Sistemas de Informação e Bases de Dados

1º Semestre 2016/2017

Projeto

Parte II

Grupo 21

João Girão, nº 78761

Luís Rei, nº 78486

Ricardo Carvalho, nº 87873

Novembro 2016

Instituto Superior Técnico
Departamento de Engenharia Electrotécnica e de Computadores

QUERY

Depois de preenchermos as tabelas com valores teste confirmámos o bom funcionamento do *query* (figuras 1 e 2).

Todo o código feito ao longo desta parte do projeto encontra-se em anexo.

```
1  SELECT name
2  FROM doctor natural join study
3
4  /*  Estudo tem que conter 'X-ray' na descrição. */
5  WHERE description like '%X-ray%'
6
7  /*  Condição que verifica que os estudos não foram feitos
8  há mais de sete dias. Implica que podem ter sido feitos
9  à precisamente sete dias. */
10 AND datediff(current_date, study_date) <= 7
11
12 /*  Condição que garante que a "Philips" foi o fabricante
13 da máquina com a qual foi feito o study. */
14 AND manufacturer = 'Philips'
15
16 /* Seleção dos médicos com mais estudos feitos */
17 GROUP BY doctor_id
18 HAVING count(request_number) >= ALL (SELECT count(request_number)
19                                      FROM doctor);
```

Figura 1 - Código do *query* comentado

```
+---------+
| name    |
+---------+
| Antonio |
+---------+
1 row in set (0.00 sec)
```

Figura 2 – Resultado do *query*

TRIGGER

Na figura 3 apresenta-se o código usado para a geração do *trigger* e na figura 4 o código que permite testar cada uma das condições de erro. Na figura 5 encontram-se os resultados obtidos para o caso do médico que fez o *request* ser o mesmo do que faz o *study*, e na figura 6 os resultados quando a data do *study* é anterior ou igual à data do pedido.

```
DELIMITER $$

/* Trigger é verificado cada vez se tenta inserir uma nova entrada na tabela study. */
CREATE TRIGGER deny_doctor BEFORE INSERT ON study
FOR EACH ROW
BEGIN
    declare medico_request varchar(255);
    declare data_consulta date;

    SELECT doctor_id into medico_request FROM request WHERE request_number = new.request_number;

    /*  'App_date' é a data da consulta. Escolhemos alterar o nome inicialmente declarado
        no enunciado do projeto por uma questão de clarificação. */
    SELECT app_date into data_consulta FROM request WHERE request.request_number = new.request_number;

    /*  Condição que garante que o médico que faz o study e o request não são a mesma pessoa. */
    IF new.doctor_id = medico_request THEN
        /*  Procedure que não existe e que força erro no programa.
            Nome esclarecedor para o utilizador entender o que aconteceu. */
        CALL same_doctor_in_request_study();


    /*  Condição que garante que a data do study é posterior à da appointment. */
    ELSEIF new.date <= data_consulta THEN
            /* Novo procedure que não existe. */
            CALL study_date_must_be_posterior_to_appointment_date();
    END IF;
END $$

DELIMITER ;
```

Figura 3 – Código do *trigger* comentado

```
/* CONDIÇAO MEDICO IGUAL NO REQUEST E STUDY */
insert into appointment values ('ER2', 'C2', '2016-11-10', 1001);
insert into request values (00000009, 'ER2', 'C2', '2016-11-10');
insert into study values (00000009, 'X-ray' , '2016-11-11', 'C2', 'Gilletteis' , 45674567);

/* CONDIÇAO DATA DO STUDY ANTERIOR A DO REQUEST */
insert into appointment values ('NOR2', 'T1', '2016-11-13', 1001);
insert into request values (00000011, 'NOR2', 'T1', '2016-11-13');
insert into study values (00000011, 'X-ray' , '2016-11-11', 'C2', 'Gilletteis' , 45674567);
```

Figura 4 – Condições de teste do *trigger*

```
ysql> insert into appointment values('ER2', 'C2', '2016-11-10', 1001);
uery OK, 1 row affected (0.01 sec)

ysql> insert into request values(00000009, 'ER2', 'C2', '2016-11-10');
uery OK, 1 row affected (0.02 sec)

ysql> insert into study values(00000009, 'X-ray', '2016-11-11', 'C2', 'Gilletteis', 45674567);
RROR 1305 (42000): PROCEDURE ist178486.same_doctor_in_request_study does not exist
```

Figura 5 – Resultados da situação I do *trigger*

```
ysql> insert into appointment values('NOR2', 'T1', '2016-11-13', 1001);
uery OK, 1 row affected (0.01 sec)

ysql> insert into request values(00000011, 'NOR2', 'T1', '2016-11-13');
uery OK, 1 row affected (0.01 sec)

ysql> insert into study values(00000011, 'X-ray', '2016-11-11', 'C2', 'Gilletteis', 45674567);
RROR 1305 (42000): PROCEDURE ist178486.study_date_must_be_posterior_to_appointment_date does not exist
```

Figura 6 – Resultados da situação II do *trigger*

FUNCTION

```
Delimiter $$
/*  Função que verifica se uma região de um elemento de uma dada série A se sobrepõe com
    uma região de um elemento de outra dada série B.  */
CREATE FUNCTION SOBREPOSICAO (series_A varchar(255), series_B varchar(255))
returns Boolean
BEGIN
    /*  Declaração de variáveis auxiliares. Aqui são declaradas as variáveis
        que assumem os valores das coordenadas das regiões a serem estudadas. */
    declare Ax1 int default 0;
    declare Ax2 int default 0;
    declare Ay1 int default 0;
    declare Ay2 int default 0;
    declare Bx1 int default 0;
    declare Bx2 int default 0;
    declare By1 int default 0;
    declare By2 int default 0;

    declare fim int default 0;  /* Variável que indica fim do ciclo */
    declare elem_reg_A int; /* Variável que guarda o atual elemento da série A */
    declare elem_reg_B int; /* Variável que guarda o atual elemento da série B */
    declare serie_reg_A int; /* Variáveis com a coluna 'series_id' da região a comparar - série A */
    declare serie_reg_B int; /* Variáveis com a coluna 'series_id' da região a comparar - série B */

    /*  Declaração de ponteiros que guardam valores dos elementos da série A e B a percorrer. */
    declare regiao_A cursor for select * from region where series_id = series_A;
    declare regiao_B cursor for select * from region where series_id = series_B;

    /*  Declaração de um 'handler' que assinala o fim do ciclo (condição que se deve verificar
        quando não há mais elementos na série). */
    declare continue handler for not found set fim = 1;

    open regiao_A;
    open regiao_B;

    /*  Loop que percorre os elementos das séries A e B e faz a comparação entre as zonas de interesse
        de cada um. Se em algum caso houver sobreposição das regiões, é retornado 'true'. Se se chegar
        ao fim do ciclo sem nenhuma interrupção, retorna-se 'false'.
        Assumimos que cada elemento só tem uma zona de interesse. */
    percorre_A: loop

        /* Vamos buscar uma zona de interesse de um elemento da série A. */
        fetch regiao_A into serie_reg_A, elem_reg_A, Ax1, Ax2, Ay1, Ay2;

        /*  Condição que verifica se há mais regiões da série A a comparar; se não houverem
            quebra-se o ciclo e para-se a comparação. */
        IF fim then leave percorre_A; end IF;

        percorre_B: loop
            /* Semelhante ao ciclo anterior. */
            fetch regiao_B into serie_reg_B, elem_reg_B, Bx1, Bx2, By1, By2;
            IF fim then leave percorre_B; end IF;

            /*  Se um vértice de um quadrado se encontra dentro da área de um outro quadrado então podemos concluir que estes estão sobrepostos.
                Na primeira condição verificamos se um vértice de B está dentro de A e na segunda o oposto.
                Considera-se uma condição de 'overlap' válida a sobreposição das arestas das regiões das duas séries.
                Assumimos que, tal como na primeira parte do projeto, as coordenadas pertencem ao primeiro quadrante do mapa cartesiano */
            IF (( ((Ax1-Bx1)*(Ax2-Bx1))<=0 || ((Ax1-Bx2)*(Ax2-Bx2))<=0 ) && ( ((Ay1-By1)*(Ay2-By1))<=0 || ((Ay1-By2)*(Ay2-By2))<=0 )) THEN
                return true;
            ELSEIF (( ((Bx1-Ax1)*(Bx2-Ax1))<=0 || ((Bx1-Ax2)*(Bx2-Ax2))<=0 ) && ( ((By1-Ay1)*(By2-Ay1))<=0 || ((By1-Ay2)*(By2-Ay2))<=0 )) THEN
                return true;
            END IF;

        end loop;
    end loop;
    /* Fecham-se os ponteiros. */
    close regiao_A;
    close regiao_B;
    /* Não foi encontrada nenhuma zona de sobreposição. */
    return false;
END $$
Delimiter ;
```

Figura 7 – Códido da função gerada comentado

ANEXOS – CÓDIGO

TESTE

```
/* DATABASE */
/* USE ist178486 */

/* Limpeza das tabelas */
SET FOREIGN_KEY_CHECKS = 0; /* Disable foreign key checking.*/
TRUNCATE TABLE region;
TRUNCATE TABLE element;
TRUNCATE TABLE series;
TRUNCATE TABLE study;
TRUNCATE TABLE equipment;
TRUNCATE TABLE request;
TRUNCATE TABLE appointment;
TRUNCATE TABLE patient;
TRUNCATE TABLE doctor;
SET FOREIGN_KEY_CHECKS = 1; /* Enable foreign key checking. */


/* INSERCAO NAS TABELAS */

/* Tabela patient */
insert into patient values ('ER1', 'Luis', '1995-09-12', 'Rua 1');
insert into patient values ('ER2', 'Ricardo', '1990-02-25' ,'Rua 2');
insert into patient values ('NOR1', 'Joao',  '1999-11-09' ,'Rua 3');
insert into patient values ('NOR2', 'Joana',  '1999-05-17' ,'Rua 4');

/* Tabela doctor */
insert into doctor values ('C1', 'Antonio', 'Clinical specialist');
insert into doctor values ('C2', 'Manuel', 'Clinical specialist');
insert into doctor values ('T1', 'Maria', 'Technical specialist');
insert into doctor values ('T2', 'Rodrigo', 'Technical specialist');
insert into doctor values ('T3', 'Alice', 'Technical specialist');

/* Tabela appointment */
insert into appointment values ('ER1', 'C1', '2016-11-10', 1000);
insert into appointment values ('ER2', 'C1', '2016-11-10', 1001);
insert into appointment values ('NOR2', 'C2', '2016-11-11', 1001);
insert into appointment values ('NOR1', 'T1', '2016-11-12', 1003);
insert into appointment values ('ER1', 'T2', '2016-11-12', 1004);
insert into appointment values ('NOR2', 'T3', '2016-11-13', 1005);
```

```
/* Tabela request */
insert into request values (00000000, 'ER1', 'C1', '2016-11-10');
insert into request values (00000001, 'ER2', 'C1', '2016-11-10');
insert into request values (00000002 , 'NOR2', 'C2', '2016-11-11');
insert into request values (00000003 , 'NOR1', 'T1', '2016-11-12');
insert into request values (00000004 , 'ER1', 'T2', '2016-11-12');

/* Tabela equipment */
insert into equipment values ( 'Philips', 11111111, 'umModelo');
insert into equipment values ( 'Philips', 22222222, 'doisModelos' );
insert into equipment values ( 'Philips', 33333333, 'tresModelos');
insert into equipment values ( 'Gilletteis', 45674567, 'quatroModelos' );
insert into equipment values ( 'SIBDMachine' , 12341234, 'cincoModelos' );

/* Tabela study */
insert into study values ( 00000000, 'Outra Coisa', '2016-11-20' , 'C1' , 'Philips', 11111111);
insert into study values ( 00000002, 'Outra coisa', '2016-11-20' , 'C2', 'Philips', 33333333);
insert into study values ( 00000002, 'X-ray' , '2016-11-21', 'C1', 'Philips', 22222222);
insert into study values ( 00000003, 'Outra Coisa', '2016-11-21', 'T1' , 'SIBDMachine', 12341234);
insert into study values ( 00000004, 'X-ray' , '2016-11-23', 'T2' , 'Gilletteis' , 45674567);

/* Tabela series */
insert into series values ('S1', 'Series A' , 'umURL' , 00000002 , 'Outra coisa');
insert into series values ('S2', 'Series B' , 'doisURLs' , 00000004 , 'X-ray');
insert into series values ('S3', 'Series C' , 'tresURLs' , 00000002 , 'X-ray');

/* Tabela element */
insert into element values ('S1', 00000001);
insert into element values ('S2', 00000001);
insert into element values ('S2', 00000002);
insert into element values ('S3', 00000001);
insert into element values ('S3', 00000002);
insert into element values ('S3', 00000003);

/* Tabela region */
insert into region values ('S1', 00000001 , 0.0 , 0.0 , 0.2 , 0.2);
insert into region values ('S2', 00000001 , 0.0 , 0.0 , 0.2 , 0.2);
insert into region values ('S2', 00000001 , 0.5 , 0.5 , 0.7 , 0.7);
insert into region values ('S2', 00000002 , 0.2 , 0.2 , 0.4 , 0.0);
insert into region values ('S3', 00000001 , 0.5 , 0.5 , 0.9 , 0.9);
insert into region values ('S3', 00000002 , 0.4 , 0.4 , 0.5 , 0.5);
```

```
/* TESTE DO TRIGGER */

/* CONDIÇAO MEDICO IGUAL NO REQUEST E STUDY */
insert into appointment values ('ER2', 'C2', '2016-11-10', 1001);
insert into request values (00000009, 'ER2', 'C2', '2016-11-10');
insert into study values (00000009, 'X-ray' , '2016-11-11', 'C2', 'Gilletteis' , 45674567);

/* CONDIÇAO DATA DO STUDY ANTERIOR A DO REQUEST */
insert into appointment values ('NOR2', 'T1', '2016-11-13', 1001);
insert into request values (00000011, 'NOR2', 'T1', '2016-11-13');
insert into study values (00000011, 'X-ray' , '2016-11-11', 'C2', 'Gilletteis' , 45674567);
```

QUERY

```
SELECT name
FROM doctor natural join study
WHERE description like '%X-ray%'
AND datediff(current_date, study_date) <= 7
AND manufacturer = 'Philips'
GROUP BY doctor_id
HAVING count(request_number) >= ALL (SELECT count(request_number) FROM doctor);
```

TRIGGER

```
DELIMITER $$

CREATE TRIGGER deny_doctor BEFORE INSERT ON study
FOR EACH ROW
BEGIN
	declare medico_request varchar(255);
	declare data_consulta date;

	SELECT doctor_id into medico_request FROM request WHERE request_number = new.request_number;
	SELECT app_date into data_consulta FROM request WHERE request.request_number = new.request_number;

	IF new.doctor_id = medico_request THEN
		CALL same_doctor_in_request_study();


	ELSEIF new.date <= data_consulta THEN
			CALL study_date_must_be_posterior_to_appointment_date();
	END IF;
END $$

DELIMITER ;
```

FUNCTION

```
Delimiter $$

CREATE FUNCTION SOBREPOSICAO (series_A varchar(255), series_B varchar(255))
returns Boolean
BEGIN
	declare Ax1 int default 0;
	declare Ax2 int default 0;
	declare Ay1 int default 0;
	declare Ay2 int default 0;
	declare Bx1 int default 0;
	declare Bx2 int default 0;
	declare By1 int default 0;
	declare By2 int default 0;
	declare fim int default 0;
	declare elem_reg_A int;
	declare elem_reg_B int;
	declare serie_reg_A int;
	declare serie_reg_B int;
	declare regiao_A cursor for select * from region where series_id = series_A;
	declare regiao_B cursor for select * from region where series_id = series_B;
	declare continue handler for not found set fim = 1;

	open regiao_A;
	open regiao_B;

	percorre_A: loop
		fetch regiao_A into serie_reg_A, elem_reg_A, Ax1, Ax2, Ay1, Ay2;
		IF fim then leave percorre_A; end IF;

		percorre_B: loop
			fetch regiao_B into serie_reg_B, elem_reg_B, Bx1, Bx2, By1, By2;
			IF fim then leave percorre_B; end IF;

			IF ((((Ax1-Bx1)*(Ax2-Bx1))<=0 || ((Ax1-Bx2)*(Ax2-Bx2))<=0) && (((Ay1-
	By1)*(Ay2-By1))<=0 || ((Ay1-By2)*(Ay2-By2))<=0)) THEN
				return true;
			ELSEIF ((((Bx1-Ax1)*(Bx2-Ax1))<=0 || ((Bx1-Ax2)*(Bx2-Ax2))<=0) && (
	((By1-Ay1)*(By2-Ay1))<=0 || ((By1-Ay2)*(By2-Ay2))<=0)) THEN
				return true;
			END IF;

		end loop;
	end loop;

	close regiao_A;
	close regiao_B;

	return false;

END $$
Delimiter ;
```