

Atividade prática | Transações em banco de dados

Um banco possui duas contas bancárias. Você realizará transferências entre elas. Execute as tarefas na ordem apresentada, na mesma conexão com o MySQL. Execute os blocos separadamente para observar os resultados.

```
CREATE DATABASE atividade_transacoes;
USE atividade_transacoes;
SET autocommit = 1;
```

```
CREATE TABLE conta (
    id INT
    PRIMARY KEY,
    titular
    VARCHAR(100) NOT NULL,
    saldo
    DECIMAL(10,2) NOT NULL )
ENGINE=InnoDB;
```

```
INSERT INTO conta VALUES
(1, 'Ana', 1000.00),
(2, 'Bruno', 500.00);
```

```
SELECT * FROM conta;
```

Registre os resultados de cada etapa para responder às questões no relatório. A tabela utiliza InnoDB para dar suporte às transações.

1) Transferência sem transação

Transfira **R\$ 200,00 de Ana para Bruno**, utilizando os dois comandos UPDATE abaixo, sem iniciar uma transação explícita.

```
UPDATE conta SET saldo = saldo - 200 WHERE id = 1;
UPDATE conta SET saldo = saldo + 200 WHERE id = 2;
```

```
SELECT * FROM conta;
```

Responda no relatório:

1. Qual era o saldo inicial das duas contas?
2. Qual foi o saldo final?
3. O que aconteceria se o sistema executasse somente o primeiro UPDATE e apresentasse um erro antes do segundo?

2)Utilizando uma transação

Restaure os valores originais:

```
UPDATE conta SET saldo = 1000 WHERE id = 1;  
UPDATE conta SET saldo = 500 WHERE id = 2;
```

Agora execute e observe os saldos antes de finalizar:

```
START TRANSACTION;  
  
UPDATE conta SET saldo = saldo - 200 WHERE id = 1;  
UPDATE conta SET saldo = saldo + 200 WHERE id = 2;  
  
SELECT * FROM conta;
```

Confirme a transação e consulte novamente:

```
COMMIT;  
SELECT * FROM conta;
```

4. O que o START TRANSACTION iniciou?

5. Qual é a função do COMMIT?

6. Depois do COMMIT, os valores permaneceram alterados?

3)Testando ROLLBACK

Sem restaurar os saldos, execute uma nova transferência de **R\$ 100,00 de Bruno para Ana**:

```
START TRANSACTION;  
  
UPDATE conta SET saldo = saldo - 100 WHERE id = 2;  
UPDATE conta SET saldo = saldo + 100 WHERE id = 1;  
  
SELECT * FROM conta;
```

Desta vez, não execute COMMIT. Desfaça a transação e consulte:

```
ROLLBACK;  
SELECT * FROM conta;
```

7. O que aconteceu com os valores após o ROLLBACK?

8. Explique com suas palavras a diferença entre COMMIT e ROLLBACK.

4) Simulando um problema real

Considere uma transferência de **R\$ 300,00 de Ana para Bruno**. Execute apenas o débito:

```
START TRANSACTION;
```

```
UPDATE conta SET saldo = saldo - 300 WHERE id = 1;
```

```
SELECT * FROM conta;
```

Imagine que ocorreu uma falha e o crédito na conta de Bruno não pôde ser executado. Simule o tratamento dessa falha cancelando a transação:

```
ROLLBACK;
```

```
SELECT * FROM conta;
```

9. Por que utilizar uma transação e executar ROLLBACK evita que o banco fique inconsistente nesse caso?

10. Dê outro exemplo de sistema real no qual várias operações deveriam ser executadas dentro de uma única transação.