



*Faculdade de
Tecnologia SENAI
“Roberto Mange”*



Aula 9 – Triggers

Triggers SQL

O Trigger no MySQL é um objeto de banco de dados (programa) associado a uma tabela. O termo trigger (gatilho em inglês) define uma estrutura do banco de dados que funciona, como o nome sugere, como uma função que é disparada mediante alguma ação. Será ativado quando uma ação definida for executada para a tabela. O trigger pode ser executado quando você executa uma das seguintes instruções do MySQL na tabela: INSERT , UPDATE e DELETE e pode ser chamada antes ou depois do evento



Principais triggers

Existem dois tipos principais: o AFTER e o BEFORE.

O AFTER é executado depois do evento que disparou o trigger. Ao passo que o BEFORE executa o trigger antes da ação DML.



Principais triggers

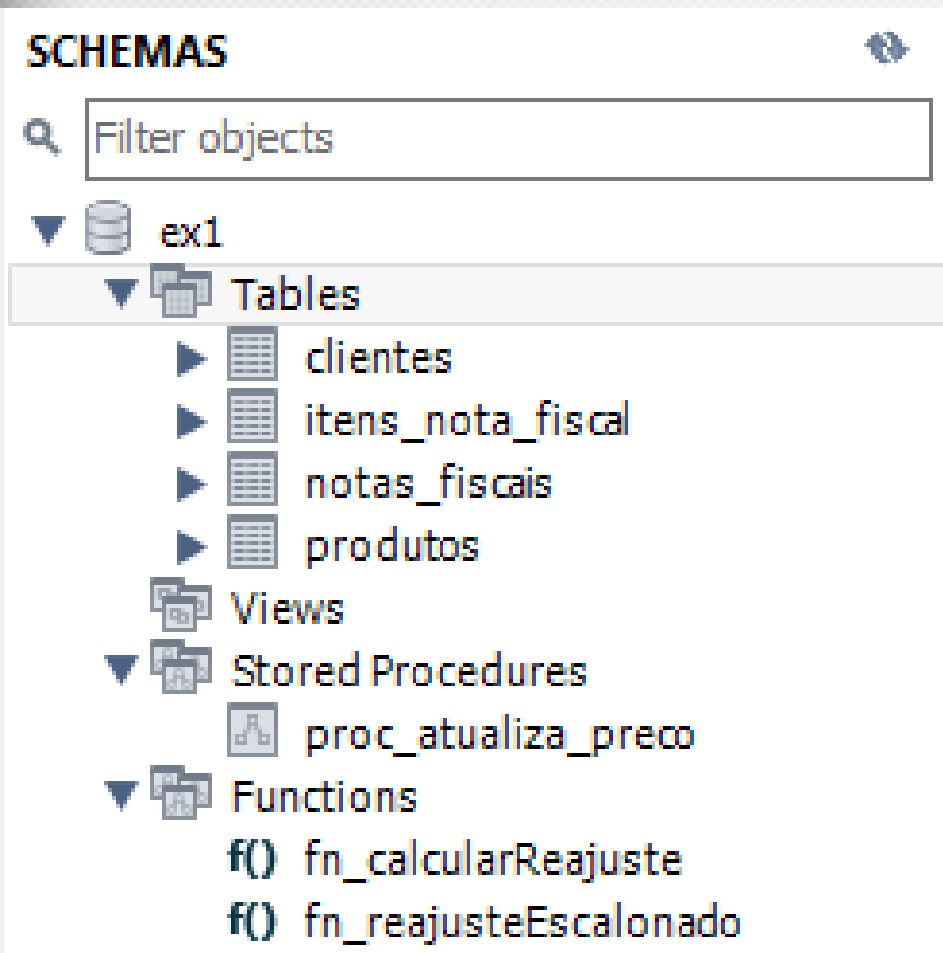
Triggers de linha → executado sempre que um linha é modificada, deverá ser acompanhada pela instrução “for each row”. Toda vez que a linha for modificada o trigger será executado também.

Triggers de instrução → executado uma única vez através de um comando DML,

Triggers de Logon → há ainda um que é executado quando um determinado usuário se conecta ao BD, não é comum a todos os BD dependendo do seu SGBD



Exemplo 1 - Triggers



Para esse exemplo vamos usar o BD do exercício 1 que estamos habituado. Vamos criar uma tabela chamada auditoria que através de um trigger vai gerar um histórico de alteração em alguma dessas tabelas. Podemos novamente usar a tabela de produtos.

Exemplo 1 - Triggers

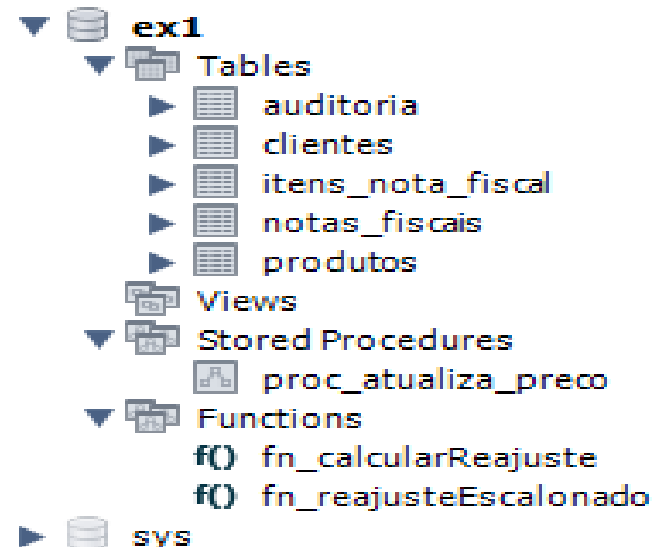
-- Criando uma nova tabela para demonstrar a ação do trigger

```
CREATE TABLE auditoria(  
  
discriminacao VARCHAR(50),  
  
p_unitario_antigo DECIMAL(10,2),  
  
p_unitario_novo DECIMAL(10,2),  
  
data_trigger DATE  
  
);
```

Crie uma nova tabela para poder lançar nela as ações da execução do trigger

SCHEMAS

Filter objects



Exemplo 1 - Triggers

Navigator: Schemas

Filter objects

ex1

- Tables
 - auditoria
 - clientes
 - items_nota_fiscal
 - notas_fiscais
 - produtos**
 - Columns
 - Indexes
 - Foreign Keys
 - Triggers
- Views
- Stored Procedures
 - proc_atualiza_preco
- Functions
 - fn_calcularReajuste
 - fn_reajusteEscalonado

Administration Schemas

Information

Table: **produtos**

Columns:

- id int AI PK
- discriminacao varchar(45)
- p_unitario decimal(10,2)

Query 1 produtos - Table

Table Name: produtos Schema: ex1

Charset/Collation: Default Charset Default Collation Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
discriminacao	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
p_unitario	DECIMAL(10,2)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name: Data Type:

Charset/Collation: Default

Comments:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns Indexes Foreign Keys **Triggers** Partitioning Options

Apply Revert

É possível criar pela QUERY direta, mas vamos usar a ferramenta, abra a tabela de produtos e vá em triggers

Exemplo 1 - Triggers

Query 1 produtos - Table x

Table Name: produtos Schema: ex1

Charset/Collation: Default Charset Default Collation Engine: InnoDB

Comments: Inserindo na tabela de histórico a mudança do preço unitário na tabela produtos, depois de um UPDATE

BEFORE INSERT
AFTER INSERT
BEFORE UPDATE
▼ AFTER UPDATE
 produtos_AFTER_UPDATE
BEFORE DELETE
AFTER DELETE

1 CREATE DEFINER = CURRENT_USER TRIGGER `ex1`.`produtos_AFTER_UPDATE` AFTER UPDATE ON `produtos` FOR EACH ROW
2 BEGIN
3
4 END

Exemplo 1 - Triggers

```
1 CREATE DEFINER='root'@'localhost' TRIGGER `produtos_AFTER_UPDATE` AFTER UPDATE ON `produtos` FOR EACH ROW
2 BEGIN
3     INSERT INTO auditoria -- INSERE NA TABELA AUDITORIA
4     VALUE (OLD.discriminacao , OLD.p_unitario, NEW.p_unitario, curdate());
5     -- OS VALORES ANTES DO UPDATE e OS VALORES DEPOIS DO UPDATE
6     -- Os valores antigos serão capturados pela palavra chave OLD
7     -- Os valores novos serão capturados pela palavra chave NEW
8 END
```

Digite o código a seguir e dê apply. Veja que foi inserido na tabela produtos esse trigger



Apply

Revert

Testando o exemplo 1

```
SELECT *FROM ex1.auditoria;
```



Result Grid



Filter Rows:

Export:



Wrap Cell Content:



discriminacao

p_unitario_antigo

p_unitario_novo

data_trigger

```
SELECT *FROM ex1.produtos;
```

Result Grid



Filter Rows:

	id	discriminacao	p_unitario
▶	1	Notebook	8318.75
	2	Laptop	9558.99
	3	MEMÓRIA RAM	1068.66
	4	Celular SAMSUNG	2778.59
●	NULL	NULL	NULL

Testando o exemplo 1

```
UPDATE produtos SET p_unitario = 8999.99 WHERE id = 1;
```

Result Grid			
Filter Rows:			
Edit:			
	id	discrimanacao	p_unitario
▶	1	Notebool	8999.99
	2	Laptop	9558.99
	3	MEMÓRIA RAM	1068.66
	4	Celular SAMSUNG	2778.59
•	NULL	NULL	NULL

```
SELECT *from auditoria;
```

Result Grid				
Filter Rows:				
Export:				
Wrap Cell Content:				
	discriminacao	p_unitario_antigo	p_unitario_novo	data_trigger
▶	Notebool	8318.76	8999.99	2023-10-30