Ricardo Bispo Amaral NUSP: 7693984
Pedro Toupitzen Specian NUSP: 8082640
Victor Caetano Da Silva NUSP: 9276999

Trabalho de LBD – 1º Semestre - 2017

Parte II – Artefato B

Regra de Negócio 1

Enunciado textual

Sempre que um funcionário for deletado, todos os seus dependentes também precisam ser deletados.

Solução textual em SQL Padrão

O gatilho que fará a regra de negócio enunciada acima vigorar consiste em verificar se a quantidade de dependentes cuja data de nascimento seja maior ou igual à de seu funcionário correspondente é igual a zero. O código para ela, em SQL padrão, segue:

```
CREATE OR REPLACE FUNCTION deleta_dependentes()
	BEGIN
		DELETE FROM DEPENDENTES WHERE id_funcionario =
OLD.id_funcionario
	END;


CREATE TRIGGER trigger_deleta_dependentes
	AFTER DELETE
	ON FUNCIONARIO
	FOR EACH ROW
	EXECUTE PROCEDURE deleta_dependentes();
```

Solução em código implementada dentro do PostgreSQL

```
CREATE OR REPLACE FUNCTION deleta_dependentes()
	BEGIN
		DELETE FROM DEPENDENTES WHERE id_funcionario =
OLD.id_funcionario
	END;


CREATE TRIGGER trigger_deleta_dependentes
	AFTER DELETE
```

```
      ON FUNCIONARIO
      FOR EACH ROW
EXECUTE PROCEDURE deleta_dependentes();
```

O código acima cria um gatilho chamado trigger_deleta_dependentes, que chama a função deleta_dependentes(). Essa função deleta todos os dependentes cujo id_funcionario forem igual ao do funcionário deletado.

Regra de Negócio 2

Enunciado textual

Sempre que um pedido for adicionado no sistema e for confirmado, a quantidade de pedidos confirmados que ele já fez precisa ser incrementada.

Solução textual em SQL Padrão

O gatilho para permitir que a regra de negócio enunciada acima vigore precisa incrementar a quantidade de pedidos de todos os clientes que tiverem feito um pedido e o mesmo seja confirmado. O código para o gatilho, em SQL padrão, segue:

```
CREATE OR REPLACE FUNCTION atualiza_quantidade()
      BEGIN
            IF OLD.confirmado = '0' AND NEW.confirmado = '1' THEN
                  UPDATE cliente SET qte_pedidos = OLD.qte_pedidos + 1
WHERE id_cli = NEW.id_cli;
                  RETURN void;
            END IF;
            IF OLD.confirmado = '1' AND NEW.confirmado = '0' THEN
                  UPDATE cliente SET qte_pedidos = OLD.qte_pedidos - 1
WHERE id_cli = NEW.id_cli;
                  RETURN void;
            END IF;
    END;


CREATE TRIGGER quantifica_pedido
      AFTER UPDATE
      ON pedido
      FOR EACH ROW
```

```
      EXECUTE PROCEDURE atualiza_quantidade();
```

Solução em código implementada dentro do PostgreSQL

```
CREATE OR REPLACE FUNCTION atualiza_quantidade()

      BEGIN

            IF OLD.confirmado = '0' AND NEW.confirmado = '1' THEN

                  UPDATE cliente SET qte_pedidos = OLD.qte_pedidos + 1
WHERE id_cli = NEW.id_cli;

                  RETURN void;

            END IF;

            IF OLD.confirmado = '1' AND NEW.confirmado = '0' THEN

                  UPDATE cliente SET qte_pedidos = OLD.qte_pedidos - 1
WHERE id_cli = NEW.id_cli;

                  RETURN void;

            END IF;

    END;


CREATE TRIGGER quantifica_pedido

      AFTER UPDATE

      ON pedido

      FOR EACH ROW

      EXECUTE PROCEDURE atualiza_quantidade();
```

O código acima cria a função atualiza_quantidade e o gatilho quantifica_pedido. O gatilho executa a função sempre que há uma atualização em um pedido, e a função então compara o valor antigo da coluna 'confirmado'. O motivo dessa comparação é o de que assim, ele só incrementará o valor da coluna qte_pedidos somente se o valor de confirmado foi alterado (era 0 e agora é 1). Isso impede que a quantidade de pedidos seja incrementada caso outra coluna do pedido seja alterada. Se foi detectado que o pedido estava não-confirmado e agora ele está confirmado, o valor de qte_pedidos da tupla que corresponde ao cliente que fez o pedido é incrementado.

Também é feita uma comparação para conferir se o pedido deixou de estar confirmado (era 1 e agora é 0), e se ela for verdadeira, o valor de qte_pedidos da tupla que corresponde ao cliente que fez o pedido é incrementado.