Trigger/Assertion 1.

- Evento: inserir ou realizar update na tabela **tem_peca**.
- Condição: operação resulte em uma linha em que uma não maquina tenha uma peça (só maquinas podem ter peças) ou um maquina tenha outra maquina.
- 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'
      )
   )
)
```

Descrição Textual Do SQL Padrão:

Criar uma asserção que checa se não existe na tabela que, seleciona todas as informações da tabela tem_peça onde a chave_estrangeira id produto não seja uma máquina (Subquery do Where). A segunda checarem (OR) é para verificar se não existe um produto dentro da tabela produto que não seja uma máquina.

Testes

Query 1:

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

Descrição textual Query 1:

Selecione todas as colunas da tabela tem_peca, onde o seu id seja igual a 16 e seu outro id seja igual a 1.

Resultado Query1

Data Output | Explain | Messages | Notifications

	fk_produto_id_produto [PK] integer	fk_produto_id_produto_ [PK] integer	qtd_peca integer
1	16	1	1

Figura 1: Query 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)
```

Descrição textual Query 2:

Mostre o id do produto e o seu nome da tabela produto, cruzando com a tabela tipo_produto por meio das ids dos tipos de produto, na qual o id esteja entre os valores (1,2,16,12.

Resultado Query 2

	id_produto integer	nome_tipo character varying
1	1	processador
2	2	processador
3	12	placa mae
4	16	maquina

Figura 1: Query 2

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

```
update
    tem_peca
set
    fk_produto_id_produto = 2
where
    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.

```
update
    tem_peca
set
    fk_produto_id_produto = 16
where
    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.

```
update
    tem_peca
set
    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.

```
update
    tem_peca
set
    fk_produto_id_produto_ = 12
where
    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
```

Descrição Textual Trigger 2

Crie uma trigger serie_padrao_processador depois de realizar um insert na tabela Processador, referenciando novas linhas como nrow.

Para cada linha, insira na tabela Proc_serie, uma lista com o nome do produto + a sua Serie + a sua id vinda da tabela processador, cruzado com a tabela produto para pegar as ids do produto e como condição, fazer essas ações quando há uma inserção de uma id de uma série nula e o id do processador for igual ao id no novo processador (Nesse caso, nulo).

Se isso ocorrer, fazer um update no processador, configurando a fk da id da serie do processador com a id da serie do processador da subquery que seleciona todos os atributos da tabela processador que cruza a tabela produto para obter o id do processador como produto e cruza com a tabela serie do processador para obter a serie. A condição para esse update acontece se a id do produto da subquery for igual ao id do produto na qual estamos tentando fazer essa ação.

Testes

```
--Cria uma série 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 processadores 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_proc_serie_id_proc_serie, id_proc_serie, nome_proc_serie
15  from Processador inner join Proc_serie on fk_Proc_serie_id_proc_serie = id_proc_serie
16
```

Data Output | Explain | Messages | Notifications

	fk_produto_id_produto integer	fk_proc_serie_id_proc_serie integer	id_proc_serie integer	nome_proc_serie character varying
1	3	71	71	Serie predefinida RYZEN BERSERK
2	1	72	72	i18 Serie 1
3	2	73	73	i1337x3 Serie 2