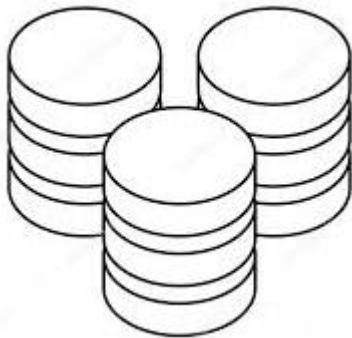


Banco de Dados II

Triggers

1ª parte



Profa. Damires Souza
damires@ifpb.edu.br

Constraints

```
create table Funcionario  
( . . . ,  
    Salario numeric(15,2) constraint check csal  
        check(Salario >= 5000),  
    DEPTNO integer constraint check cdeptno  
        check(DEPTNO between 10 and 100)  
);
```

**** Os tipos básicos de constraints provêem um mecanismo declarativo para associar condições “simples” a colunas**

Constraints e Triggers

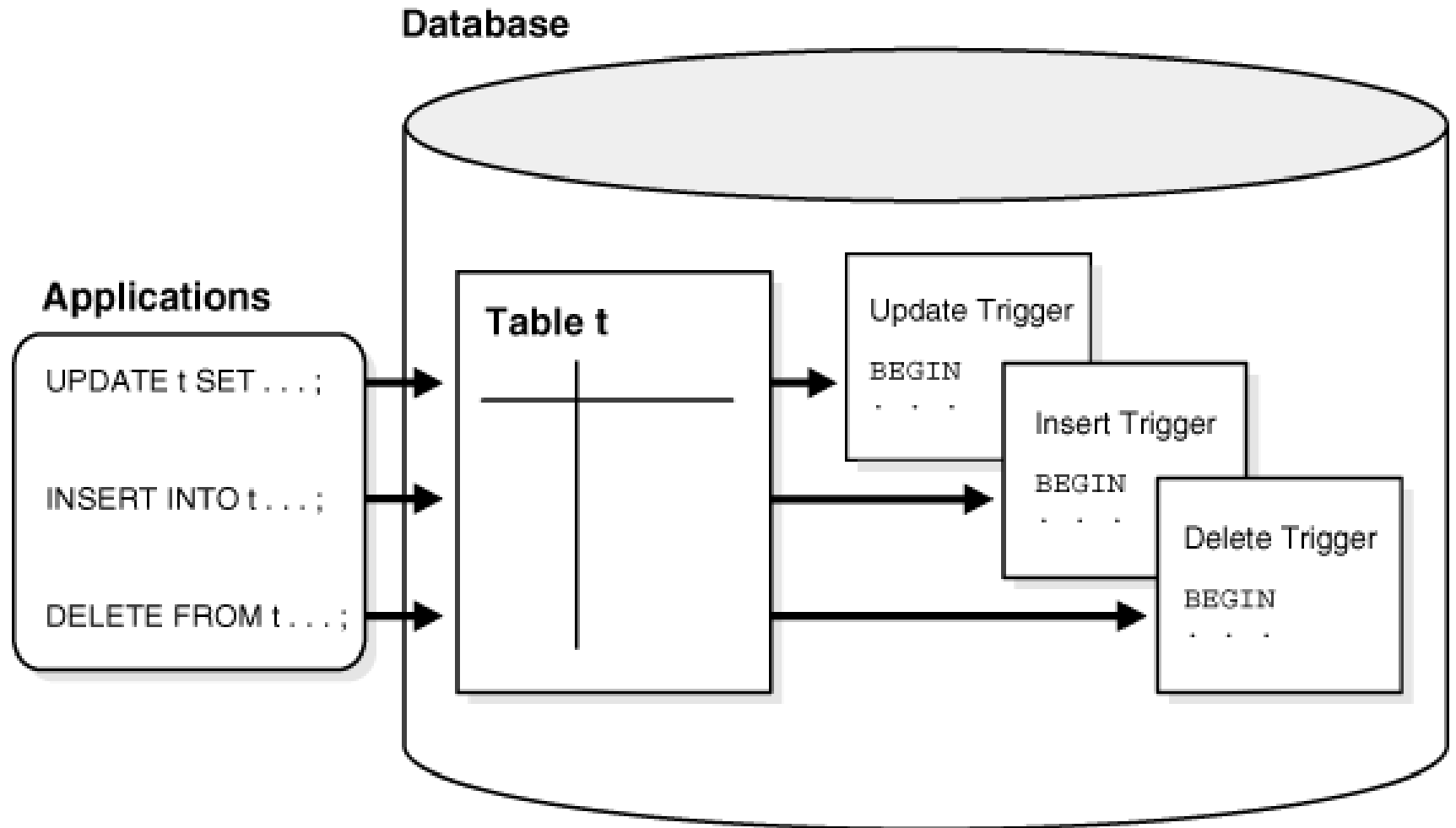
- **Constraints complexas** não podem ser especificadas dentro de definições de tabelas
 - Geralmente referem-se a **diversas tabelas e colunas**

Triggers provêem uma técnica **procedimental** para especificar e manter **restrições de integridade mais complexas**

TRIGGER

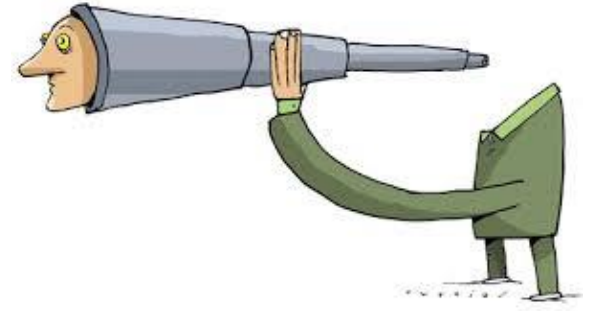


Como???



Trigger

- O **trigger** ou **gatilho** é uma programa normalmente PL/pgSQL armazenado no BD que é **executado imediatamente antes ou após um comando disparador**
- O comando disparador pode ser:
 - Comando **DML**
 - **INSERT, UPDATE ou DELETE**
 - Comando **DDL** (create, alter, drop, truncate)



EVENTO

CONDIÇÃO

AÇÃO

Exemplo

REORDER Trigger

```
AFTER UPDATE OF parts_on_hand ON inventory
```

Triggering Statement

Evento

```
WHEN (new.parts_on_hand < new.reorder_point)
```

Trigger Restriction

Condição

Triggered Action

```
FOR EACH ROW
DECLARE                                /* a dummy variable for counting */
    NUMBER X;
BEGIN
    SELECT COUNT(*) INTO X              /* query to find out if part has already been */
    FROM pending_orders                /* reordered-if yes, x=1, if no, x=0 */
    WHERE part_no=:new.part_no;

    IF x = 0
    THEN                                /* part has not been reordered yet, so reorder */
        INSERT INTO pending_orders
        VALUES (new.part_no, new.reorder_quantity, sysdate);
    END IF;                            /* part has already been reordered */
END;
```

Ação

Criando um trigger

CREATE TRIGGER name

{ **BEFORE** | **AFTER** | **INSTEAD OF** } { event [**OR** ...] }

ON table_name [FROM referenced_table_name]

[FOR [**EACH**] { **ROW** | **STATEMENT** }]

[WHEN (condition)]

EXECUTE PROCEDURE function_name (arguments)

onde evento pode ser:

**INSERT UPDATE [OF column_name [, ...]] DELETE
TRUNCATE**

Exemplo 1

```
CREATE TABLE empaudit (matemp integer NOT NULL,  
dataalter varchar NOT NULL );
```

```
CREATE OR REPLACE FUNCTION geralogemp()
```

```
RETURNS trigger AS $$
```

```
BEGIN
```

```
    INSERT INTO empaudit (matemp, dataalter)
```

```
    VALUES (new.matricula, current_timestamp);
```

```
    RETURN NEW;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```




Exemplo 1

```
CREATE TRIGGER logEmptrigger  
AFTER INSERT ON empregado FOR EACH ROW  
EXECUTE PROCEDURE geralogEmp();
```

```
select * from empregado order by matricula;
```

```
select * from empaudit;
```

```
INSERT INTO empregado VALUES (10,'Patricia', 'Novais','03-  
01-2022','Analista de Requisitos', 6000, 1, 2);
```

| Data Output | | Explain | Messages | Notifications |
|-------------------------------------------------------------------------------------|--------------------|--------------------------------|----------|---------------------------------------------------------------------------------------|
|  | matemp. integer | dataalter character varying | |  |
| 1 | 10 | 2022-10-19 11:09:44.197921-03 | | |

Algumas variáveis

- **NEW:** do tipo RECORD
 - Nova linha para operações INSERT / UPDATE
 - É NULL em gatilhos no nível de statement (instrução) e para operações DELETE
- **OLD:** do tipo RECORD
 - Mantém a linha antiga para operações UPDATE / DELETE
 - É NULL em gatilhos no nível de instrução e para operações INSERT
- **TG_WHEN:** do tipo text
 - Contém BEFORE, AFTER ou INSTEAD OF
- **TG_LEVEL:** do tipo text
 - Contém ROW ou STATEMENT
- **TG_OP:** do tipo text
 - Contém INSERT, UPDATE, DELETE ou TRUNCATE
- **TG_table_name:** nome da tabela

Exemplo

Update empregado

Set dataadmissão = '05-03-2022'

Where matricula = 3

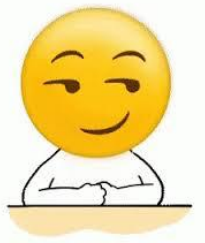
Old.dataadmissão = '11/02/2021'

New.dataadmissão = '05-03-2022'

Trigger

- A **ação** pode ser executada antes ou depois do evento
 - **BEFORE** Insert: Trigger dispara imediatamente **antes** do dado ser de fato **inserido** na tabela ...
 - **AFTER** Insert: Trigger dispara imediatamente **depois** do dado ser realmente **inserido** na tabela.
- Eventos de **update** podem ser limitados a um atributo particular ou a um conjunto de atributos
- Uma condição pode ser especificada por uma cláusula **WHEN**: a ação é executada somente se a regra for disparada e a condição for verdadeira

Exemplo 2



```
create table depto_backup as  
select * from departamento where 1= 2;
```

```
select * from departamento;  
select * from depto_backup;
```

```
CREATE OR REPLACE FUNCTION gerareplicaDepto()  
  RETURNS trigger AS $$  
  BEGIN  
    Insert into depto_backup values  
      (new.coddepto, new.nome, new.local);  
    RETURN NEW;  
  END;  
  $$ LANGUAGE plpgsql;
```

Exemplo 2

CREATE TRIGGER **replicaInsDepto**




After INSERT ON departamentoo




For each row Execute procedure gerarepDepto();

Insert into departamentoo values (3,'Marketing','Campina Grande');

select * from departamentoo;

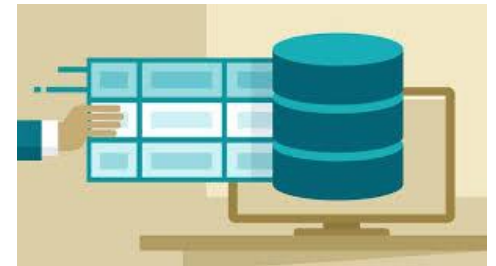
select * from depto_backup;

| Data Output | | Explain | Messages | Notifications |
|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|---------------|
|  | coddepto [PK] integer  | nome character varying (20)  | local character varying (20) | |
| 1 | 1 | Informática | Sede | |
| 2 | 2 | Pessoal | Sede | |
| 3 | 3 | Marketing | Campina Grande | |

| Data Output | | Explain | Messages | Notifications | |
|-------------------------------------------------------------------------------------|----------------------------|-------------------------------------------------------------------------------------|---------------------------------------|---------------------------------------------------------------------------------------|----------------------------------------|
|  | coddepto integer |  | nome character varying (20) |  | local character varying (20) |
| 1 | | 3 | Marketing | Campina Grande | |

Quando usar??

- Podem ser definidos **vários gatilhos para uma única tabela**
 - Criar o conteúdo de uma coluna **derivada** de outras colunas
 - Criar **mecanismos de validação** envolvendo múltiplas tabelas
 - Criar **logs** para registrar a utilização de uma tabela
 - **Atualizar outras tabelas** em função de inclusão ou alteração da tabela atual
 - Efetuar **atualizações a partir de views**



Como alterar ou remover??

```
ALTER TRIGGER name ON table_name  
    RENAME TO new_name;
```

```
ALTER TABLE nome  
    DISABLE TRIGGER [ trigger_name | ALL ]  
    ENABLE TRIGGER [ trigger_name | ALL ]
```

```
DROP TRIGGER nome on tabela;
```

