MAC439 – Laboratório de Bancos de Dados

Lista de Exercícios 13

Prazo: 10/07/2020

Considere o esquema relacional a seguir, que é parte do BD de uma loja virtual de livros:

```
Livro(idL, título, editora, preço, qtdeEmEstoque)
Pedido(idP, idC, dtPedido, preçoTotal, boletoPago)
ItemPedido(idP, idL, qtde)
```

Por meio do *site* da loja, os clientes fazem pedidos de compra de livros que ficam registrados na relação `Pedido`. Cada tupla de `Pedido` possui um número identificador do pedido, a identificação do cliente, a data do pedido, o preço total e um sinalizador do pagamento do pedido (que indica se o boleto já foi ou não pago).
Em um mesmo pedido, um cliente pode incluir vários livros. Os livros que compõem cada pedido e suas respectivas quantidades são registrados na relação `ItemPedido`.

Explique o que faz o conjunto de *triggers* (= regras ativas) a seguir, definidas sobre o esquema relacional acima. A resposta deve indicar também quando cada *trigger* é disparada e quando e como sua respectiva ação é executada.

```
CREATE OR REPLACE FUNCTION F1() RETURNS TRIGGER AS $$
DECLARE q INTEGER;
BEGIN
   q = (SELECT qtdeEmEstoque FROM Livro WHERE NEW.idL = idL);
   IF (SELECT boletoPago FROM PEDIDO WHERE idP = NEW.idP) = True OR q = 0 THEN
      RETURN NULL;
   ELSE
      IF q < NEW.qtde THEN
          NEW.qtde = q;
      END IF;
      UPDATE Livro SET qtdeEmEstoque = qtdeEmEstoque - NEW.qtde
          WHERE NEW.idL = idL;
      RETURN NEW;
   END IF;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER T1
BEFORE INSERT ON ItemPedido
FOR EACH ROW
EXECUTE PROCEDURE F1();

-----------------------------------------------------------
CREATE OR REPLACE FUNCTION F2() RETURNS TRIGGER AS $$
DECLARE q INTEGER;
BEGIN
   q = (SELECT qtdeEmEstoque FROM Livro WHERE NEW.idL = idL);
   IF (SELECT boletoPago FROM PEDIDO WHERE idP = NEW.idP) = True OR
      (NEW.qtde > OLD.qtde AND q = 0) THEN
      RETURN NULL;
```

```
   ELSE
      IF q < (NEW.qtde - OLD.qtde) THEN
          NEW.qtde = OLD.qtde + q;
      END IF;
      UPDATE Livro SET qtdeEmEstoque = qtdeEmEstoque + OLD.qtde - NEW.qtde
          WHERE NEW.idL = idL;
      RETURN NEW;
   END IF;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER T2
BEFORE UPDATE OF qtde ON ItemPedido
FOR EACH ROW
EXECUTE PROCEDURE F2();

-------------------------------------------------------------
CREATE OR REPLACE FUNCTION F3() RETURNS TRIGGER AS $$
DECLARE p NUMERIC(8,2);
BEGIN
   p = (SELECT preço FROM LIVRO WHERE idL = NEW.idL);
   UPDATE Pedido SET preçoTotal = preçoTotal + (p * NEW.qtde)
       WHERE idP = NEW.idP;
   RETURN NULL;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER T3
AFTER INSERT ON ItemPedido
FOR EACH ROW
EXECUTE PROCEDURE F3();

-------------------------------------------------------------
CREATE FUNCTION F4() RETURNS TRIGGER AS $$
DECLARE p NUMERIC(8,2);
BEGIN
   p = (SELECT preço FROM LIVRO WHERE idL = NEW.idL);
   UPDATE Pedido SET preçoTotal = preçoTotal + (p * (NEW.qtde - OLD.qtde))
       WHERE idP = NEW.idP;
   RETURN NULL;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER T4
AFTER UPDATE OF qtde ON ItemPedido
FOR EACH ROW
EXECUTE PROCEDURE F4();
```