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 A

Regra de Negócio 1

Enunciado textual

Todo dependente que é filho de algum funcionário precisa ter a idade menor que a de seu funcionário correspondente.

Solução textual em SQL Padrão

A asserção que irá garantir a regra de negócio enunciada acima consiste 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 ASSERTION assertion_1
CHECK (0 = (
      SELECT COUNT(*) FROM DEPENDENTE AS d
      WHERE data_nascimento >= (
            SELECT data_nascimento
            FROM FUNCIONARIO AS f
            WHERE f.id_funcionario = d.id_funcionario
            )
      )
);
```

Solução em código implementada dentro do PostgreSQL

O PostgreSQL não dá suporte a asserções, portanto foi usada uma função para garantir que a regra de negócio especificada seja satisfeita. O código dessa função segue:

```
CREATE OR REPLACE FUNCTION assertion_1_a(id_fun VARCHAR(9), dat_nasc
DATE)
    BEGIN
        IF (
            (SELECT data_nascimento FROM FUNCIONARIO WHERE
id_funcionario = id_fun < dat_nasc)
```

```
            AND

            (SELECT PARENTESCO FROM DEPENDENTE WHERE id_funcionario =
id_fun) = 'filho(a)'

        )

            THEN

                  SELECT 'ERRO: A DATA DE NASCIMENTO DO DEPENDENTE É
MAIOR QUE A DE SEU PAI/MÃE' as Msg

                  RETURN FALSE

            ENDIF

            RETURN TRUE

    END;


CREATE OR REPLACE FUNCTION assertion_1_b(id_fun VARCHAR(9), dat_nasc
DATE)

    BEGIN

        IF(SELECT data_nascimento FROM DEPENDENTE WHERE
PARENTESCO='filho(a)' AND id_funcionario = id_fun < dat_nasc )

            THEN

                  SELECT 'ERRO: A DATA DE NASCIMENTO DO FUNCIONARIO É
MAIOR QUE A DE SEU(S) FILHO(S)' as Msg

                  RETURN FALSE

            ENDIF

            RETURN TRUE

    END;


CREATE TRIGGER trigger_assertion_1_a

  BEFORE INSERT OR UPDATE

  ON DEPENDENTE

  FOR EACH ROW

  EXECUTE PROCEDURE assertion_1_a(ROW.id_funcionario);


CREATE TRIGGER trigger_assertion_1_b

  BEFORE INSERT OR UPDATE

  ON FUNCIONARIO

  FOR EACH ROW
```

```
EXECUTE PROCEDURE assertion_1_b(ROW.id_funcionario);
```

O código acima cria duas funções e dois gatilhos. Assertion_1_a e trigger_assertion_1_a e garantem que quando um dependente é inserido ou alterado, a data de nascimento dele não pode ser menor (ou seja, mais velho) que a de seu funcionário correspondente caso ele seja filho do funcionário. Assertion_1_b e trigger_assertion_1_b garantem que quando um funcionário é inserido ou alterado, a data de nascimento dele não pode ser maior (ou seja, mais novo) que a de seus dependentes correspondentes que forem filhos dele. Em ambas as funções, é exibida uma mensagem de erro e é retornado FALSE, indicando que não foi possível realizar a inserção/alteração pois ela violaria a regra de negócio estabelecida.

Casos de Teste

Considerações Iniciais:

Considere o funcionário Antônio Faria Moreira, de id_funcionario = 65 e data_nascimento = ’01-07-1970’. Ele tem uma dependente chamada Ana Faria Moreira, de id_dependente = 60, id_funcionario = 65, parentesco = 'irmã(o)' e data_nascimento = '05-04-1979'

Caso de teste 1:

Esse caso testa a inserção de um dependente cuja data de nascimento maior (mais novo) que a de Antônio. Esse teste deve retornar TRUE, ou seja, deve permitir a inserção.

INSERT INTO DEPENDENTE (id_dependente, id_funcionario, nome, sobrenome, parentesco, data_nascimento) VALUES ('58', '65', 'Gabriel', 'Faria Moreira', 'filho(a)', to_date('21-11-2003', 'dd-mm-yyyy'));

Caso de teste 2:

Esse caso testa a alteração da data de nascimento da tupla inserida no caso de teste acima (id_dependente = 58) para que ela fique menor (mais velho) que a de Antônio. Esse teste deve retornar FALSE, ou seja, não deve permitir a inserção.

UPDATE DEPENDENTE SET data_nascimento = to_date('21-11-1913', 'dd-mm-yyyy') WHERE id_dependente = '58';

Caso de teste 3:

Esse caso testa a inserção de um dependente cuja data de nascimento menor (mais velho) que a de Antônio. Esse teste deve retornar FALSE, ou seja, não deve permitir a inserção.

INSERT INTO DEPENDENTE (id_dependente, id_funcionario, nome, sobrenome, parentesco, data_nascimento) VALUES ('59', '65', 'Lívia', 'Meirelles', 'filho(a)', to_date('14-01-1953', 'dd-mm-yyyy'));

Caso de teste 4:

Esse caso testa a alteração da tupla de Ana (id_dependente=60) para que sua data de nascimento seja menor (ou seja, mais velho) que a do registro de Antônio. Esse teste deve retornar TRUE, ou seja, deve ser permitida a inserção, afinal a regra de negócio estipulada se aplica apenas para dependentes filhos, e Ana é irmã de Antônio.

UPDATE DEPENDENTE SET data_nascimento= to_date('29-08-1965', 'dd-mm-yyyy') WHERE id_dependente='60';

Regra de Negócio 2

Enunciado textual

Nenhum ajudante pode ter o salário maior que o de quaisquer motoristas.

Solução textual em SQL Padrão

A asserção que irá garantir a regra de negócio enunciada acima consiste verificar se a quantidade de funcionários que são ajudantes e cujo salário é maior que o de algum funcionário que seja um motorista é igual a zero. O código para ela, em SQL padrão, segue:

```
CREATE ASSERTION assertion_2

CHECK (0 = (

      SELECT COUNT(*) FROM AJUDANTE

      INNER JOIN FUNCIONARIO ON FUNCIONARIO.id_funcionario =
AJUDANTE.id_funcionario

      WHERE FUNCIONARIO.salario > (

            SELECT salario FROM FUNCIONARIO AS f1

            INNER JOIN MOTORISTA ON MOTORISTA.id_funcionario =
f1.id_funcionario

            )

      )

);
```

Solução em código implementada dentro do PostgreSQL

O PostgreSQL não dá suporte a asserções, portanto foi usada uma função para garantir que a regra de negócio especificada seja satisfeita. O código dessa função segue:

```
CREATE OR REPLACE FUNCTION assertion_2(id_fun VARCHAR(9))

    BEGIN

        IF(
```

```
            (SELECT SALARIO FROM AJUDANTE

                  INNER JOIN FUNCIONARIO ON FUNCIONARIO.id_funcionario
= AJUDANTE.id_funcionario

                  WHERE AJUDANTE.id_funcionario = id_fun

                  >

                  SELECT SALARIO FROM MOTORISTA

                  INNER JOIN FUNCIONARIO ON FUNCIONARIO.id_funcionario
= MOTORISTA.id_funcionario)

      OR

      (SELECT SALARIO FROM MOTORISTA

                  INNER JOIN FUNCIONARIO ON FUNCIONARIO.id_funcionario
= AJUDANTE.id_funcionario

                  WHERE MOTORISTA.id_funcionario = id_fun

                  <

                  SELECT SALARIO FROM AJUDANTE

                  INNER JOIN FUNCIONARIO ON FUNCIONARIO.id_funcionario
= AJUDANTE.id_funcionario)



        )

            THEN

                  SELECT 'ERRO: NENHUM AJUDANTE PODE TER O SALARIO
MAIOR QUE O DE UM MOTORISTA' as Msg;

            RETURN FALSE;

            ENDIF;

      RETURN TRUE;

    END;

CREATE TRIGGER trigger_assertion_2

  BEFORE INSERT OR UPDATE

  ON FUNCIONARIO

  FOR EACH ROW

  EXECUTE PROCEDURE assertion_2(ROW.id_funcionario);
```

O código acima cria a função assertion_2_ajudante e o gatilho trigger_assertion_2. O gatilho é acionado sempre que há inserção ou alteração Sempre que há inserção ou alteração na tabela FUNCIONARIO, e então ele executa a função assertion_2.

A função assertion_2 faz o seguinte: Ela busca pelo salário de um ajudante com o id_funcionario especificado e confere se ele é maior que o salário de algum motorista, e depois busca pelo salário de um motorista com o id_funcionario especificado e confere se ele é menor que o salário de algum ajudante. Se o salário do ajudante inserido ou alterado é maior que o de algum motorista, ou se o salário do motorista inserido ou alterado é menor que o de algum ajudante, a função exibe uma mensagem de erro e retorna FALSE. Caso contrário, ele retorna TRUE e a linha é inserida/alterada normalmente.

Essa função realiza duas comparações para conferir se a regra de negócio está sendo satisfeita. O motivo pela qual ela faz isso é o seguinte: a primeira comparação é para o caso de um ajudante ter sido inserido ou alterado e seu salário é maior que o de pelo menos um funcionário motorista. Já a segunda comparação é para o caso de um motorista ter sido inserido ou alterado e seu salário é menor que o de algum funcionário ajudante. Se pelo menos uma das condições forem atendidas, a função exibe a mensagem de erro e retorna FALSE. Essa decisão foi tomada pois assim, é possível abranger os dois casos de violação da regra de negócio (ajudante com salário maior que o de motorista e motorista com salário menor que o de ajudante) com uma só função e um só gatilho, que é ativado com a inserção ou alteração de uma única tabela (FUNCIONARIO). Isso garante que apenas alterações na tabela de funcionários (onde é armazenado o salário) ative o gatilho, portanto alterações nas tabelas de ajudantes e funcionários não o ativam.

Casos de Teste

Considerações Iniciais:

Considere que o motorista com menor salário é Bernardo Oliveira, de id_funcionario=40 e salario=3000.00 e que o ajudante com maior salário é Paulo Amaral, de id_funcionario=45 e salario=1500.00.

Caso de teste 1:

Esse caso testa a inserção de um ajudante com salário menor que o de algum motorista. Esse teste deve retornar TRUE, ou seja, deve ser permitida a inserção.

INSERT INTO AJUDANTE (id_funcionario, tipo_ajudante) VALUES (50, 'Carregador');

INSERT INTO FUNCIONARIO (id_funcionario, nome, sobrenome, rua, numero, cidade, salario, data_admissao) VALUES ('50', 'José', 'Barbosa', 'Rua da Conceição', '279', 'Santos', 1240.00, to_date('26-08-1989', 'dd-mm-yyyy'));

Caso de teste 2:

Esse caso testa a inserção de um ajudante com salário maior que o de algum motorista. Esse teste deve retornar FALSE, ou seja, não deve ser permitida a inserção.

INSERT INTO AJUDANTE (id_funcionario, tipo_ajudante) VALUES ('51', 'Carregador');

INSERT INTO FUNCIONARIO (id_funcionario, nome, sobrenome, rua, numero, cidade, salario, data_admissao) VALUES ('51', 'Ronaldo', 'Antunes', 'Rua da Conceição', '279', 'Santos', 4500.00, to_date('26-08-1989', 'dd-mm-yyyy'));

Caso de teste 3:

Esse caso testa a inserção de um motorista com salário menor que o de algum ajudante. Esse teste deve retornar FALSE, ou seja, não deve ser permitida a inserção.

INSERT INTO MOTORISTA (id_funcionario, habilitacao_num, habilitacao_categoria) VALUES ('52', '12345678', 'B');

INSERT INTO FUNCIONARIO (id_funcionario, nome, sobrenome, rua, numero, cidade, salario, data_admissao) VALUES ('52', 'Brenda', 'Siqueira', 'Rua da Conceição', '279', 'Santos', 1400.00, to_date('26-08-1989', 'dd-mm-yyyy'));

Caso de teste 4:

Esse caso testa a alteração de um motorista (no caso, Bernardo Oliveira, cujo id_funcionário é 40) para que seu salário fique menor que o de algum ajudante. Esse teste deve retornar FALSE, ou seja, não deve ser permitida a alteração.

ALTER TABLE FUNCIONARIO SET salario=1400,00 WHERE id_funcionario='40'

Caso de teste 5:

Esse caso testa a alteração de um ajudante (no caso, Paulo Amaral, cujo id_funcionário é 45) para que seu salário continue menor que 3000,00, que é o salário de Bernardo. Esse teste deve retornar TRUE, ou seja, deve ser permitida a alteração.

ALTER TABLE FUNCIONARIO SET salario=2500,00 WHERE id_funcionario='45'

Caso de teste 6:

Esse caso testa a alteração de um ajudante (no caso, Paulo Amaral, cujo id_funcionário é 45) para que seu salário continue fique igual a 3000,00, que é o salário de Bernardo. Esse teste deve retornar TRUE, ou seja, deve ser permitida a alteração, pois a regra de negócio permite que um motorista tenha o mesmo salário que o de um ajudante.

ALTER TABLE FUNCIONARIO SET salario=3000,00 WHERE id_funcionario='45'