

# C07 - BD

Arthur Openheimer

# Triggers

Triggers são ativados quando uma declaração (INSERT, UPDADTE ou DELETE) ocorre na tabela associada

O gatilho pode ser ativado ANTES ou DEPOIS da declaração ser realizada

Seus principais usos são para executar verificações de valores ou fazer cálculos sobre os valores informados ao executar uma declaração

# Triggers

**CREATE** [DEFINER = {user | CURRENT\_USER}] **TRIGGER** <trigger\_name>  
<trigger\_time><trigger\_event> **ON** <table\_name> **FOR EACH ROW** <trigger\_statement>

# Triggers

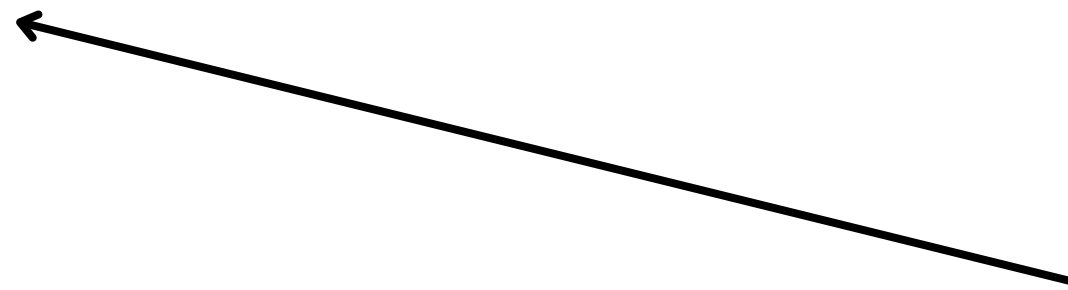
DEFINER - São comandos opcionais para um trigger, porém não iremos abordar eles aqui



**CREATE** [DEFINER = {user | CURRENT\_USER}] **TRIGGER** <trigger\_name>  
<trigger\_time><trigger\_event> **ON** <table\_name> **FOR EACH ROW** <trigger\_statement>

# Triggers

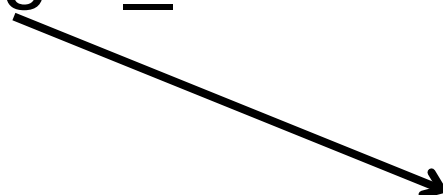
<trigger\_name> - Define o nome para o processo trigger



```
CREATE [DEFINER = {user | CURRENT_USER}] TRIGGER <trigger_name>  
<trigger_time><trigger_event> ON <table_name> FOR EACH ROW <trigger_statement>
```

# Triggers

**CREATE** [DEFINER = {user | CURRENT\_USER}] **TRIGGER** <trigger\_name>  
<trigger\_time><trigger\_event> **ON** <table\_name> **FOR EACH ROW** <trigger\_statement>



<trigger\_time> - Define se o trigger será ativado antes (BEFORE) ou depois (AFTER) do comando que o disparou

BEFORE - Antes dos dados chegarem à tabela

AFTER - Depois da atualização dos dados na tabela

# Triggers

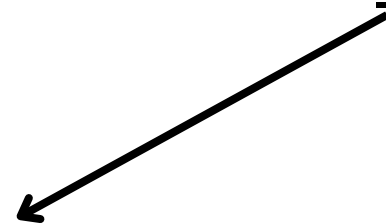
**CREATE** [DEFINER = {user | CURRENT\_USER}] **TRIGGER** <trigger\_name>  
<trigger\_time><trigger\_event> **ON** <table\_name> **FOR EACH ROW** <trigger\_statement>



<trigger\_event> - Define qual será o comando de disparo (INSERT, UPDATE ou DELETE)

# Triggers

**CREATE** [DEFINER = {user | CURRENT\_USER}] **TRIGGER** <trigger\_name>  
<trigger\_time><trigger\_event> **ON** <table\_name> **FOR EACH ROW** <trigger\_statement>

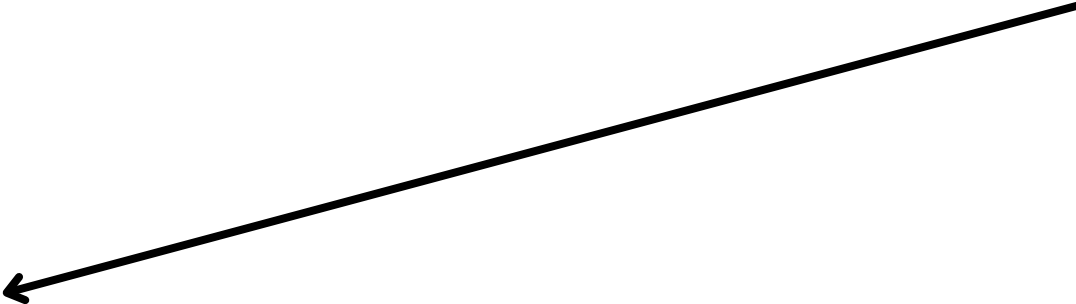


<table\_name> - Nome da tabela cujo  
trigger estará associado, aguardando o  
trigger event



# Triggers

**CREATE** [DEFINER = {user | CURRENT\_USER}] **TRIGGER** <trigger\_name>  
<trigger\_time><trigger\_event> **ON** <table\_name> **FOR EACH ROW** <trigger\_statement>



<trigger\_statement> - Procedimento que o trigger irá realizar, iniciado com o comando BEGIN e finalizado com o comando END

# Triggers

```
CREATE TABLE IF NOT EXISTS conta(  
    numero INT PRIMARY KEY,  
    total FLOAT  
);  
  
CREATE TRIGGER int_soma BEFORE INSERT  
ON conta  
FOR EACH ROW SET @soma = @soma + NEW.total;  
  
SET @soma = 0;  
  
INSERT INTO conta() VALUES  
(1,20),  
(2,180),  
(3,-50);  
  
SELECT @soma;
```

	@soma
▶	150

Comandos OLD e NEW - Servem para obter os valores antes ou depois de uma modificação

OLD - Recupera o valor de um campo antes de um UPDATE ou DELETE

NEW - Recupera o valor de um campo depois de um INSERT ou UPDATE

# Triggers

```
DELIMITER $$  
CREATE TRIGGER update_check BEFORE UPDATE  
ON conta  
FOR EACH ROW  
BEGIN  
  
IF NEW.total < 0 THEN  
SET NEW.total = 0;  
ELSEIF NEW.total > 100 THEN  
SET NEW.total = 100;  
END IF;  
  
END; $$  
DELIMITER ;  
  
UPDATE conta SET total = 200 WHERE numero = 2;  
UPDATE conta SET total = -92 WHERE numero = 3;  
  
SELECT * FROM conta;
```

	numero	total
▶	1	20
	2	100
	3	0
⌵	NULL	NULL

# Triggers

## Vantagens x Desvantagens

Vantagens - Parte do processamento que seria executado na aplicação passa a ser executado pelo banco, poupando recursos, além de facilitar a manutenção

Desvantagens - Pessoas com acesso não autorizado ao banco podem visualizar e alterar o funcionamento dos triggers. Além disso, o uso de triggers requer maior conhecimento de BDs em SQL

# Views

Views simplificam a especificação de algumas consultas, evitando erros, pois elimina o uso de tabelas temporárias

Views também podem ser configuradas para mostrar diferentes resultados de acordo com o usuário que acessa o banco de dados

Views evitam com que sejam feitas alterações indevidas nas tabelas

# Views

```
CREATE VIEW NomeView AS (SELECT * FROM NomeTabela);  
DROP VIEW NomeView;
```

# Views

itens_solicitacao
id_solicitacao INT(11)
cod_item INT(11)
nome VARCHAR(20)
qtd INT(11)
Indexes
PRIMARY
Triggers
AFT INSERT trg_itens_solicitacao_insert
AFT DELETE trg_itens_solicitacao_delete

estoque
cod_item INT(11)
qtd_disponivel INT(11)
Indexes
PRIMARY

```
CREATE VIEW qtd_estoque AS (  
    SELECT  
        i.nome 'Item', e.qtd_disponivel 'Quantidade no estoque'  
    FROM  
        estoque e,  
        itens_solicitacao i  
    WHERE  
        e.cod_item = i.cod_item  
);  
  
SELECT * from qtd_estoque;
```

	Item	Quantidade no estoque
▶	Resistor 100k	1500
	Resistor 1k	500
	Capacitor 100n	300
	Transistor TBJ	600

# Exercício

```
DROP DATABASE IF EXISTS loja;
CREATE DATABASE loja;
USE loja;
SET GLOBAL log_bin_trust_function_creators = 1; -- Para permitir a criação de triggers

CREATE TABLE compra(
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    preco FLOAT,
    pagamento FLOAT,
    troco FLOAT
);

INSERT INTO compra(preco, pagamento) VALUES(9.5, 10.25);
INSERT INTO compra(preco, pagamento) VALUES(18.99, 25);
INSERT INTO compra(preco, pagamento) VALUES(5.99, 5.99);
INSERT INTO compra(preco, pagamento) VALUES(10.99, 10.99);
INSERT INTO compra(preco, pagamento) VALUES(15.99, 15.99);
select * from compra;
```

Crie uma VIEW que busca a quantidade de compras, usando a função COUNT(), cujo preço é maior ou igual a 10

Crie um TRIGGER que é executado ANTES do insert, que calcula e atribui o valor do troco (pagamento - preço)