Banco de Dados 2

Aula 11

MSc Sandir Rodrigues Campos

Exercícios - Popule com o mínimo de 5 tuplas cada tabela

Exercício 1 — Subconsulta Correlacionada com EXISTS Enunciado: Liste os nomes dos clientes que compraram pelo menos 3 produtos diferentes em algum pedido.

Exercício 2 — Subconsulta com IN + Agregação Enunciado: Liste os nomes dos clientes cujo valor total de pedidos é superior ao valor total de pedidos dos clientes da cidade 'São Paulo'.

Exercício 3 — Subconsulta no FROM (Tabela Derivada) Enunciado:Crie uma subconsulta que calcula o total de vendas por mês. A consulta principal deve retornar somente os meses onde o total de vendas foi superior a R$ 10.000.

Exercício 4 — Subconsulta no UPDATE Enunciado:Atualize os preços dos produtos aumentando em 5% apenas aqueles cujo preço esteja abaixo da média dos preços de todos os produtos.

Exercício 5 — Subconsulta com NOT IN Enunciado:Liste os nomes dos produtos que nunca foram vendidos em nenhum pedido.

Exercício 6 — Subconsulta no SELECT com cálculo Enunciado:Liste os pedidos (ID e valor) junto com a média de valor dos pedidos feitos pelo mesmo cliente. (Dica: subconsulta correlacionada no SELECT)

ÍNDICES

Índice de um livro

- Imagine que você tem um **livro com 1.000 páginas**. Se quiser encontrar o capítulo sobre “História da Computação”, você **não vai ler página por página** — vai ao **índice** no início do livro e descobre que está na página 732.
- No banco de dados é igual: um **índice** permite **ir direto** ao registro certo, sem "ler" linha por linha da tabela.

POR QUE USAMOS ÍNDICES?

Sem índice:

- SELECT * FROM alunos WHERE nome = 'João';
- O MySQL **lê linha por linha** até achar “João” (muito lento se tiver 1 milhão de alunos).

Com índice:

- MySQL já sabe onde está o “João” — é **rápido como um foguete**

COMO CRIAR UM ÍNDICE

- Exemplo: Tabela de alunos

```
CREATE TABLE alunos (
    id INT PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100)
);
```

- Agora vamos criar um índice na coluna nome:

```
CREATE INDEX idx_nome ON alunos(nome);
```

- Agora buscas por nome ficam muito mais rápidas.

TIPOS DE ÍNDICES

Tipo	Pra que serve	Exemplo
PRIMARY KEY	Identificador único da tabela	id INT PRIMARY KEY
UNIQUE	Garante valores únicos	email UNIQUE
INDEX (ou KEY)	Índice comum para acelerar buscas	CREATE INDEX idx_nome ON alunos(nome);
FULLTEXT	Busca por palavras em textos grandes	CREATE FULLTEXT INDEX texto_idx ON artigos(texto);

TESTE PRÁTICO

- CREATE TABLE produtos (
- id INT PRIMARY KEY,
- nome VARCHAR(100),
- preco DECIMAL(10,2),
- estoque INT
-);

- CREATE INDEX idx_nome ON produtos(nome);

- SELECT * FROM produtos WHERE nome = 'Notebook';

COMANDOS

VERIFICAR ÍNDICES EXISTENTES

- SHOW INDEX FROM produtos;

REMOVER UM ÍNDICE

- DROP INDEX idx_nome ON produtos;

DICAS

- Use índice quando:
  - A coluna aparece muito em WHERE
  - A coluna aparece em JOIN, ORDER BY, ou GROUP BY
- Evite:
  - Colocar índice em todas as colunas (deixa INSERT/UPDATE mais lentos)
  - Índices em colunas com valores repetidos demais (como "sexo", "ativo", "sim/não")

ÍNDICE COMPOSTO (mais de uma coluna)

CREATE INDEX idx_nome_email ON alunos(nome, email);

- Ajuda em buscas como:

*SELECT * FROM alunos WHERE nome = 'João' AND email = 'joao@email.com';*

- Mas não ajuda se buscar só por email — precisa começar pela primeira coluna (nome).

COMO SABER SE UM ÍNDICE ESTÁ SENDO USADO?

*EXPLAIN SELECT * FROM alunos WHERE nome = 'João';*

O MySQL vai mostrar se o índice foi utilizado na consulta.

EXERCÍCIO

- Criar uma tabela clientes
- Inserir 10 registros
- Criar índice em nome
- Fazer busca com e sem índice
- Usar EXPLAIN para verificar diferença

EXERCÍCIOS

- QUESTÃO 1 — TABELA: clientes - Objetivo: Melhorar buscas por nome de cliente
- QUESTÃO 2 — TABELA: vendas - Objetivo: Acelerar relatórios por data da venda
- QUESTÃO 3 — TABELA: usuários - Objetivo: Garantir que e-mails não se repitam

Exercícios

- Cenário Base (use como referência para todos os exercícios):
  - Tabelas:
    - clientes(id_cliente, nome, cidade)
    - pedidos(id_pedido, id_cliente, data_pedido, valor_total)
    - produtos(id_produto, nome_produto, preco)
    - itens_pedido(id_pedido, id_produto, quantidade)

    - Popule com o mínimo de 10 tuplas cada tabela

O que são funções em MySQL?

Funções são blocos de código reutilizáveis que recebem parâmetros, executam instruções e retornam um resultado. Elas são usadas para evitar repetição de código e organizar lógicas complexas.

Tipos de Funções no MySQL

Funções internas (built-in functions)

Funções definidas pelo usuário (UDF - User Defined Functions)

1. Funções internas (built-in functions)

Já vêm prontas no MySQL. Exemplos:

- String: CONCAT(), UPPER(), LOWER(), SUBSTRING()
- Numéricas: ROUND(), FLOOR(), CEIL(), ABS()
- Data e hora: NOW(), CURDATE(), DATE_ADD(), DATEDIFF()
- Lógicas: IF(), IFNULL(), CASE

1. Funções internas (built-in functions)

- CONCAT()
  - Une (concatena) duas ou mais strings.
    - *SELECT CONCAT('Olá, ', 'Thor!');*
    - *-- Resultado: Olá, Thor!*
- UPPER()
  - Transforma a string em **maiúsculas**.
    - *SELECT UPPER('olá');*
    - *-- Resultado: OLÁ*
- LOWER()
  - Transforma a string em minúsculas.
    - SELECT LOWER('THOR É LINDO');
    - -- Resultado: thor é lindo
- SUBSTRING()
  - Extrai parte de uma string.
    - *SELECT SUBSTRING('Thor é bonito', 1, 4);*
    - *-- Resultado: Thor*
- ROUND()
  - Arredonda um número.
    - *SELECT ROUND(3.14159, 2);*
    - *-- Resultado: 3.14*
- FLOOR()
  - Arredonda para baixo..
    - *SELECT FLOOR(7.9);*
    - *-- Resultado: 7*
- CEIL() ou CEILING()
  - Arredonda para cima.
    - SELECT CEIL(4.2);
    - -- Resultado: 5
- ABS()
  - Retorna o valor absoluto (sem sinal).
    - *SELECT ABS(-10);*
    - *-- Resultado: 10*

1. Funções internas (built-in functions)

- NOW()
  - Data e hora atual.
    - *SELECT NOW();*
    - *-- Resultado: 2025-05-07 15:30:45 (exemplo)*
- CURDATE()
  - Só a data atual (sem hora).
    - *SELECT CURDATE();*
    - *-- Resultado: 2025-05-07*
- DATE_ADD()
  - Adiciona tempo (dias, meses, anos...) a uma data.
    - SELECT DATE_ADD('2025-05-07', INTERVAL 10 DAY);
    - -- Resultado: 2025-05-17
- DATEDIFF()
  - Calcula a diferença em dias entre duas datas.
    - *SELECT DATEDIFF('2025-05-10', '2025-05-01');*
    - *-- Resultado: 9*

1. Funções internas (built-in functions)

- IF()
  - Retorna um valor se a condição for verdadeira, outro se for falsa.
    - *SELECT IF(10 > 5, 'Maior', 'Menor');*
    - *-- Resultado: Maior*
- IFNULL()
  - Retorna um valor alternativo se o original for NULL.
    - *SELECT IFNULL(NULL, 'Sem valor');*
    - *-- Resultado: Sem valor*
- CASE
  - Tipo de estrutura condicional (como "se... então...").
    - SELECT
    - CASE
    - WHEN 10 > 20 THEN 'Maior que 20'
    - WHEN 10 = 10 THEN 'Igual a 10'
    - ELSE 'Outro'
    - END;
    - -- Resultado: Igual a 10
    - SELECT DATE_ADD('2025-05-07', INTERVAL 10 DAY);
    - -- Resultado: 2025-05-17

Funções internas

- Crie consultas com resultados que uma para cada tipo de função interna

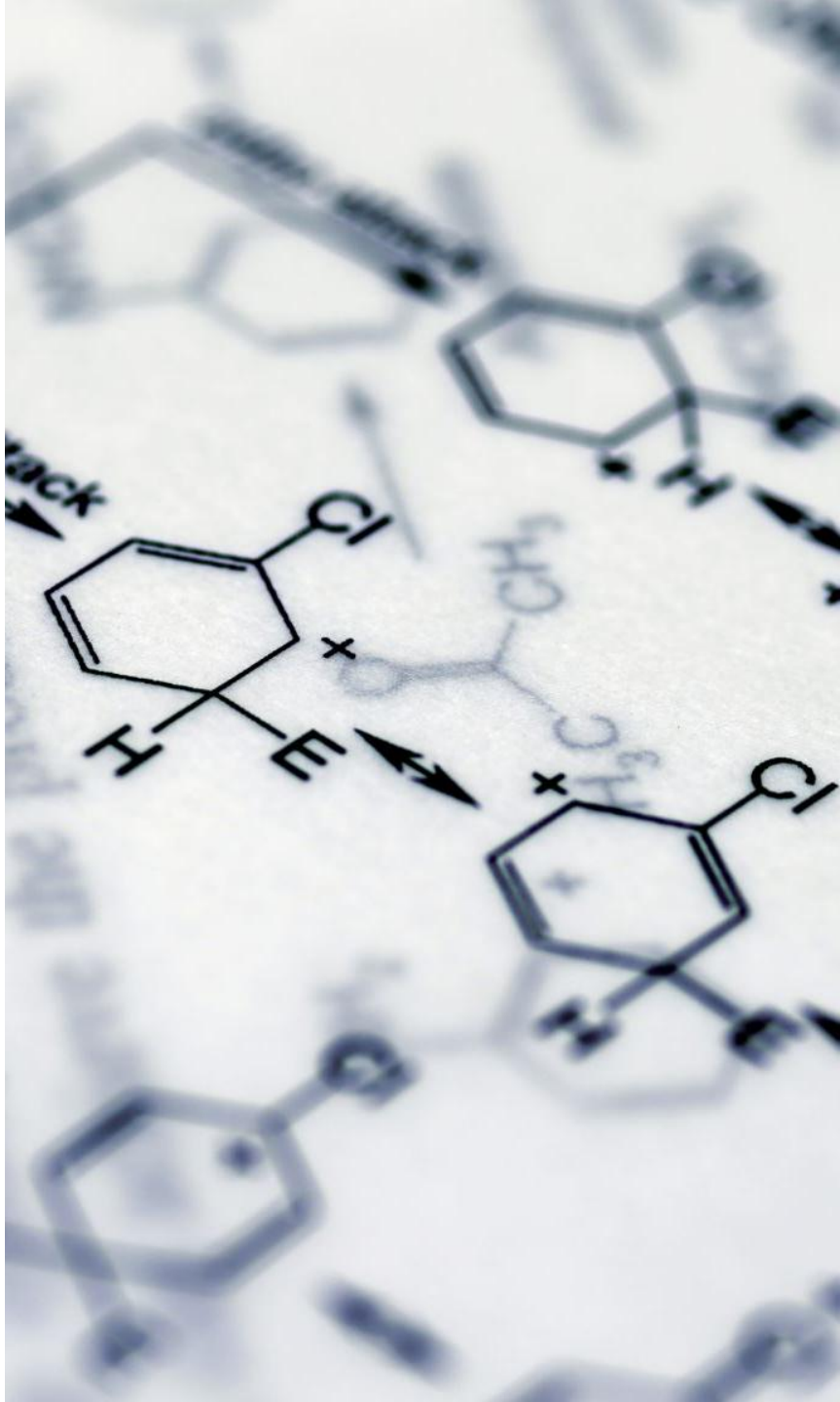
2. **Funções definidas pelo usuário (UDF - User Defined Functions)**

- **Criando Funções no MySQL - SINTAXE**
- DELIMITER //
- CREATE FUNCTION nome_da_funcao(param1 TIPO, param2 TIPO, ...)
- RETURNS TIPO_RETORNO
- DETERMINISTIC
- BEGIN
- -- declarações, lógica, retorno
- RETURN valor;
- END;
- //
- DELIMITER ;

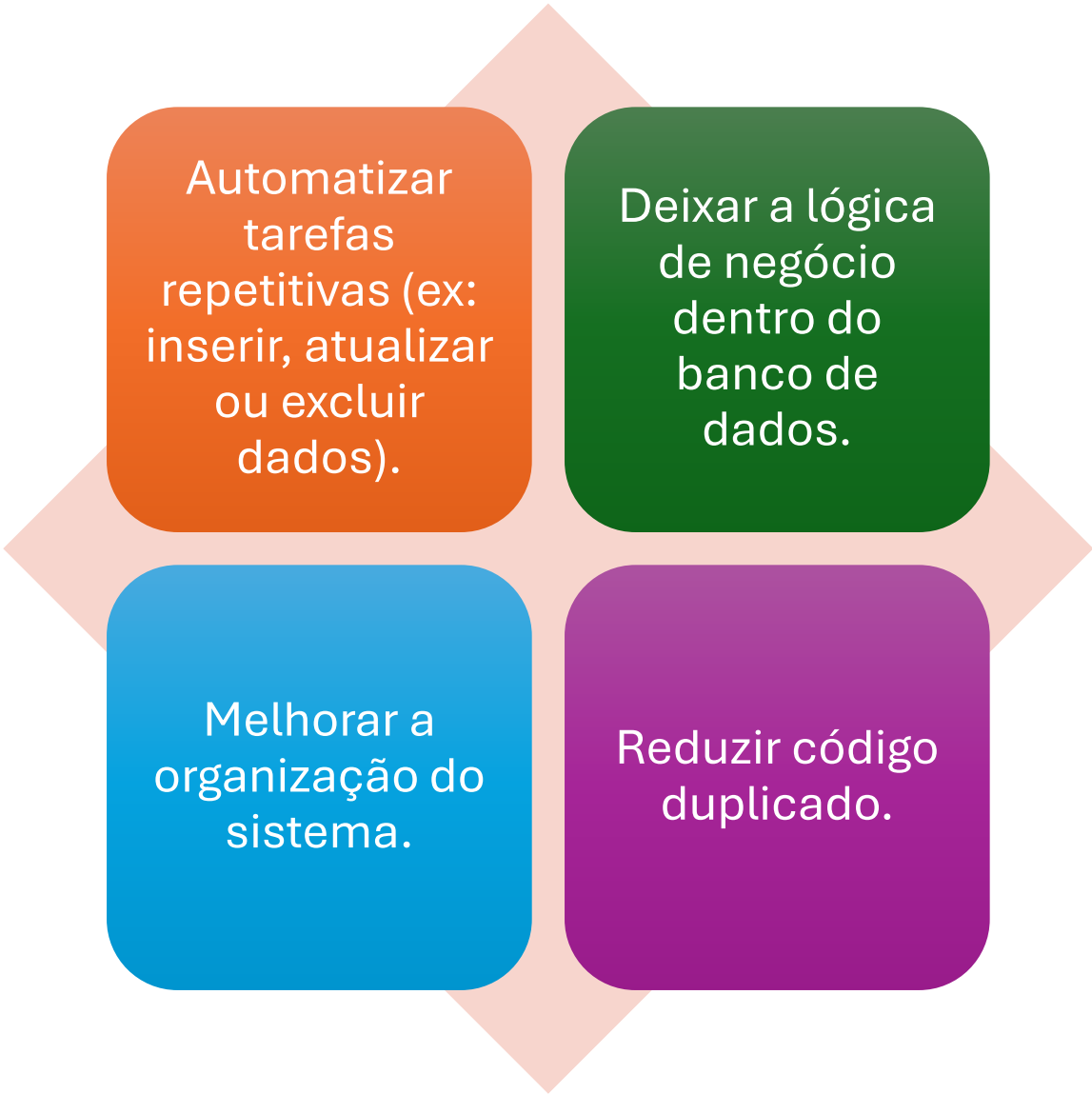
- **Características importantes:**
- **DETERMINISTIC:** sempre retorna o mesmo resultado para os mesmos parâmetros.
- **NOT DETERMINISTIC:** pode retornar resultados diferentes (ex: funções com NOW() ou RAND()).
- Você precisa de permissão (**CREATE ROUTINE**) para criar funções.
- Funções devem sempre retornar um valor.

O que é uma Procedure?

- Uma procedure (ou procedimento armazenado) é um bloco de código SQL salvo no banco de dados, que pode ser executado sempre que quiser, com ou sem parâmetros.
- É como uma receita que o MySQL guarda para repetir depois.



Para que serve?



Sintaxe básica

```
DELIMITER //

CREATE PROCEDURE nome_da_procedure(param1 TIPO, param2 TIPO)
BEGIN
  -- comandos SQL aqui
END;
//

DELIMITER ;
```

O DELIMITER // muda o final de comando temporariamente, porque o MySQL termina comandos com ; por padrão.

Como executar uma procedure?

- CALL nome_da_procedure(valor1, valor2);

- DELIMITER //

- CREATE PROCEDURE diga_ola()
- BEGIN
- SELECT 'Olá, Thor!';
- END;
- //

- DELIMITER ;

- -- Executando:
- CALL diga_ola();