JCAVI
TREINAMENTOS EM TI

Criando e trabalhando com FUNCTIONS.

As funções são utilizadas especificamente para retornar algo. Podemos passar algum parâmetro com o mesmo tipo de declaração que fazemos e informar o tipo de retorno que teremos. Se você quiser criar algo para ter algum retorno, utilizamos uma function, pois poderemos utilizá-las no meio de uma consulta, ao contrário da procedure, que temos que executar com um comando específico.

Foi criado uma Function que informa o ID do Cadastro e retorna o nome.

```
delimiter $$
create function retorna_nome(id_cad_input int)
returns varchar(50) deterministic
begin
	declare	nome varchar(50);
	select Nome_Cliente into nome
	from cadastro
	where id_cad = id_cad_input;
	return nome;
	end $$
delimiter ;
```

Utilizando a Function com um JOIN da Tabelas Produtos. Ordenando por nome de Produto e Cadastro.

```
select Nome, retorna_nome(id_cad),Quantidade from  produtos
order by	1,2;
```

JCAVI
TREINAMENTOS EM TI

Criando e Trabalhando com Stored Procedures.

As procedures ou procedimentos são scripts visando alguma modificação no banco de dados ou simplesmente uma busca de dados do mesmo.

As Stored Procedures são armazenadas e pré-compiladas diretamente no SGBD, isso torna sua execução mais rápida.

Criar Procedures de Select tabela Produtos

```
DELIMITER $$

CREATE PROCEDURE st_select()

 BEGIN

 SELECT * from produtos;

 END

 $$
```

Criar uma procedure para copiar os dados da tabela produtos para produtos2

Antes de iniciar vamos limpar os registros da tabela produtos2 com o comando truncate.

truncate produtos2

Criando a procedure

```
DELIMITER $$

CREATE PROCEDURE st_input_tabela_produtos2()

BEGIN

 DECLARE done INT DEFAULT FALSE;

DECLARE INSERE_ID_PROD int default 0;

DECLARE INSERE_NOME varchar(30) default 0;

 DECLARE INSERE_Valor float(10,2) default 0;

 DECLARE INSERE_Quantidade int default 0;

 DECLARE INSERE_ID_CAD int;

DECLARE cursor1 CURSOR FOR SELECT id_prod,Nome,Valor,Quantidade,id_cad from produtos;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

  OPEN cursor1;

 read_loop: LOOP

 IF done THEN

     LEAVE read_loop;

      END IF;
```

```
FETCH cursor1 INTO
INSERE_ID_PROD,INSERE_NOME,INSERE_VALOR,INSERE_QUANTIDADE,INSERE_ID_CAD;

insert into produtos2
Values(INSERE_ID_PROD,INSERE_NOME,INSERE_VALOR,INSERE_QUANTIDADE,INSERE_ID_CAD);

end loop;

 CLOSE cursor1;

END

$$
```

Para executar a procedure execute o comando **call.**

```
Call st_input_tabela_produtos2()
```

Automatizando a execução de uma Stored Procedure.

No Mysql o processo de Agendamento de procedures e script por padrão é desativado. NO SQL Server o serviço do SQL Agent também deve ser ativado. Como no Oracle também deve ser ativado.

Ativar o Scheduler.

```
set global event_scheduler = on;
```

Para criar um event scheduler chamando a Procedure criada anteriormente.

```
delimiter $$

create event chama_procedure_input ON SCHEDULE EVERY 1 minute

ON COMPLETION NOT PRESERVE DISABLE

do

call st_input_tabela_produtos2()
```

Pode ser agendado de forma personalizada seguindo alguns exemplos

on schedule every 1 year: uma vez por ano;
on schedule every 1 month: uma vez por mês;
on schedule every 1 day: uma vez ao dia;
on schedule every 1 hour: uma vez por hora;
on schedule every 1 minute: uma vez por minuto;
on schedule every 1 second: uma vez por segundo.

Para ativar um event scheduler. Utilizaremos o exemplo do criado

alter event chama_procedure_input enable;

Para excluir um event scheduler use o comando DROP.

drop event chama_procedure_input

Trabalhando com Triggers(Gatilhos)

A trigger é um conjunto de operações que são executadas automaticamente quando uma alteração é feita em um registro que está relacionado a uma tabela. Ela pode ser invocada antes ou depois de uma alteração em um insert , update ou delete ,podendo haver até 6 triggers por tabela.
Alguns benefícios de sua utilização são:

Verificar a integridade dos dados, pois é possível fazer uma verificação antes da inserção do registro;
Contornar erros na regra de negócio do sistema no banco de dados;
Utilizar como substituta para event_scheduler .
Entretanto, ela não o substitui em processos que não são disparados a partir de uma tabela;
Auditar as mudanças nas tabelas.

```
CREATE TABLE ItensVenda
(
   Venda      INT,
   Produto VARCHAR(50),
   Qt_Vendida  INT
);
```

Ao inserir e remover registro da tabela ItensVenda, o estoque do produto referenciado deve ser alterado na tabela Produtos. Para isso, serão criados dois triggers: um *AFTER INSERT* para dar baixa no estoque e um *AFTER DELETE* para fazer a devolução da quantidade do produto.

```
DELIMITER $

CREATE TRIGGER ItensVenda_Insert AFTER INSERT

ON ItensVenda

FOR EACH ROW

BEGIN

   UPDATE Produtos SET Quantidade = Quantidade - NEW.Qt_Vendida

WHERE id_prod = NEW.Produto;

END$

CREATE TRIGGER ItensVenda_Delete AFTER DELETE

ON ItensVenda

FOR EACH ROW

BEGIN

   UPDATE Produtos SET Quantidade = Quantidade + OLD.Qt_Vendida

WHERE id_prod = OLD.Produto;

END$

DELIMITER ;
```

Para testar as triggers vamos fazer um insert na tabela ItensVendas e um delete.

INSERT INTO ItensVenda VALUES (1,2,3)

DELETE FROM ItensVenda WHERE Venda = 1 AND Produto = '2';

Para exibir uma trigger

SHOW TRIGGERS;

Para desativar uma trigger e praticamente o mesmo conceito e comando de uma Procedure ou Function

alter trigger ItensVenda_Insert disable;***

Para excluir uma trigger utilize o comando DROP.

drop trigger ItensVenda_Insert ;

Podemos trabalhar as Condicionais com as Triggers

Vamos recriar nossa Trigger de Insert de Vendas com condicionais nos campos da tabela ItensVenda.

Para isso vamos adicionar um campo na nossa tabela de ItensVenda. O campo de status de venda.

alter table itensvenda add column status_venda varchar(50)

```
DELIMITER $

CREATE TRIGGER ItensVenda_Insert AFTER INSERT

ON ItensVenda

FOR EACH ROW

BEGIN

if new.status_venda = "vendeu" then

   UPDATE Produtos SET Quantidade = Quantidade - NEW.Qt_Vendida

WHERE id_prod = NEW.Produto;

end if;

IF NEW.status_venda = 'devolveu' THEN

 UPDATE produtos SET Quantidade = Quantidade + NEW.Qt_Vendida

 WHERE id_prod = NEW.Produto;

 END IF;

END$
```

Vamos fazer uns inserts para testar o status de venda e devolução do mesmo produto. E validar se a quantidade aumenta ou diminui da tabela produtos.

```
INSERT INTO ItensVenda VALUES (1,3,3,"vendeu");

INSERT INTO ItensVenda VALUES (1,3,3,"devolveu");
```