


Objetivos da Aula:

- Triggers

8/11/2024

Trigger

- É um conjunto de operações que são executadas, automaticamente, **antes** ou **depois** da **ocorrência de um evento**, em uma tabela.
- **Ocorrências de evento**  **Comando disparador**

DML (update, insert ou delete)

Trigger

Exemplos de Utilização:

- Atualização do estoque.
- Implantação de regras de *negócios* (Ex. Envio de e-mail ao fornecedor para compra de produtos quando atingir a quant. *Mínima em estoque*)
- Criação de valores. (Ex. Ao cadastrar um novo funcionário realiza a soma do salários dos funcionários e atualiza na tabela departamento).
- Manutenção de um registro histórico das alterações.
- Implantação de segurança (auditoria).

Preparar o ambiente

Executar os scripts:

- funcionário.sql ;
- func_log.sql

Trigger - Sintaxe

```
CREATE TRIGGER nome_do_trigger
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON nome_da_tabela
FOR EACH ROW
BEGIN
    -- Corpo do trigger (código SQL)
END;
```

Exemplo 1 – Trigger histórico_salarial

9

```
DELIMITER $  
  
CREATE TRIGGER historico_salario AFTER UPDATE ON funcionario  
  
FOR EACH ROW  
  
> BEGIN  
  
> IF NEW.fun_sal > 1000 THEN  
    INSERT INTO func_log (fun_id,fun_data,novo_sal,msg) VALUES (NEW.fun_cod,NOW(),NEW.fun_sal, "Novo salário Superavaliado");  
- END IF;  
  
- END$  
  
DELIMITER ;
```

Trigger – Exemplo 1 (COMENTÁRIOS)

10

- Comentários sobre o exemplo anterior
 - A ativação do *gatilho ocorre* quando **comandos DML** são executados sobre a tabela de funcionários
 - Uso de **AFTER** possibilita o acesso a valores novos.
 - Uso de **FOR EACH ROW** gera várias execuções do gatilho
- Por exemplo, o comando DML abaixo:




```
UPDATE  
FUNCIONARIOSET  
FUN_SAL=FUN_SAL*1.1  
WHERE FUN_COD IN  
(1,2,3,4,5,6,7);
```



Ativa o gatilho
uma vez para
cada tupla
atualizada

Exibindo os triggers

14 • `show triggers;`

Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 								
	Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer
▶	historico_salario	INSERT	funcionario	BEGIN IF NEW.fun_sal > 1000 THEN INSERT...	AFTER	2023-10-29 22:03:26.60	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost


Ativando o Trigger histórico_salarial

UPDATE FUNCIONARIO

SET FUN_SAL=FUN_SAL*1.1 WHERE FUN_COD IN
(1,2,3,4,5,6);

22 • `SELECT * FROM FUNC_LOG;`

23

Result Grid				
Filter Rows: <input type="text"/>				
Export:  Wrap Cell Conte				
	fun_id	fun_data	novo_sal	msg
▶	1	2023-10-29	4400.00	Novo salário Superavaliado
	2	2023-10-29	1650.00	Novo salário Superavaliado
	3	2023-10-29	1100.00	Novo salário Superavaliado
	4	2023-10-29	5500.00	Novo salário Superavaliado
	5	2023-10-29	6600.00	Novo salário Superavaliado