Mão na massa #3: povoando as tabelas]

```sql
INSERT INTO curso (nome) VALUES
  ('Engenharia de Software'), ('Ciencia da Computacao'),
  ('Sistemas de Informacao'), ('Engenharia de Producao');
```

```sql
INSERT INTO membro (nome, email, periodo, data_ingresso,
    curso_id)
VALUES
  ('Ana Souza',   'ana@liga.edu.br',   7, '2024-03-11', 1),
  ('Bruno Lima',  'bruno@liga.edu.br', 3, '2025-02-20', 2),
  ('Carla Dias',  'carla@liga.edu.br', 5, '2024-08-05', 1),
  ('Diego Rocha', 'diego@liga.edu.br', 9, '2023-09-14', 3),
  ('Elisa Prado', 'elisa@liga.edu.br', 1, '2026-03-02', NULL);
```

Repare: `id` e `ativo` **não** foram informados.

[SELECT: a estrutura completa]

```
SELECT    colunas
FROM      tabela
WHERE     condicao              -- filtra LINHAS
ORDER BY  coluna [ASC|DESC]     -- ordena
LIMIT     n;                    -- limita a quantidade
```

```
SELECT * FROM membro;
SELECT nome, email FROM membro;
SELECT nome AS aluno, periodo AS sem FROM membro;
```

[WHERE: filtrando linhas]

```sql
SELECT * FROM membro WHERE periodo > 5;
SELECT * FROM membro WHERE ativo = TRUE AND periodo >= 5;
SELECT * FROM membro WHERE periodo IN (1, 3, 5);
SELECT * FROM membro WHERE periodo BETWEEN 3 AND 7;
SELECT * FROM membro WHERE nome LIKE 'A%';
SELECT * FROM membro WHERE email LIKE '%@liga.edu.br';
SELECT * FROM membro WHERE curso_id IS NULL;
```

`WHERE curso_id = NULL` nunca funciona.

`NULL` não é comparável com `=`. Use `IS NULL`.

[Ordenando e limitando]

```sql
SELECT nome, periodo FROM membro ORDER BY periodo DESC;

SELECT nome, periodo FROM membro
ORDER BY periodo DESC, nome ASC;

-- os 3 membros mais antigos da liga
SELECT nome FROM membro ORDER BY data_ingresso ASC LIMIT 3;
```

`ASC` = crescente (padrão) `DESC` = decrescente

[Exercício relâmpago]

3 minutos. Escreva o `SELECT` para:

1. Todos os membros **ativos**, mostrando só nome e e-mail
2. Membros do **1º ao 5º** período, do mais novo para o mais antigo
3. Quantos ingressaram **a partir de 2025**

Dica do 3: `SELECT COUNT(*) FROM ...`

BLOCO 5

Alterando e apagando

01:25 · 10 min

[UPDATE e DELETE]

```sql
UPDATE membro
SET    periodo = 8
WHERE  id = 1;
```

```sql
-- crie um registro descartavel para poder apagar com
    seguranca
INSERT INTO membro (nome, email, periodo, data_ingresso)
VALUES ('Registro Teste', 'teste@liga.edu.br', 1, '2026-01-01'
    );

DELETE FROM membro WHERE email = 'teste@liga.edu.br';
```

[O erro mais caro da carreira de vocês]

```sql
UPDATE membro SET periodo = 1;    -- zera TODOS os membros
DELETE FROM membro;               -- apaga TODOS os membros
```

Sem WHERE, vale para a tabela inteira.

E não existe Ctrl+Z em SQL.

```sql
-- 1) veja EXATAMENTE o que sera afetado
SELECT * FROM membro WHERE id = 5;

-- 2) so entao troque SELECT * por DELETE, com o MESMO where
DELETE FROM membro WHERE id = 5;
```

BLOCO 6

Ligando tabelas

01:35 · 20 min

[Até agora isso é uma planilha]

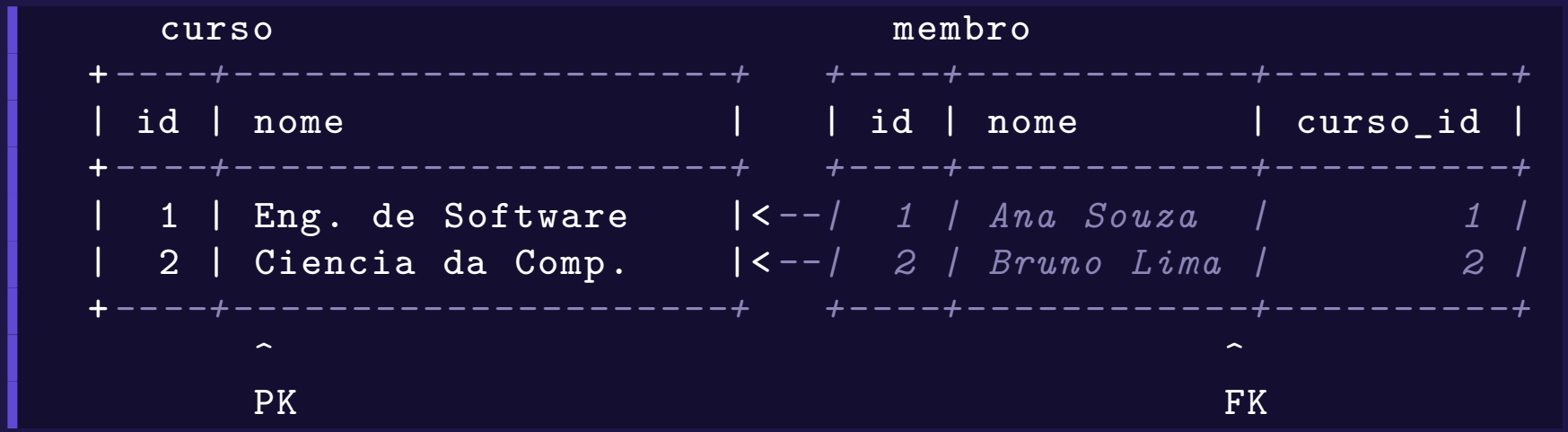
E se guardássemos o nome do curso dentro de `membro`?

- "Engenharia de Software" repetido em dezenas de linhas
- Alguém digita "Eng. de Software"
- Outro digita "engenharia de software"
- **A consulta por curso quebra**
- Corrigir o nome exige alterar todas as linhas

Solução: o curso vira uma tabela própria.

E `membro` guarda apenas o `id` do curso.

[Chave primária e chave estrangeira]



- **PK** identifica a linha **dentro** da própria tabela
- **FK** **aponta** para a PK de outra tabela
- O SGBD passa a **garantir** que aquele valor existe do outro lado

[Mão na massa #4: declarando a FK]

```
ALTER TABLE membro
  ADD CONSTRAINT fk_membro_curso
  FOREIGN KEY (curso_id) REFERENCES curso(id)
  ON DELETE SET NULL;
```

Agora tente inserir um curso que não existe:

```
INSERT INTO membro (nome, email, periodo, data_ingresso,
    curso_id)
VALUES ('Fake', 'fake@liga.edu.br', 1, '2026-01-01', 999);
```

ERROR 1452: a foreign key constraint fails

O erro é o **sucesso**: o banco se recusou a guardar lixo.

[JOIN: as duas tabelas juntas]

```sql
-- INNER JOIN: so os membros QUE TEM curso
SELECT m.nome AS membro, c.nome AS curso, m.periodo
FROM   membro m
INNER  JOIN curso c ON m.curso_id = c.id
ORDER  BY c.nome;
```

```sql
-- LEFT JOIN: TODOS os membros. Elisa aparece com curso NULL.
SELECT m.nome AS membro, c.nome AS curso
FROM   membro m
LEFT   JOIN curso c ON m.curso_id = c.id;
```

A condição do `ON` é sempre **FK = PK**.

[Bônus: contar e agrupar]

```sql
SELECT COUNT(*) FROM membro;
SELECT AVG(periodo) FROM membro;
```

```sql
-- quantos membros por curso
SELECT c.nome AS curso, COUNT(m.id) AS total
FROM   curso c
LEFT   JOIN membro m ON m.curso_id = c.id
GROUP  BY c.id, c.nome
ORDER  BY total DESC;
```

`WHERE` filtra **linhas antes** de agrupar · `HAVING` filtra **grupos depois**

BLOCO 7

Fechando

01:55 · 5 min

[Os erros que vão acontecer com você]

`ERROR 1046` No database selected	Faltou o `USE liga_SEUNOME`
`ERROR 1044` Access denied	Seu banco não começa com `liga_`
`ERROR 1050` Table already exists	Rodou o `CREATE TABLE` duas vezes
`ERROR 1062` Duplicate entry	Violou `PRIMARY KEY` ou `UNIQUE`
`ERROR 1452` FK constraint fails	Apontou para um `id` que não existe
`ERROR 1052` Column is ambiguous	Faltou qualificar: `m.nome`, `c.nome`
Acento virou caractere estranho	Banco criado sem `utf8mb4`
O comando não executa	Faltou o `;` no fim

[Para onde ir depois]

Estudar, nesta ordem

1. Normalização (1FN, 2FN, 3FN)
2. Modelagem ER
3. Relação N:N
4. `GROUP BY` e subconsultas
5. Índices e `EXPLAIN`
6. Transações e ACID

Praticar de graça

- SQLBolt
- SQLZoo
- DB Fiddle
- HackerRank SQL

O guia completo e o script da aula
estão no repositório da capacitação.

for_code[ ]

Obrigado!

Capacitação de MySQL

Ronivaldo D. Andrade

27 de agosto de 2026