

### Trigger/Assertion 1.

- Evento: inserir ou realizar update na tabela **tem\_peca**.
- Condição: operação resulte em uma linha em que uma não máquina tenha uma peça (só máquinas podem ter peças) ou uma máquina tenha outra máquina.
- Ação: gere uma exceção (plpgsql) ou não insira ou não atualize (SQL padrão).

PS: No SQL padrão foi implementado com assertion pois é mais fácil assim, já no plpgsql do postgres, foi implementando com trigger, uma vez que não há suporte para assertions. Não foi feito o SQL padrão em plpgsql pois aquele não tem os recursos necessários para “parar” um insert ou update em certas condições.

### Query SQL padrão:

```
CREATE assertion consistencia_tem_peca CHECK
(
  NOT EXISTS
  (
    SELECT
      *
    FROM
      tem_peca
    WHERE
      fk_Produto_id_produto NOT IN
      (
        SELECT
          id_produto
        FROM
          produto p
        INNER JOIN tipo_produto tp
          ON p.fk_tipo_produto_id_tipo = tp.id_tipo
        WHERE
          nome_tipo = 'maquina'
      )
    OR
      fk_Produto_id_produto_ IN
      (
        SELECT
          id_produto
        FROM
          produto p
        INNER JOIN tipo_produto tp
          ON p.fk_tipo_produto_id_tipo = tp.id_tipo
        WHERE
          nome_tipo = 'maquina'
      )
  )
)
```

### Testes

Query 1:

```
SELECT * FROM tem_peca WHERE fk_produto_id_produto = 16 AND
fk_produto_id_produto_ = 1
```

Query 2:

```

SELECT
    p.id_produto, tp.nome_tipo
FROM
    produto p
INNER JOIN tipo_produto tp
ON p.fk_tipo_produto_id_tipo = tp.id_tipo
WHERE
    id_produto IN (1, 2, 16, 12)

```

Data Output	Explain	Messages	Notifications
fk_produto_id_produto [PK] integer		fk_produto_id_produto_ [PK] integer	qtd_pecas integer
1		16	1

Figura 2: Query 1

id_produto	nome_tipo
integer	character varying
1	processador
2	processador
3	placa mae
4	maquina

Figura 1: Query 2

Teste 1: tentando fazer um processador ter outro processador como peça.

```

1  update
2      tem_pecas
3  set
4      fk_produto_id_produto = 2
5  where
6      fk_produto_id_produto = 16 and fk_produto_id_produto_ = 1

```

Data Output	Explain	Messages	Notifications
ERROR: Só maquinas podem ter peças, máquinas não podem ter elas mesmas e maquinas não podem ser tidas por outras maquinas. CONTEXT: PL/pgSQL function consistencia_tem_pecas() line 37 at RAISE SQL state: P0001			

Teste 2: tentando fazer uma maquina ter um processador como peça.

```

1  update
2      tem_pecas
3  set
4      fk_produto_id_produto = 16
5  where
6      fk_produto_id_produto = 16 and fk_produto_id_produto_ = 1

```

Data Output	Explain	Messages	Notifications
UPDATE 1			
Query returned successfully in 122 msec.			

Teste 3: tentando fazer maquinas terem uma maquina como peça.

```

1  update
2      tem_pecas
3  set
4      fk_produto_id_produto_ = 16

```

Data Output	Explain	Messages	Notifications
ERROR: Só maquinas podem ter peças, máquinas não podem ter elas mesmas e maquinas não podem ser tidas por outras maquinas. CONTEXT: PL/pgSQL function consistencia_tem_pecas() line 37 at RAISE SQL state: P0001			

Teste 4: tentando fazer uma maquina ter uma placa mãe como peça.

```
1 update
2   tem_peca
3 set
4   fk_produto_id_produto_ = 12
5 where
6   fk_produto_id_produto = 16 and fk_produto_id_produto_ = 1
```

Data Output Explain Messages Notifications

UPDATE 1

Query returned successfully in 45 msec.

## Trigger 2.

- Evento: inserir um processador na tabela **Processador**.
- Condição: o processador não tem um id de série.
- Ação: cria uma nova série para o processador novo e atualiza a chave de serie para o processador.

## Query SQL padrão:

```
CREATE TRIGGER serie_padrao_processador after INSERT ON Processador
REFERENCING NEW ROW AS nrow
FOR each ROW
BEGIN atomic
  INSERT INTO
    Proc_serie
  SELECT
    p.nome_produto || ' Serie ' || pr.fk_Produto_id_produto
  FROM
    Processador pr
  INNER JOIN Produto p
    ON p.id_produto = pr.fk_Produto_id_produto
  WHERE
    NEW.fk_Proc_serie_id_proc_serie IS NULL
    AND
    pr.fk_Produto_id_produto = NEW.fk_Produto_id_produto;

  UPDATE
    Processador pr
  SET
    fk_Proc_serie_id_proc_serie = x.id_proc_serie
  FROM
    (
      SELECT
        *
      FROM
        Processador pr
      INNER JOIN Produto p
        ON p.id_produto = pr.fk_Produto_id_produto
      INNER JOIN Proc_serie ps
        ON ps.nome_proc_serie = (p.nome_produto || ' Serie ' ||
NEW.fk_Produto_id_produto)
    ) x
  WHERE
    x.fk_Produto_id_produto = pr.fk_Produto_id_produto;
END
```

## Testes

--Cria uma serie não padrão

```
INSERT INTO Proc_Serie VALUES  
( 'Serie predefinida RYZEN BERSERK');
```

--Insere um processador para essa série





```
INSERT INTO Processador VALUES  
(3599, 250.0, 5000, 3, 71);
```

--Insere dois processador sem série, que receberão duas novas séries padrão

```
INSERT INTO Processador VALUES  
(125, 3.3, 4, 1, NULL)  
, (200, 4.2, 8, 2, NULL);
```

## Resultados

```
14 select fk_produto_id_produto, fk_procserie_id_procserie, id_procserie, nome_procserie  
15 from Processador inner join Procserie on fk_Procserie_id_procserie = id_procserie  
16
```

	Data Output	Explain	Messages	Notifications
	 fk_produto_id_produto integer	 fk_procserie_id_procserie integer	 id_procserie integer	 nome_procserie character varying
1	3	71	71	Serie predefinida RYZEN BERSERK
2	1	72	72	i18 Serie 1
3	2	73	73	i1337x3 Serie 2