Funções

Vantagens:

↪ Agrupamentos de instruções
↪ Redução de rodadas entre cliente e servidor
↪ Redução da geração de dados intermediários
↪ Melhora na performance

Function vs Trigger:

↪ Triggers:

- São acionadas por operações de Insert, Update e Delete
- Triggers são implementadas através de funções

↪ Funções:

- Podem implementar triggers
- Retornam e manipulam dados

```
CREATE or REPLACE FUNCTION somefunc()
RETURNS void AS  $$
DECLARE
    quantity integer := 30;
BEGIN
    RAISE NOTICE 'Quantidade é %',
quantity;
END; $$
LANGUAGE plpgsql;
```

Tipos de variáveis

↪ **Integer**: -/+2147483648

↪ **Numeric** (precisão, escala): 12.1245, precisão: 6 Escala: 4

↪ **Varchar**: string

↪ **Record**:

- Acomoda a estrutura durante For/Loop
- Dinâmico
- NÃO é um tipo realmente

Exemplo da varável Record:

```
CREATE or replace FUNCTION
merge_fields() RETURNS text AS $$
DECLARE
    r record;
BEGIN
    SELECT * INTO r FROM dados where
id =1 ;
    RETURN r.name || r.salary;
 END;
$$ LANGUAGE plpgsql;
```

Parâmetros:

↪ São passados pela chamada da função
↪ Nome da variável Tipo da variável

```
CREATE FUNCTION selecionar(p_itemno
int)
RETURNS TABLE(name
varchar(50),  salary float) AS $$
BEGIN
    RETURN QUERY SELECT s.name,
s.salary FROM dados AS s WHERE s.id =
p_itemno;
END;
$$ LANGUAGE plpgsql;
```

Triggers

Utilização:

↪ Definição de regras de negócio
↪ Adição de funcionalidades ao banco
↪ Auditoria

Como ocorrem:

↪ Em uma tabela
↪ De acordo com uma operação de Insert, Update e Delete
↪ Antes ou depois da operação (Before / After)
↪ É uma função normal que tem retorno tipo Trigger

```
CREATE FUNCTION emp_time() RETURNS
trigger AS $$
    BEGIN
        NEW.last_date :=
current_timestamp;
        NEW.last_user := current_user;
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;
```

Identificadores de Correlação:

Instrução	Old	New
INSERT	NULL	Novos valores
UPDATE	Valores Anteriores	Novos valores
DELETE	Valores Anteriores	NULL

Variáveis implícitas:

↪ **TG_NAME:** tipo de dado NAME Variável que contém o nome da trigger disparada
↪ **TG_WHEN**: tipo de dado TEXT. Contém os valores BEFORE ou AFTER dependendo da definição da trigger e de como a trigger foi disparada
↪ **TG_LEVEL:** tipo de dado TEXT. Contém ROW ou STATEMENT dependendo do tipo declarado da trigger.
↪ **TG_OP:** tipo de dado TEXT. Contém INSERT, UPDATE ou DELETE indicando qual operação de atualização disparou a trigger.
↪ **TG_TABLE_NAME:** nome da tabela que disparou a trigger.

```
CREATE OR REPLACE FUNCTION
process_emp_audit() RETURNS TRIGGER AS
$$
    BEGIN
          IF (TG_OP = 'DELETE') THEN
            INSERT INTO emp_audit
values ('D', now(), current_user,
OLD.*);
            RETURN OLD;

        ELSIF (TG_OP = 'UPDATE') THEN
            INSERT INTO emp_audit
SELECT 'U', now(), current_user,
NEW.*;
            RETURN NEW;

        ELSIF (TG_OP = 'INSERT') THEN
            INSERT INTO emp_audit
values ('I', now(), current_user,
NEW.*);
            RETURN NEW;
        END IF;

        RETURN NULL;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER emp_audit
AFTER INSERT OR UPDATE OR DELETE ON
emp FOR EACH ROW EXECUTE PROCEDURE
process_emp_audit();
```