

Nome: Diogo Campos
Nome: Gabriel Augusto

Commit = O comando **COMMIT** é usado para confirmar e salvar permanentemente todas as alterações feitas e uma transação. ele finaliza a transação, tornando todas as modificações disponíveis para outros usuários e sessões.

Quando Usar o COMMIT

Você usa **COMMIT** quando está realizando várias operações que **devem ser concluídas com sucesso em conjunto**. Se uma delas falhar, você pode usar **ROLLBACK** para desfazer todas.

exemplos de COMMIT

Inserção de dados

Você insere dados em duas tabelas relacionadas e só quer confirmar se ambas as inserções forem bem-sucedidas.

Atualização em massa

Você atualiza preços de produtos de uma categoria específica, mas só quer confirmar após revisar.

Exclusão de dados com proteção contra perda acidental

Você quer limpar registros antigos, mas só se tiver certeza de que os filtros estão corretos.

```

109     INSERT INTO clientes(nome, email)
110     VALUES (nome_cliente, email_cliente);
111 END //
112 DELIMITER ;
113
114 -- Inserir um cliente de exemplo
115 CALL inserir_cliente('Diogo Campos', 'diogo@email.com');
116 • COMMIT;
117 • SELECT * FROM clientes;
118
119 -- Procedure para calcular total de vendas por cliente
120 DELIMITER //
121 • CREATE PROCEDURE total_vendas_cliente(
122     IN cliente INT,
123     OUT total DECIMAL(10,2)
124 )
125 BEGIN
126     SELECT SUM(valor) INTO total
127     FROM vendas

```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id	nome	email	ativo
1	Diogo Campos	diogo@email.com		1
2	Cliente sem vendas	semvendas@teste.com		1
3	Cliente com vendas	comvendas@teste.com		1
4	Cliente1	cliente1@teste.com		1
5	Cliente2	cliente2@teste.com		1
6	Cliente3	cliente3@teste.com		1
7	Cliente4	cliente4@teste.com		1

Result 15

Result 16

Result 17

produtos 18

EVENTS 19

clientes 20

Result 21

vendas 22

Result 23

Output

Action Output

#	Time	Action
115	16:21:28	SELECT @total LIMIT 0, 1000
116	16:21:28	CREATE PROCEDURE inserir_clientes_automatico() BEGIN DECLARE i INT DEFAULT 1; WHILE i <= 100 DO CALL
117	16:21:28	-- Executar inserção em massa CALL inserir_clientes_automatico()
118	16:21:28	COMMIT

Rollback = O comando **ROLLBACK** em SQL serve para anular operações que ainda não foram aprovadas no banco de dados. Ele ajuda o banco de dados a voltar ao último estado confirmado.

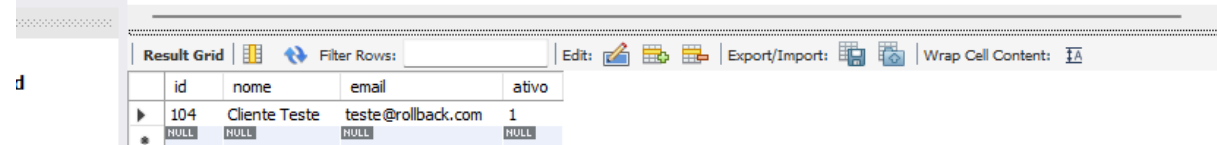
Exemplos de uso do **ROLLBACK**

Erro ao inserir dados - Se você tentou inserir vários registros, mas um deles tem erro, você pode fazer **ROLLBACK** para não salvar nenhum registro parcialmente inserido.

Validação de dados falhou - Durante uma atualização em várias tabelas, uma regra de negócio falha. Você faz **ROLLBACK** para não deixar dados inconsistentes.

Cancelamento manual pelo usuário - O usuário decide cancelar uma operação complexa antes de finalizar. O rollback desfaz todas as alterações feitas na transação.

```
228 -- Usando banco de dados 'loja' do seu exemplo
229 • USE loja;
230
231 -- Iniciar uma transação
232 • START TRANSACTION;
233
234 -- Tentar inserir um cliente
235 • INSERT INTO clientes(nome, email) VALUES ('Cliente Teste', 'teste@rollback.com');
236
237 -- Tentar inserir uma venda inválida (cliente_id 9999 pode não existir)
238 • INSERT INTO vendas(cliente_id, valor, data_venda) VALUES (9999, 150.00, '2025-06-01');
239
240 -- Verificar erro e decidir desfazer alterações
241 -- Como o cliente_id não existe, o segundo INSERT falhará (chave estrangeira)
242 -- Então executamos:
243 • ROLLBACK;
244
245 -- Verifique que nenhum dado foi inserido
246 • SELECT * FROM clientes WHERE email = 'teste@rollback.com';
247
```



Savepoint = O comando **SAVEPOINT** em SQL serve para marcar um momento dentro de uma transação. Assim, se for necessário, você pode voltar a esse ponto usando o comando **ROLLBACK TO SAVEPOINT**, sem precisar desfazer toda a transação.

Exemplo de **SAVEPOINT**

Inserções parciais com validação - Você insere vários registros, mas um deles falha. Em vez de perder tudo com **ROLLBACK**, você usa **SAVEPOINT** antes de cada inserção e volta apenas até o último ponto válido.

Atualizações em lote com cancelamento parcial - Durante uma atualização em massa, o sistema detecta um erro lógico após algumas etapas. Você usa **ROLLBACK TO SAVEPOINT** para desfazer só a parte problemática.

Etapas condicionais em scripts manuais - Ao executar um script SQL passo a passo manualmente, você pode definir **SAVEPOINTS** como "pontos de segurança", e desfazer até eles caso mude de ideia.

```
260 -- Inserir Cliente B
261 • INSERT INTO clientes(nome, email)
262   VALUES ('Cliente B', 'clienteB@email.com');
263
264 -- Criar SAVEPOINT após Cliente B
265 • SAVEPOINT depois_cliente_b;
266
267 -- Inserir Cliente C
268 • INSERT INTO clientes(nome, email)
269   VALUES ('Cliente C', 'clienteC@email.com');
270
271 -- ⚠ Decidimos que Cliente C não deve ser inserido
272 • ROLLBACK TO SAVEPOINT depois_cliente_b;
273
274 -- Finalizamos a transação confirmando Cliente A e B
275 • COMMIT;
276
277 -- Verificar resultado final
278 • SELECT * FROM clientes
279   WHERE email IN ('clienteA@email.com', 'clienteB@email.com', 'clienteC@email.com');
280
281
```

Result Grid

	id	nome	email	ativo
▶	106	Cliente A	clienteA@email.com	1
	107	Cliente B	clienteB@email.com	1
*	NULL	NULL	NULL	NULL