САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
Институт компьютерных наук и технологий
Кафедра Компьютерных Систем и Программных Технологий

Отчет по дисциплине
«Базы данных»

SQL-программирование: Триггеры, вызовы процедур.

Работу выполнил студент группы №: 43501/3 Крутихин М.В.
Работу принял преподаватель: _____ Мяснов А.В.

Санкт-Петербург
2016 г.

Программа работы

1. Создать два триггера: один триггер для автоматического заполнения ключевого поля, второй триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице
2. Создать триггер в соответствии с индивидуальным заданием, полученным у преподавателя
3. Создать триггер в соответствии с индивидуальным заданием, вызывающий хранимую процедуру
4. Выложить скрипт с созданными сущностями в svn
5. Продемонстрировать результаты преподавателю

Ход работы.

1. Создание триггера для автоматического заполнения ключевого поля

Был создан генератор STORAGE_GEN:

```
CREATE sequence STORAGE_GEN;
```

Далее был создан триггер, использующий данный генератор:

```
create trigger storage_id_autoinc for DOCTORS
active before insert position 0
as
declare variable tmp DECIMAL (18 ,0);
begin
   if( new."DOCTOR ID" is null ) then
      new."DOCTOR ID" = gen_id ( STORAGE_GEN ,1);
   else
   begin
      tmp= gen_id ( STORAGE_GEN ,0);
      if(tmp <new."DOCTOR ID" ) then
         tmp= gen_id ( STORAGE_GEN , new."DOCTOR ID" -tmp);
   end
end
```

Вставка данных в таблицу:

```
Insert into DOCTORS ("DOCTOR ID", "DEPATMENT ID", "POSITION",
"PHONE NUMBER", FAMILY, NAME, SURNAME)
values (100000, 1, 'Травматолог', 84994031086,     'Иванов', 'Гле', 'Богданович');
```

```
Insert into DOCTORS ("DEPATMENT ID", "POSITION", "PHONE NUMBER",
FAMILY, NAME, SURNAME)
values (1, 'Травматолог', 84994031086,     'Иванов', 'Гле', 'Богданович');

Insert into DOCTORS ("DOCTOR ID", "DEPATMENT ID", "POSITION",
"PHONE NUMBER", FAMILY, NAME, SURNAME)
values (100010, 5, 'Гинеколог', 84994031086,  'Волченко', 'Павел',
'Владимирович');

Insert into DOCTORS ("DEPATMENT ID", "POSITION", "PHONE NUMBER",
FAMILY, NAME, SURNAME)
values (7, 'Хирург', 84994031086,  'Касов', 'Игнат', 'Тимофеевич');
```

Результат:

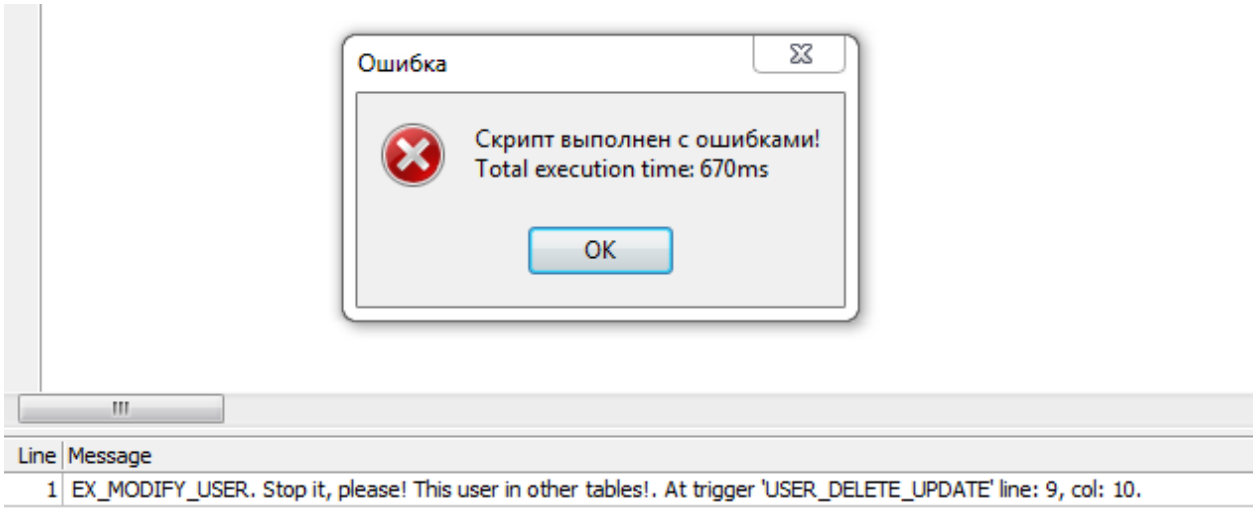
100 000	1	Травматолог	Иванов	84994031086	Гле	Богданович
100 001	1	Травматолог	Иванов	84994031086	Гле	Богданович
100 010	5	Гинеколог	Волченко	84994031086	Павел	Владимирович
100 011	7	Хирург	Касов	84994031086	Игнат	Тимофеевич

2. Создание триггера для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице.

Триггер:

```
create exception ex_modify_user 'Stop it, please! This user in other tables!';
create trigger user_delete_update for DEPARTMENTS
before delete or update
as
begin
   if( old."DEPARTMENT ID" in
   (select "DEPATMENT ID" from DOCTORS
   union
   select "DEPARTMENT ID" from PATIENTS))
   then exception ex_modify_user ;
end
```

При попытке удалить строчку в DEPARTMENTS, ключ которой содержится в подчиненных таблица, произойдет ошибка и будет выведено исключение:



3. Написать триггер, который не дает назначить пациенту процедуру, если она не соответствует диагнозу (выбрасывает в этом случае исключение)

Перед написанием данного триггера, была добавлена таблица, задающая для каждого диагноза набор допустимых процедур:

```
CREATE TABLE PROCEDURES_DIAGNOSIS (
    DIAGNOSIS_TITLE  VARCHAR(300) NOT NULL,
    PROCEDURE_TITLE  VARCHAR(300) NOT NULL
);

ALTER TABLE PROCEDURES_DIAGNOSIS ADD CONSTRAINT
FK_PROCEDURES_DIAGNOSIS_1 FOREIGN KEY (DIAGNOSIS_TITLE)
REFERENCES "DIAGNOSIS GUIDE" (TITLE);
ALTER TABLE PROCEDURES_DIAGNOSIS ADD CONSTRAINT
FK_PROCEDURES_DIAGNOSIS_2 FOREIGN KEY (PROCEDURE_TITLE)
REFERENCES "PROCEDURES GUIDE" ("TITLE PROCEDURE");
```

Так же была написана вспомогательная процедура, для проверки соответствия назначенной процедуры диагнозу:

```
Create procedure HELP1 (
help_proc integer,
help_diag integer)
returns (counth integer)
as
/*Вспомогательные переменные*/
declare variable help_id1 integer;
```

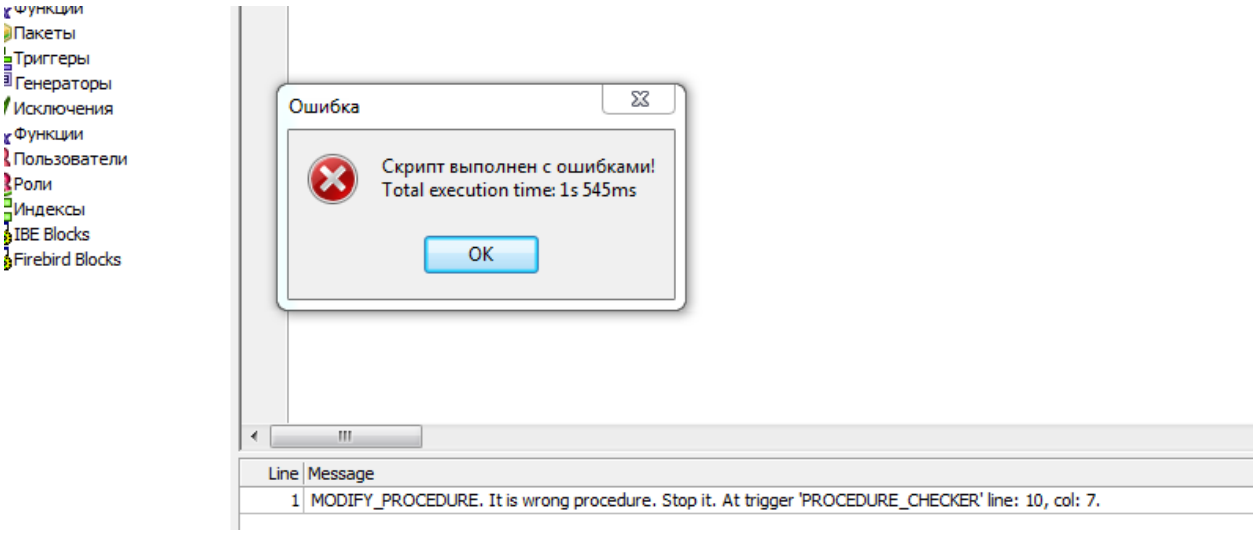
```
declare variable help_proc1 integer;
declare variable help_diag1 integer;
begin
:help_proc1=:help_proc;
:help_diag1=:help_diag;
:counth=0;
/*Цикл, в котором проверяется наличие поставленной процедуры в списке
допустимых процедур диагноза*/
 FOR select "DIAGNOSIS ID" from "DIAGNOSIS PROCEDURES"
 where "DIAGNOSIS ID" in
(select "DIAGNOSIS ID" from DIAGNOSIS
 where "DIAGNOSIS ID" in
(select "DIAGNOSIS ID" from CONDITIONS
 where "DIAGNOSIS ID" = 0))into :help_id1
 DO BEGIN
 if (:help_id1 = :help_proc1) then
 :counth=:counth+1;
 End
 Suspend;
 end
```

Триггер:

```
create trigger procedure_checker for CONDITIONS
active before insert or update
as
declare variable temp integer;
begin

select * from HELP1 (new."PROCEDURE ID", new."DIAGNOSIS ID" )
into :temp;
if (temp=0)
 then exception modify_procedure;
end
```

Результат работы, при попытке добавить процедуру, не соответствующую диагнозу:



4. Для заданного набора диагнозов после выписки пациента ставить план поступление его через заданный промежуток времени.

Процедура, реализующая создание плана поступления:

```
create procedure HELP2 (
help_id integer,
help_date date)
as
declare variable help_max integer;
begin
select max ("CONDITION ID") from "PATIENT CONDITION"
into help_max;
insert into "PATIENT CONDITION" ("PATIENT ID", "CONDITION ID")
values (:help_id, :help_max+1);
:help_date = dateadd (month,  6, :help_date);
insert into "CONDITIONS" ("CONDITION ID", DATE_OF_RECEIT)
values (:help_max+1, :help_date);
end
```

Триггер:

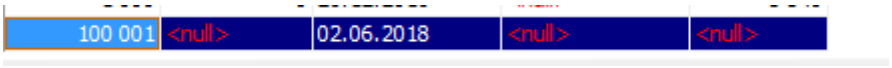
```
create trigger new_condition for "CONDITIONS"
active before insert or update
as
begin
if (new.DATE_OF_ISSUE is not null) then
    execute procedure HELP2 (new."CONDITION ID", new.DATE_OF_ISSUE);
end
```

Пример работы:

Пациенту с состоянием была добавлена дата выписки:

```
update CONDITIONS
set DATE_OF_ISSUE='12/02/2017'
where "CONDITION ID"=3;
```

После этого был автоматически сормирован план его поступления через пол года:



Вывод:

В результате работы было проведено знакомство с триггерами. Триггер можно считать автоматической процедурой, срабатывающей на стороне сервера в результате некоторого события. Триггер может срабатывать до наступления события или после его наступления.

Главным преимуществом триггеров является контроль целостности базы данных любой сложности. Так же упрощается приложение, так как часть логики уже выполняется на сервере.

Однако, у триггеров есть и недостатки. Например, большое количество триггеров сильно уменьшает производительность системы.