



LAB_05

Participantes: Adriano Vale, Ary Farah, Caroline Assis, Ícaro Kuchanovicz

5.1 → Function

```
CREATE DATABASE lab_05;
USE lab_05;

-- a)
DELIMITER $$
-- function: não usa IN / OUT nos parâmetros da função
CREATE FUNCTION Diagonal (ladoA FLOAT, ladoB FLOAT)
RETURNS FLOAT
DETERMINISTIC -- define que a função é determinística
BEGIN
    DECLARE DIAG FLOAT DEFAULT -1;
    SET DIAG = SQRT(POWER(ladoA, 2) + POWER(ladoB, 2));
    RETURN DIAG;
END; $$
DELIMITER ;
SELECT Diagonal(3, 4) AS 'Diagonal do retângulo 3m x 4m';
```

1. Essa FUNCTION tem parâmetros de entrada? Se sim, quais e de quais TIPOS?

Sim, são eles: ladoA e ladoB. Ambos do tipo FLOAT.

2. Qual a diferença de RETURNS e RETURN, nas linhas .7 e .12, respectivamente?

RETURNS: declara o tipo de dado a ser retornado, no caso acima, FLOAT.

RETURN: indica o valor a ser retornado.

3. Em qual linha do código chamamos a FUNCTION definida em (a)?

A **FUNCTION** criada é chamada na linha 17 (última).

4. Apresente e explique o resultado do SELECT.

Diagonal do retângulo 3m x 4m
5

É calculado a hipotenusa de um triângulo através do Teorema de Pitágoras, nesse caso com os catetos de 3 e 4 metros.

```
-- b)
DELIMITER $$
CREATE FUNCTION CalcSalario (valor_inicial INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE salario INT DEFAULT 0;
    WHILE salario <= 3000 DO
        SET salario = salario + valor_inicial;
    END WHILE;
    RETURN salario;
END; $$
```

```
DELIMITER ;
SELECT CalcSalario(500) AS 'Salário Final';
```

1. Essa FUNCTION tem parâmetros de entrada? Se sim, quais e de quais TIPOS?

Sim, é chamado de 'valor_inicial' e é do tipo INT.

2. Em qual linha do código chamamos a FUNCTION definida em (b)?

A function é chamada na linha 16 (última).

3. Apresente e explique o resultado do SELECT.

Salário Final
3500

É definida uma variável chamada 'salario' com o valor DEFAULT = 0. Enquanto 'salario' for menor ou igual a 3000, será adicionado a ele o valor de 'valor_inicial'.

Nesse caso, 'valor_inicial' era igual a 500. Assim, foi sendo adicionado ao valor de 'salario' até que esse fosse maior de 3000.

4. Qual é o Salário Final para um valor inicial = 200?

Salário Final
3200

5. Qual é o Salário Final para um valor inicial = 2000?

Salário Final
4000

5.2 → Commit / Rollback

```
-- a)
DROP TABLE IF EXISTS Tab_Testes;
CREATE TABLE Tab_Testes (
  col1 INT NOT NULL PRIMARY KEY,
  col2 INT NOT NULL);
SELECT * FROM Tab_Testes;

-- b)
START TRANSACTION ;
INSERT Tab_Testes VALUES (1,111) ;
INSERT Tab_Testes VALUES (2,222) ;
COMMIT;
SELECT * FROM Tab_Testes;

-- c)
START TRANSACTION ;
INSERT Tab_Testes VALUES (3,333) ;
INSERT Tab_Testes VALUES (4,444) ;
```

```
ROLLBACK;  
SELECT * FROM Tab_Testes;
```

1. Em (b), os comandos de INSERT da transação funcionaram? Por que? Justifique, apresentado também o resultado do SELECT.

col1	col2
1	111
2	222
NULL	NULL

Os comandos funcionaram porque foi utilizado um `COMMIT`, que confirma a execução dos comandos dentro de uma TRANSACTION.

2. Em (c), os comandos de INSERT da transação funcionaram? Por que? Justifique, apresentado também o resultado do SELECT.

col1	col2
1	111
2	222
NULL	NULL

Os comandos não funcionaram porque foi utilizado um 'ROLLBACK', que não autoriza a execução dos comandos, voltando o código para o estado salvo antes do início da TRANSACTION.

5.3 → Transação SEM tratamento de erro

```
-- a)  
DROP TABLE IF EXISTS Tab_Testes;  
CREATE TABLE Tab_Testes (  
  col1 INT NOT NULL PRIMARY KEY,  
  col2 INT NOT NULL);  
  
SELECT * FROM Tab_Testes;  
  
-- b)  
DROP PROCEDURE IF EXISTS nãoTratErroTransact;  
  
DELIMITER $$  
CREATE PROCEDURE nãoTratErroTransact()  
BEGIN  
  START TRANSACTION;  
  INSERT Tab_Testes VALUES (1,111) ;  
  INSERT Tab_Testes VALUES (2,222) ;  
  INSERT Tab_Testes VALUES (3,333) ;  
  INSERT Tab_Testes VALUES (1,101) ;  
  COMMIT; -- esse commando executa?  
END $$  
DELIMITER ;  
  
CALL nãoTratErroTransact();  
SELECT * FROM Tab_Testes;
```

1. Em (a), apresente e explique o resultado do SELECT.

col1	col2
NULL	NULL

É criada uma tabela com duas colunas de tipo INT e NOT NULL, col1 sendo uma PK.

2. Em (b), a SP tem algum comando incorreto? Qual e porque está incorreto?

Sim, o comando `INSERT Tab_Testes VALUES (1,101)`, pois ele repete o valor da chave primária, que deve ser única.

3. Em (b), o COMMIT da SP executa? Explique.

O COMMIT é executado, porém somente para os INSERTS que não deram erro, pulando aquela linha.

4. Em (b), apresente e explique o resultado do SELECT.

	col1	col2
▶	1	111
	2	222
	3	333
•	NULL	NULL

Os primeiros 3 INSERTs são realizados, já o quarto, que é quando repete a chave primária, apresenta um erro e não é realizado.

5.4 → Transação COM tratamento de erro

```
-- a)
DROP TABLE IF EXISTS Tab_Testes;
CREATE TABLE Tab_Testes (
col1 INT NOT NULL PRIMARY KEY,
col2 INT NOT NULL);

SELECT * FROM Tab_Testes;

-- b)
DROP PROCEDURE IF EXISTS tratErroTransact;
DELIMITER $$

CREATE PROCEDURE tratErroTransact()
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
ROLLBACK;
RESIGNAL;
END;
START TRANSACTION;
INSERT Tab_Testes VALUES (1,111) ;
INSERT Tab_Testes VALUES (2,222) ;
INSERT Tab_Testes VALUES (3,'um') ;
INSERT Tab_Testes VALUES (3,333) ;
COMMIT; -- esse comando executa?
END $$
DELIMITER ;

CALL tratErroTransact();
SELECT * FROM Tab_Testes;
```

1. Em (a), apresente e explique o resultado do SELECT.

col1	col2
NULL	NULL

É criada uma tabela com duas colunas de tipo INT e NOT NULL, col1 sendo uma PK.

2. Em (b), a SP tem algum comando incorreto? Qual e porque está incorreto?

Sim, os comandos `INSERT Tab_Testes VALUES (3, 'um');` e `INSERT Tab_Testes VALUES (3,333);`

O primeiro por tentar incluir uma STRING em um campo INT, o segundo por repetir a PK = 3.

3. Em (b), o COMMIT da SP executa? Explique.

Não, pois foi dito para que se algum erro fosse encontrado em `DECLARE EXIT HANDLER FOR SQLEXCEPTION... END;` o comando `ROLLBACK` seja executado, voltando o estado do código para o salvo antes do erro. (Nesse caso, a criação da tabela)

4. Em (b), apresente e explique o resultado do SELECT.

	col1	col2
*	NULL	NULL

Os comandos não foram realizados pois devido ao tratamento de erro e o comando `ROLLBACK`, o código voltou para o estado salvo antes do erro acontecer (criação da tabela vazia.)

5.5 → Trigger

```
-- a)

DROP TABLE IF EXISTS EstoqueProduto;
CREATE TABLE EstoqueProduto (
  ID_Prod INT PRIMARY KEY,
  Nome_Prod VARCHAR(20) NOT NULL UNIQUE,
  Estoque INT NOT NULL
);
INSERT INTO EstoqueProduto (ID_Prod, Nome_Prod, Estoque) VALUES (123, 'Caderno', 100);
INSERT INTO EstoqueProduto (ID_Prod, Nome_Prod, Estoque) VALUES (456, 'Bloco A4', 50);
INSERT INTO EstoqueProduto (ID_Prod, Nome_Prod, Estoque) VALUES (789, 'Caneta', 200);

SELECT * FROM EstoqueProduto;

DROP TABLE IF EXISTS ItensVenda;
CREATE TABLE ItensVenda (
  ID_Venda INT AUTO_INCREMENT PRIMARY KEY,
  fk_ID_Pedido INT, -- Tab Pedido não criada nesta demonstração
  fk_ID_Prod INT NOT NULL REFERENCES EstoqueProduto(ID_Prod), -- FK
  Quantidade INT NOT NULL, UNIQUE (fk_ID_Pedido, fk_ID_Prod)
);

SELECT * FROM ItensVenda;
```

1. O que significa o UNIQUE da tabela EstoqueProduto?

Significa que o campo 'Nome_Prod' deve ter valores únicos.

2. Apresente o resultado do comando SELECT * FROM EstoqueProduto;

ID_Prod	Nome_Prod	Estoque
123	Caderno	100
456	Bloco A4	50
789	Caneta	200
NULL	NULL	NULL

3. Apresente um comando de INSERT na tabela EstoqueProduto que viola a restrição de UNIQUE.

O comando que poderia violar é:

```
INSERT INTO EstoqueProduto (ID_Prod, NomeProd, Estoque) VALUES (100, 'Caneta', 150)
```

```
-- b)
DROP TRIGGER IF EXISTS Tgr_ItensVenda_Insert;

DELIMITER $$

CREATE TRIGGER Tgr_ItensVenda_Insert
AFTER INSERT
ON ItensVenda
FOR EACH ROW
BEGIN
    UPDATE EstoqueProduto
    SET Estoque = Estoque - NEW.Quantidade
    WHERE ID_Prod = NEW.fk_ID_Prod;
END $$

DELIMITER ;

-- c)
DROP TRIGGER IF EXISTS Tgr_ItensVenda_Delete;

DELIMITER $$

CREATE TRIGGER Tgr_ItensVenda_Delete
AFTER DELETE
ON ItensVenda
FOR EACH ROW
BEGIN
    UPDATE EstoqueProduto
    SET Estoque = Estoque + OLD.Quantidade
    WHERE ID_Prod = OLD.fk_ID_Prod;
END $$

DELIMITER ;
```

1. Sobre o TRIGGER em (b):

- Ele atua para qual tabela? Para a tabela 'EstoqueProduto'
- Quando ele é disparado? Depois de um `INSERT` feito na tabela 'ItensVenda'
- Que atualização ele realiza em que outra tabela? Diminui da tabela 'EstoqueProduto' o mesmo número inserido na coluna 'Quantidade' de 'ItensVenda' de onde o 'ID_Prod' for igual ao novo inserido.

2. Sobre o TRIGGER em (c):

- Ele atua para qual tabela? Para a tabela 'EstoqueProduto'
- Quando ele é disparado? Depois de um `DELETE` feito na tabela 'ItensVenda'
- Que atualização ele realiza em que outra tabela? Aumenta na tabela 'EstoqueProduto' o mesmo número deletado da coluna 'Quantidade' de 'ItensVenda' onde o 'ID_Prod' for igual ao deletado.

```
-- d)
INSERT INTO ItensVenda (fk_ID_Pedido, fk_ID_Prod, Quantidade) VALUES (1, 123, 30);
INSERT INTO ItensVenda (fk_ID_Pedido, fk_ID_Prod, Quantidade) VALUES (1, 456, 10);
INSERT INTO ItensVenda (fk_ID_Pedido, fk_ID_Prod, Quantidade) VALUES (1, 789, 25);

SELECT * FROM ItensVenda;
SELECT * FROM EstoqueProduto;

-- e)
DELETE FROM ItensVenda WHERE fk_ID_Pedido = 1 AND fk_ID_Prod = 123;
```

```
DELETE FROM ItensVenda WHERE fk_ID_Pedido = 1 AND fk_ID_Prod = 789;

SELECT * FROM ItensVenda;
SELECT * FROM EstoqueProduto;
```

1. Sobre o comandos em (d):

- Qual o TRIGGER é disparado pelos INSERTs? `Tgr_ItensVenda_Insert`
- Após os INSERTs, o que foi alterado, em que tabelas? Por que? Além do `INSERT` na tabela ItensVenda, os produtos 'Caderno', 'Bloco A4' e 'Caneta' tiveram o estoque diminuído na tabela EstoqueProduto em 30, 10 e 25 respectivamente. Isso aconteceu por conta do `TRIGGER` que foi acionado.
- Apresente os resultados dos SELECTs.

ID_Prod	Nome_Prod	Estoque
123	Caderno	70
456	Bloco A4	40
789	Caneta	175
NULL	NULL	NULL

EstoqueProduto

ID_Venda	fk_ID_Pedido	fk_ID_Prod	Quantidade
1	1	123	30
2	1	456	10
3	1	789	25
NULL	NULL	NULL	NULL

ItensVenda

2. Sobre o comandos em (e):

- Qual o TRIGGER é disparado pelos DELETES? `Tgr_ItensVenda_Delete`
- Após os DELETES, o que foi alterado, em que tabelas? Por que? Além do `DELETE` na tabela ItensVenda, os produtos 'Caderno' e 'Caneta' tiveram o estoque aumentado na tabela EstoqueProduto em 30 e 25 respectivamente. Isso aconteceu por conta do `TRIGGER` que foi acionado.
- Apresente os resultados dos SELECTs

ID_Prod	Nome_Prod	Estoque
123	Caderno	100
456	Bloco A4	40
789	Caneta	200
NULL	NULL	NULL

EstoqueProduto

ID_Venda	fk_ID_Pedido	fk_ID_Prod	Quantidade
2	1	456	10
NULL	NULL	NULL	NULL

ItensVenda

5.6 → Triggers para LOG

```
-- a)
CREATE TABLE Editora (
  ID_edit INT AUTO_INCREMENT PRIMARY KEY, -- Tabela PAI
  Nome_Edit VARCHAR(60) NOT NULL,
  Cidade VARCHAR(60) NOT NULL,
  Estado CHAR(2) NOT NULL,
  Pais VARCHAR(50) NOT NULL
);

INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('Editora AAA', 'São Paulo', 'SP', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('Editora Sul', 'Porto Alegre', 'RS', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('LTC', 'São Paulo', 'SP', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('CENGAGE', 'Rio de Janeiro', 'RJ', 'Brasil');
INSERT Editora (Nome_Edit, Cidade, Estado, Pais) VALUES ('Três Estrelas', 'Alagoas', 'CE', 'Brasil');

SELECT * FROM Editora;

-- b)
CREATE TABLE Autor(
```

```

ID_Autor INT AUTO_INCREMENT PRIMARY KEY, -- Tabela FILHO
Nome_Autor VARCHAR(60) NOT NULL,
Dt_Nasc DATE NOT NULL,
fk_ID_Edit INT NULL
);

ALTER TABLE Autor ADD CONSTRAINT FK_Autor_Editora FOREIGN KEY(fk_ID_edit)
REFERENCES Editora (ID_edit);
ALTER TABLE Autor AUTO_INCREMENT = 100; -- Seed = 100 (início do AUTO_INCREMENT)
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('José', '1956-09-08', 1);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Maria', '1975-04-18', 2);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Antônia', '1954-12-10', 3);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Arminio', '1976-07-28', 5);
INSERT Autor (Nome_Autor, Dt_Nasc, fk_ID_Edit) VALUES ('Luiza', '1945-11-09', 5);

SELECT * FROM Autor;

```

1. Apresente os resultados dos SELECTs em Autor e Editora;

ID_edit	Nome_Edit	Cidade	Estado	Pais
1	Editora AAA	São Paulo	SP	Brasil
2	Editora Sul	Porto Alegre	RS	Brasil
3	LTC	São Paulo	SP	Brasil
4	CENGAGE	Rio de Janeiro	RJ	Brasil
5	Três Estrelas	Alagoas	CE	Brasil
NULL	NULL	NULL	NULL	NULL

ID_Autor	Nome_Autor	Dt_Nasc	fk_ID_Edit
100	José	1956-09-08	1
101	Maria	1975-04-18	2
102	Antônia	1954-12-10	3
103	Arminio	1976-07-28	5
104	Luiza	1945-11-09	5
NULL	NULL	NULL	NULL

Autor

Editora

```

-- c)
DROP TABLE IF EXISTS AutorLog;

-- Tabela de LOG (rastreamento) referente à Tabela Autor
CREATE TABLE AutorLog (
ID_log INT AUTO_INCREMENT PRIMARY KEY,
Operation CHAR(6) NOT NULL, -- Operação realizada
ChangeDate DATETIME NOT NULL, -- Data da realização da operação
UserName VARCHAR(20) NOT NULL, -- Usuário de BD que realizou a operação
OldID_Autor INT NULL, -- Valor antigo para ID_Autor
NewID_autor INT NULL, -- Valor novo para ID_Autor
OldAutor VARCHAR(50) NULL, -- Valor antigo para Nome de Autor
NewAutor VARCHAR(50) NULL, -- Valor novo para Nome de Autor
OldDtNasc DATE NULL, -- Valor antigo para Data de Nascimento do Autor
NewDtNasc DATE NULL, -- Valor novo para Data de Nascimento do Autor
OldID_Edit INT NULL, -- Valor antigo para ID do Editor do Autor
NewID_Edit INT NULL -- Valor novo para ID do Editor do Autor
);

SELECT * FROM AutorLog;

```

1. Para que servem os campos ID_log, Operation e ChangeDate e UserName?

ID_log → Para identificação, já que é uma chave primária

Operation → Nome da operação realizada

ChangeDate → Data da realização da operação

UserName → Usuário do Banco de Dados que realizou a operação

2. Qual a diferença entre os campos OldID_autor e NewID_autor?

OldID_autor → Valor para o ID_Autor antigo

NewID_autor → Valor para o ID_Autor novo

3. Qual a diferença entre os campos OldAutor e NewAutor?

OldAutor → Nome do ator antigo

NewAutor → Nome do ator novo

4. Apresente o resultado do SELECT em AutorLog . Quando esta tabela será povoada?

ID_log	Operation	ChangeDate	UserName	OldID_Autor	NewID_autor	OldAutor	NewAutor	OldDtNasc	NewDtNasc	OldID_Edit	NewID_Edit
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Esta tabela será povoada quando forem feitos INSERTS nela ou quando for alterado algo na tabela Autor.

```
-- d)

DROP TRIGGER IF EXISTS AutorLogInsert;

DELIMITER $$

CREATE TRIGGER AutorLogInsert
AFTER INSERT -- A Trigger dispara após o INSERT
ON Autor
FOR EACH ROW
BEGIN
    INSERT INTO -- Insere registro na tabela AutorLog
    AutorLog (Operation, ChangeDate, UserName, NewID_Autor, NewAutor, NewDtNasc, NewID_Edit)
    SELECT 'Insert', NOW(), CURRENT_USER(), NEW.ID_autor, NEW.nome_autor, NEW.dt_nasc, NEW.fk_ID_edit;
END $$

DELIMITER ;

-- e)

DROP TRIGGER IF EXISTS AutorLogDelete;

DELIMITER $$

CREATE TRIGGER AutorLogDelete
AFTER DELETE -- A Trigger dispara após o DELETE
ON Autor
FOR EACH ROW
BEGIN
    INSERT INTO -- Insere registro na tabela AutorLog
    AutorLog (Operation, ChangeDate, UserName, OldID_Autor, OldAutor, OldDtNasc, OldID_Edit)
    SELECT 'Delete', NOW(), CURRENT_USER(), OLD.ID_autor, OLD.nome_autor, OLD.dt_nasc, OLD.fk_ID_edit;
END $$

DELIMITER ;

-- f)

DROP TRIGGER IF EXISTS AutorLogUpdate;

DELIMITER $$

CREATE TRIGGER AutorLogUpdate
AFTER UPDATE -- A Trigger dispara após o UPDATE
ON Autor
FOR EACH ROW
BEGIN
    INSERT INTO -- Insere registro na tabela AutorLog
    AutorLog (Operation, ChangeDate, UserName, OldID_Autor, NewID_Autor, OldAutor, NewAutor, OldDtNasc, NewDtNasc, OldID_Edit, NewID_Edit)
    SELECT 'Update', NOW(), CURRENT_USER(), OLD.ID_autor, NEW.ID_autor, OLD.nome_autor, NEW.nome_autor, OLD.dt_nasc, NEW.dt_nasc, OLD.fk_ID_edit, NEW.fk_ID_edit;
END $$

DELIMITER ;
```

1. Sobre o TRIGGER em (d):

- Ele atua para qual tabela? Para a tabela AutorLog
- Quando ele é disparado? Após um **INSERT** na tabela Autor
- Que atualização ele realiza em que outra tabela? Ele insere na tabela AutorLog os valores

```
'Insert', NOW(), CURRENT_USER(), NEW.ID_autor, NEW.nome_autor, NEW.dt_nasc, NEW.fk_ID_edit
```

para as colunas: (respectivamente)

```
(Operation, ChangeDate, UserName, NewID_Autor, NewAutor, NewDtNasc, NewID_Edit)
```

2. Sobre o TRIGGER em (e):

- Ele atua para qual tabela? Para a tabela AutorLog
- Quando ele é disparado? Após um `DELETE` na tabela Autor
- Que atualização ele realiza em que outra tabela? Ele insere na tabela AutorLog os valores

```
'Delete', NOW(), CURRENT_USER(), OLD.ID_autor, OLD.nome_autor, OLD.dt_nasc, OLD.fk_ID_edit
```

para as colunas: (respectivamente)

```
(Operation, ChangeDate, UserName, OldID_Autor, OldAutor, OldDtNasc, OldID_Edit)
```

3. Sobre o TRIGGER em (f):

- Ele atua para qual tabela? Para a tabela AutorLog
- Quando ele é disparado? Após um `UPDATE` na tabela Autor
- Que atualização ele realiza em que outra tabela? Ele insere na tabela AutorLog os valores

```
'Update', NOW(), CURRENT_USER(), OLD.ID_autor, NEW.ID_autor, OLD.nome_autor, NEW.nome_autor, OLD.dt_nasc, NEW.dt_nasc,  
OLD.fk_ID_edit, NEW.fk_ID_edit
```

para as colunas: (respectivamente)

```
(Operation, ChangeDate, UserName, OldID_Autor, NewID_Autor, OldAutor, NewAutor, OldDtNasc, NewDtNasc, OldID_Edit, NewID_Edit)
```

```
-- g)  
-- Comandos 1) Teste de UPDATE ----  
UPDATE Autor SET nome_autor = 'José da Silva'  
WHERE ID_autor = 100;  
  
SELECT * FROM Autor;  
SELECT * FROM AutorLog;  
  
-- Comandos 2) Teste de INSERT ----  
INSERT Autor (nome_autor, dt_nasc, fk_ID_Edit)  
VALUES  
( 'Karolina', '1976-06-18', 3),  
( 'Cláudio', '1982-10-28', 4),  
( 'Ricardo', '1990-02-13', 3);  
  
SELECT * FROM Autor;  
SELECT * FROM AutorLog;  
  
-- Comandos 3) Teste de DELETE ---  
DELETE FROM Autor  
WHERE ID_autor = 102 OR ID_autor = 103;  
  
SELECT * FROM Autor;  
SELECT * FROM AutorLog;
```

1. Sobre o Comandos 1 em (g):

- Qual foi o Trigger disparado? `AutorLogUpdate`
- O que foi preenchido em que tabela?

Tabela: Autor

Campos: `Nome_Autor`

Valores: `'José da Silva'`

Tabela: AutorLog

Campos: (ID_Log, Operation, ChangeDate, UserName, OldID_Autor, NewID_Autor, OldAutor, NewAutor, OldDtNasc, NewDtNasc, OldID_Edit, NewID_Edit)

Valores: 1, 'Update', '2023-11-07 19:36:05', 'root@localhost', 100, 100, José, José da Silva, '1956-09-08', '1956-09-08', '1956-09-08', 1, 1

- Apresente o resultado dos SELECTs?

ID_Autor	Nome_Autor	Dt_Nasc	fk_ID_Edit
100	José da Silva	1956-09-08	1
101	Maria	1975-04-18	2
102	Antônia	1954-12-10	3
103	Arminio	1976-07-28	5
104	Luiza	1945-11-09	5

ID_Log	Operation	ChangeDate	UserName	OldID_Autor	NewID_autor	OldAutor	NewAutor	OldDtNasc	NewDtNasc	OldID_Edit	NewID_Edit
1	Update	2023-11-07 19:36:05	root@localhost	100	100	José	José da Silva	1956-09-08	1956-09-08	1	1

2. Sobre o Comandos 2 em (g):

- Qual foi o Trigger disparado? AutorLogInsert
- O que foi preenchido em que tabela?

Tabela: Autor

Campos: (Nome_Autor, Dt_Nasc, fk_ID_Edit)

Valores: ('Karolina', '1976-06-18', 3),

('Cláudio', '1982-10-28', 4),

('Ricardo', '1990-02-13', 3);

Tabela: AutorLog

Campos: (ID_Log, Operation, ChangeDate, UserName, OldID_Autor, NewID_Autor, OldAutor, NewAutor, OldDtNasc, NewDtNasc, OldID_Edit, NewID_Edit)

Valores: 2, 'Insert', '2023-11-07 19:45:58', 'root@localhost', ' ', 105, ' ', 'Karolina', ' ', ' ', '1976-06-18', ' ', 3

3, 'Insert', '2023-11-07 19:45:58', 'root@localhost', ' ', 106, ' ', 'Cláudio', ' ', ' ', '1982-10-28', ' ', 4

4, 'Insert', '2023-11-07 19:45:58', 'root@localhost', ' ', 107, ' ', 'Ricardo', ' ', ' ', '1990-02-13', ' ', 3

- Apresente o resultado dos SELECTs?

ID_Autor	Nome_Autor	Dt_Nasc	fk_ID_Edit
100	José da Silva	1956-09-08	1
101	Maria	1975-04-18	2
102	Antônia	1954-12-10	3
103	Arminio	1976-07-28	5
104	Luiza	1945-11-09	5
105	Karolina	1976-06-18	3
106	Cláudio	1982-10-28	4
107	Ricardo	1990-02-13	3

ID_Log	Operation	ChangeDate	UserName	OldID_Autor	NewID_autor	OldAutor	NewAutor	OldDtNasc	NewDtNasc	OldID_Edit	NewID_Edit
1	Update	2023-11-07 19:36:05	root@localhost	100	100	José	José da Silva	1956-09-08	1956-09-08	1	1
2	Insert	2023-11-07 19:45:58	root@localhost	NULL	105	NULL	Karolina	NULL	1976-06-18	NULL	3
3	Insert	2023-11-07 19:45:58	root@localhost	NULL	106	NULL	Cláudio	NULL	1982-10-28	NULL	4
4	Insert	2023-11-07 19:45:58	root@localhost	NULL	107	NULL	Ricardo	NULL	1990-02-13	NULL	3

3. Sobre o Comandos 3 em (g):

- Qual foi o Trigger disparado? AutorLogDelete
- O que foi preenchido em que tabela?

Tabela: Autor

Campos: (ID_Autor, Nome_Autor, Dt_Nasc, fk_ID_Edit)

Valores: As linha onde ID_Autor = 102 ou 103 foram deletadas.

Tabela: AutorLog

Campos: (ID_Log, Operation, ChangeDate, UserName, OldID_Autor, NewID_Autor, OldAutor, NewAutor, OldDtNasc, NewDtNasc, OldID_Edit, NewID_Edit)

Valores:

```
5, 'Delete', '2023-11-07 20:12:49', 'root@localhost', 102, ' ', 'Antônia', ' ', '1954-12-10', ' ', ' ', '3', ' '
6, 'Delete', '2023-11-07 20:12:49', 'root@localhost', 103, ' ', ' ', 'Armínio', ' ', ' ', '1976-07-28', ' ', ' ', '5', ' '
```

- Apresente o resultado dos SELECTs?

ID_Autor	Nome_Autor	Dt_Nasc	fk_ID_Edit
100	José da Silva	1956-09-08	1
101	Maria	1975-04-18	2
104	Luiza	1945-11-09	5
105	Karolina	1976-06-18	3
106	Cláudio	1982-10-28	4
107	Ricardo	1990-02-13	3

ID_Log	Operation	ChangeDate	UserName	OldID_Autor	NewID_autor	OldAutor	NewAutor	OldDtNasc	NewDtNasc	OldID_Edit	NewID_Edit
1	Update	2023-11-07 19:36:05	root@localhost	100	100	José	José da Silva	1956-09-08	1956-09-08	1	1
2	Insert	2023-11-07 19:45:58	root@localhost	NULL	105	NULL	Karolina	NULL	1976-06-18	NULL	3
3	Insert	2023-11-07 19:45:58	root@localhost	NULL	106	NULL	Cláudio	1982-10-28	1982-10-28	NULL	4
4	Insert	2023-11-07 19:45:58	root@localhost	NULL	107	NULL	Ricardo	NULL	1990-02-13	NULL	3
5	Delete	2023-11-07 20:12:49	root@localhost	102	NULL	Antônia	NULL	1954-12-10	1954-12-10	3	NULL
6	Delete	2023-11-07 20:12:49	root@localhost	103	NULL	Arminio	NULL	1976-07-28	1976-07-28	5	NULL