Professores
Celso Rodrigo Giusti
Daniel Manoel Filho
Marlon P. F. Rodrigues

COMANDOS TCL E PROGRAMAÇÃO EM BANCO DE DADOS

SENAI

BANCO DE DADOS

O que é TCL (Transaction Control Language)?

TCL, ou Linguagem de Controle de Transação, gerencia como as alterações de DML (*INSERT*, *UPDATE*, *DELETE*) são salvas no banco.

Uma **Transação** é uma sequência de uma ou mais operações SQL tratadas como uma **unidade atômica**.

Isso garante o princípio "Tudo ou Nada": ou todas as operações da transação são executadas com sucesso, ou nenhuma delas é aplicada.

BANCO DE DADOS

COMMIT

O comando *COMMIT* (Confirmar) é usado para **salvar permanentemente** todas as alterações feitas na transação atual.

Uma vez que você executa *COMMIT*, as alterações (seus *INSERTs*, *UPDATEs*, etc.) tornam-se visíveis para todos os outros usuários e não podem mais ser desfeitas com *ROLLBACK*.

```sql
START TRANSACTION;
UPDATE tbl_membro SET telefone = '11-99999-0000' WHERE id_membro = 101;
COMMIT;
```

BANCO DE DADOS

ROLLBACK

O comando *ROLLBACK* (Reverter) é o "Ctrl+Z" do banco de dados. Ele **descarta** todas as alterações feitas desde o último *COMMIT* ou *START TRANSACTION*.
É usado quando algo dá errado no meio de uma transação ou se o usuário simplesmente muda de ideia *antes* de salvar.

```
START TRANSACTION;

INSERT INTO tbl_membro (id_membro, nome_membro, ...)
VALUES (999, 'Membro Teste', ...);

/* O 'Membro Teste' existe aqui dentro da transação */

ROLLBACK;

/* O 'Membro Teste' foi desfeito e nunca existiu */
```

BANCO DE DADOS

SAVEPOINT

O *SAVEPOINT* é um "marcador" ou "ponto de verificação" *dentro* de uma transação longa.
Ele permite que você execute um *ROLLBACK* parcial, voltando apenas até aquele marcador, sem descartar a transação inteira.

```
START TRANSACTION;
  INSERT ...; /* Operação 1 */

  SAVEPOINT ponto_A;

  UPDATE ...; /* Operação 2 */

  /* Ops, a Op. 2 deu errado. Não quero
     desfazer a Op. 1, apenas a 2. */
  ROLLBACK TO SAVEPOINT ponto_A;

COMMIT; /* Salva apenas a Operação 1 */
```

BANCO DE DADOS

Programação no Banco de Dados

Além de armazenar dados, podemos embutir lógica de programação *diretamente* no SGBD.

Isso é feito usando objetos como:

VIEW (Visões)
STORED PROCEDURE (Procedimentos Armazenados)
FUNCTION (Funções)
TRIGGER (Gatilhos)
EVENT (Eventos)

Vantagens: Performance (reduz tráfego de rede), Segurança (encapsula regras de negócio) e Reutilização.

BANCO DE DADOS

VIEW

Uma *VIEW* é uma **tabela virtual**. Ela não armazena dados fisicamente, mas sim a consulta *SELECT* que a define.

É uma "consulta salva" que pode ser tratada como se fosse uma tabela.

Utilidade:
Simplificar: Esconde *JOINs* complexos.
Segurança: Permite que usuários vejam apenas colunas específicas de uma tabela.

BANCO DE DADOS

VIEW

Uma *VIEW* ... a consulta S...

É uma "co...

Utilidade
Simplifica...
Segurança...bela.

```sql
CREATE VIEW V_Relatorio_Emprestimos AS
SELECT
    M.nome_membro,
    L.titulo_livro,
    E.data_emprestimo,
    E.data_devolucao
FROM tbl_membro M
JOIN tbl_emprestimo E ON M.id_membro = E.id_membro
JOIN tbl_exemplar EX ON E.id_exemplar = EX.id_exemplar
JOIN tbl_livro L ON EX.isbn = L.isbn;
```

```sql
/* Em vez de escrever todo o JOIN,
   agora basta fazer: */
SELECT * FROM V_Relatorio_Emprestimos
WHERE nome_membro = 'Ana Silva';
```

BANCO DE DADOS

STORED PROCEDURE

Uma *PROCEDURE* é um conjunto de comandos SQL nomeado e armazenado no banco, que pode ser executado com um simples *CALL*.

- Pode receber parâmetros de entrada (*IN*) e saída (*OUT*).
- Pode conter lógica (*IF*, *CASE*, *LOOP*).
- Pode executar DML (*INSERT*, *UPDATE*, *DELETE*).

BANCO DE DADOS

STORE

SENAI

Uma *PROCEDU* ...nado no banco, que pode ser e...

- Pode recebe...
- Pode conter...
- Pode execut...

```
/* DELIMITER é um comando do cliente MySQL
   para mudar o "fim" do comando */
DELIMITER $$

CREATE PROCEDURE sp_novo_emprestimo (
    IN p_id_exemplar INT,
    IN p_id_membro INT
)
BEGIN
    INSERT INTO tbl_emprestimo (
        data_emprestimo,
        data_devolucao,
        data_devolucao_efetiva,
        id_exemplar,
        id_membro
    )
    VALUES (
        CURDATE(),
        CURDATE() + INTERVAL 14 DAY, /* Prazo de 14 dias */
        NULL,
        p_id_exemplar,
        p_id_membro
    );
END$$

DELIMITER ;
```

Como usar: `CALL sp_novo_emprestimo(101, 101);`

BANCO DE DADOS

FUNCTION

Uma *FUNCTION* é um bloco de código que **obrigatoriamente retorna um valor único**.

Ela é projetada para ser usada *dentro* de outras consultas SQL (como *SELECT* ou *WHERE*), similar às funções nativas (*UCASE(), NOW()*).

Regra: Funções, por padrão, não podem executar DML (*INSERT*, *UPDATE*). Elas servem para **cálculos**.

```
DELIMITER $$

CREATE FUNCTION fn_status_membro (p_id_membro INT)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    DECLARE v_atrasos INT;

    SELECT COUNT(*) INTO v_atrasos
    FROM tbl_emprestimo
    WHERE id_membro = p_id_membro
      AND data_devolucao < CURDATE()
      AND data_devolucao_efetiva IS NULL;

    IF v_atrasos > 0 THEN
        RETURN 'Com Atraso';
    ELSE
        RETURN 'Regular';
    END IF;
END$$

DELIMITER ;
```

Como usar: `SELECT nome_membro, fn_status_membro(id_membro) FROM tbl_membro;`

nte retorna um valor único.

SQL (como *SELECT* ou

SERT, *UPDATE*). Elas servem

BANCO DE DADOS

TRIGGER

Um *TRIGGER* é um tipo especial de procedimento que é executado **automaticamente** quando um evento DML (*INSERT*, *UPDATE* ou *DELETE*) ocorre em uma tabela específica.

Você não pode "chamar" um trigger. Ele "dispara" sozinho.

Usos comuns:

- *BEFORE INSERT*: Validar dados antes de inserir.
- *AFTER INSERT/UPDATE/DELETE*: Criar logs de auditoria.

BANCO DE DADOS

TRIGGER

Vamos criar um log de auditoria. Queremos saber quem e quando alterou o título de um livro.

```
CREATE TABLE tbl_livro_log (
    id_log INT AUTO_INCREMENT PRIMARY KEY,
    isbn_livro VARCHAR(16),
    titulo_antigo VARCHAR(200),
    titulo_novo VARCHAR(200),
    data_mudanca DATETIME
);
```

BANCO DE DADOS

TRIGGER

Vamos criar um log de auditoria. Queremos saber quem e quando alterou o título de um livro.

```
DELIMITER $$

CREATE TRIGGER trg_log_mudanca_livro
AFTER UPDATE ON tbl_livro
FOR EACH ROW
BEGIN
    IF OLD.titulo_livro != NEW.titulo_livro THEN
        INSERT INTO tbl_livro_log (isbn_livro, titulo_antigo, titulo_novo, data_mudanca)
        VALUES (OLD.isbn, OLD.titulo_livro, NEW.titulo_livro, NOW());
    END IF;
END$$

DELIMITER ;
```

BANCO DE DADOS

TRIGGER

Testando o TRIGGER.

```
UPDATE tbl_livro
SET titulo_livro = 'Duna (Edição Especial)'
WHERE isbn = '978-85-390-0064-8';
```

```
SELECT * FROM tbl_livro_log;
```

BANCO DE DADOS

EVENT

Um *EVENT* (ou "Scheduled Event") é uma tarefa que o SGBD executa automaticamente em um **agendamento (schedule)**.

Diferente de Triggers (que disparam com DML), Events disparam com o **tempo**.

É o "Cron" (Linux) ou "Agendador de Tarefas" (Windows) *dentro* do banco.

Atenção: O agendador de eventos do MySQL pode vir desligado. (Comando para ligar: *SET GLOBAL event_scheduler = ON;*)

BANCO DE DADOS

EVENT

Vamos criar um evento que, toda noite, verifica os exemplares emprestados que estão atrasados e muda seu status para 'Atrasado'.

```
CREATE EVENT evt_verifica_atrasados
ON SCHEDULE EVERY 1 DAY
STARTS '2025-01-01 02:00:00' /* Começa amanhã, às 2h da manhã */
DO
BEGIN
    UPDATE tbl_exemplar EX
    JOIN tbl_emprestimo E ON EX.id_exemplar = E.id_exemplar
    SET EX.status_exemplar = 'Atrasado'
    WHERE E.data_devolucao < CURDATE()
      AND E.data_devolucao_efetiva IS NULL;
END$$
```

BANCO DE DADOS

RESUMO DA AULA

SENAI

✅ **TCL (Transação)** = Linguagem para COMMIT (salvar) ou ROLLBACK (descartar) alterações de DML, garantindo "tudo ou nada".
✅ **COMMIT** = Salva permanentemente as alterações da transação.
✅ **ROLLBACK** = Descarta as alterações da transação (antes do COMMIT).
✅ **SAVEPOINT** = Cria um "ponto de verificação" para ROLLBACKs parciais.
✅ **VIEW** = Tabela virtual baseada em um SELECT (simplifica consultas).
✅ **STORED PROCEDURE** = Bloco de código (com INSERT/UPDATE) chamado com CALL (ex: sp_novo_emprestimo()).
✅ **FUNCTION** = Bloco de código que *retorna um valor único* e é usado em SELECT (ex: fn_status_membro()).
✅ **TRIGGER** = Ação *automática* disparada por um INSERT, UPDATE ou DELETE em uma tabela (ex: auditoria).
✅ **EVENT** = Ação *automática* disparada por *agendamento de tempo* (ex: "rodar todo dia às 2h").

BANCO DE DADOS

EXERCÍCIOS

Exercício 1: Praticando TCL (COMMIT / ROLLBACK)

1. Inicie uma transação com START TRANSACTION;.
2. Insira um novo membro fictício na tbl_membro com ID 999.
3. Execute SELECT * FROM tbl_membro WHERE id_membro = 999; (Você deve ver o membro).
4. Execute ROLLBACK;.
5. Execute SELECT * FROM tbl_membro WHERE id_membro = 999; (O membro deve ter sumido).
6. *Desafio:* Repita o processo, mas use COMMIT e verifique se o dado persistiu. (Lembre-se de deletar o membro 999 depois).

BANCO DE DADOS

EXERCÍCIOS

Exercício 2: Criando uma VIEW

1. Crie uma VIEW chamada V_Livros_Autores que contenha o titulo_livro, ano_publicacao e nome_autor (usando os JOINs da Aula 06).
2. Após criar, execute SELECT * FROM V_Livros_Autores WHERE nome_autor LIKE 'Machado%’;.

Exercício 3: Criando uma STORED PROCEDURE

1. Crie uma PROCEDURE chamada sp_cadastrar_livro que receba os 4 parâmetros da tbl_livro (isbn, titulo, ano, editora) e execute o INSERT.
2. Teste-a usando: CALL sp_cadastrar_livro('978-85-390-0064-8', 'Duna', 1965, 'Aleph');.

BANCO DE DADOS

EXERCÍCIOS

Exercício 4: Criando uma FUNCTION

1. Crie uma FUNCTION chamada fn_get_titulo_livro que recebe um isbn e retorna o titulo_livro (VARCHAR 200).
2. Teste-a usando: SELECT fn_get_titulo_livro('978-85-390-0064-8');.
3. *Desafio:* Use esta função em um SELECT na tbl_exemplar para mostrar o *título* ao lado do *isbn*: SELECT id_exemplar, fn_get_titulo_livro(isbn) FROM tbl_exemplar;.

BANCO DE DADOS

EXERCÍCIOS

SENAI

Exercício 5: Criando um TRIGGER (Auditoria)

1. Crie uma nova tabela chamada tbl_emprestimo_log com as colunas (id_log INT PK AUTO_INCREMENT, acao VARCHAR(10), id_emprestimo_afetado INT, data_hora DATETIME).
2. Crie um TRIGGER chamado trg_log_novo_emprestimo que dispara AFTER INSERT ON tbl_emprestimo.
3. Dentro do trigger, insira um registro na tbl_emprestimo_log, informando a acao ('NOVO'), o id_emprestimo (usando NEW.id_emprestimo) e a data (NOW()).
4. Teste o trigger chamando a procedure do Ex. 3 (sp_novo_emprestimo) ou fazendo um INSERT manual, e depois dê SELECT * FROM tbl_emprestimo_log;.

BANCO DE DADOS

Referências

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

MACHADO, Felipe N. R. **Banco de dados**: projeto e implementação. 4. ed. São Paulo: Erica, 2014.

MYSQL. **MySQL 8.0 Reference Manual**. Sunnyvale, CA: Oracle Corporation, 2023. Disponível em: https://dev.mysql.com/doc/refman/8.0/en/. Acesso em: 30 out. 2025.

SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL (SENAI). Departamento Regional de São Paulo. **PLANO DE CURSO**: Técnico em Desenvolvimento de Sistemas. São Paulo: SENAI-SP, 2023.

Escola SENAI "Italo Bologna"

Av. Goiás, 139 – Itu/SP

Telefone
(11) 2396-1999

Instagram
@senai.itu

Facebook
/senai.itu

Site
https://sp.senai.br/unidade/itu/