

Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

По дисциплине «Базы данных»

«Триггеры, вызовы процедур»

Работу выполнил студент группы № 43501/4

Смирнов Л.А.

Работу принял преподаватель

Мяснов А.А.

Санкт-Петербург

2015

Цели работы

Познакомить студентов с возможностями реализации более сложной обработки данных на стороне сервера с помощью хранимых процедур и триггеров.

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

Выполнение работы

1.1 Создадим триггер для автоматического заполнения ключевого поля.

Немного информации, зачем это нужно. Мы знаем, что поля первичного ключа обязательно должны быть уникальными в пределах таблицы. Чтобы обеспечить выполнение этого требования, и служит триггер. Вместе с ним будем использовать генератор, который будет выдавать значение для триггера и подставлять его в таблицу. Таким образом, в поле ID будет всегда уникальное значение, так как значение генератора будет увеличиваться каждый раз при обращении к триггеру.

```
CREATE OR ALTER TRIGGER PERSONAL_FILETRIG FOR PERSONAL_FILE
ACTIVE BEFORE INSERT POSITION 0
AS
begin
    IF (NEW.id_file IS NULL) THEN
        NEW.id_file = GEN_ID(GEN_TABLE_ID, 1);
    end
```

|

Триггер добавляет сгенерированное значение если ID будет NULL.

Проверим его работу. Генератор настроен на значение 116.

До добавления:

ID_FILE	ID_RECR...	REGISTR_DATE
100	1	12.10.2010
101	2	11.06.2010
102	3	11.01.2010
103	4	12.10.2010
104	5	12.10.2010
105	6	12.10.2010
106	7	12.10.2010
107	8	12.10.2010
108	9	12.10.2010
109	10	12.10.2010
110	11	12.10.2010
111	12	12.10.2010
112	13	12.10.2010
113	14	12.10.2010
114	15	12.10.2010

Введем добавление ниже несколько раз:

```
insert into Personal_file (Id_file, Id_recruit, Registr_date) values (NULL, 13, '2011-10-12');
```

Результат:

ID_FILE	ID_RECR...	REGISTR_DATE
100	1	12.10.2010
101	2	11.06.2010
102	3	11.01.2010
103	4	12.10.2010
104	5	12.10.2010
105	6	12.10.2010
106	7	12.10.2010
107	8	12.10.2010
108	9	12.10.2010
109	10	12.10.2010
110	11	12.10.2010
111	12	12.10.2010
112	13	12.10.2010
113	14	12.10.2010
114	15	12.10.2010
116	12	12.10.2011
117	13	12.10.2011

- 1.2 Создадим триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице

Для правильной работы данного триггера требуется временно удалить описание внешнего ключа на исследуемую таблицу.

```
ALTER TABLE DELAYS DROP CONSTRAINT FK_DELAYS1
```

Таблица Recruit до удаления

ID_RECR...	NAME
1	ISAKOV G.Y.
2	ISAKOV A.Y.
3	VASILIEV O.V.
4	IVANOV I.I.
5	PIROGOV 'Б.
6	JURAVLEV V.M.
7	DAVIDOV I.C.
8	BACUEV N.F.
9	TIHONOV E.C.
10	KOROLYOV V.M.
11	KOROLYOV K.M.
12	MALAHOV K.E.

Таблица Personal_file до удаления

ID_FILE	ID_RECR...	REGISTR_DATE
100	1	12.10.2010
101	2	11.06.2010
102	3	11.01.2010
103	4	12.10.2010
104	5	12.10.2010
105	6	12.10.2010
106	7	12.10.2010
107	8	12.10.2010
108	9	12.10.2010
109	10	12.10.2010
110	11	12.10.2010

Код создания триггера:

```
CREATE OR ALTER TRIGGER SECOND TRIGGER FOR recruit
ACTIVE AFTER DELETE POSITION 0
AS
begin
    delete from personal file where personal file.id_recruit = old.id_recruit;
end
```

Выполним запрос на удаление Личного дела с ID = 100:

```
delete from personal_file where Personal_file.id_file = 100;
```

2. Создадим триггер в соответствии с **индивидуальным заданием**, полученным у преподавателя

2.1 Сделать невозможным призыв военнослужащего, если у него есть действующая отсрочка.

```
CREATE OR ALTER trigger append sluzhba for activities
active before insert position 0
AS
declare variable id integer;
declare variable coun integer;
begin
    for select p1.id_file, count(p1.id_reason) from delays as p1 group by p1.id_file
    into :id, :coun
    do
        begin
            if (:id = new.id_file) then
                begin
                    if (coun = 1) then
                        begin
                            exception NOT COMMIT;
                        end
                    end
                end
            end
        end
    end
end
```

Так как если у призывника действует любая отсрочка, то, следовательно, является невозможным добавить для него какую-либо активность. Поэтому добавляемая активность ни коем образом не проверяется на «активность» = «призыв» и сразу блокируется триггером.

Приведем примеры блокировки. Пусть у нас есть следующий призывник:

```
insert into Personal_file (Id_file, Id_recruit, Registr_date) values (100, 1, '2010-10-12');
```

```
insert into Recruit (Id_recruit, Name, Birth_date, Registr_place, Id_mother, Id_father, Id_academic) values (1, 'ISAKOV G.Y.', '1990-10-09', 'Санкт-Петербург, Ленинский проспект, д. 11', 1, 1, 1);
```

Который имеет отсрочку:

```
insert into Delays (Id_file, Id_reason, Delay_date_to) values (100, 1, '2014-10-12');
```

```
insert into Kind_activity (Id_activity, Kind_activity) values (1, 'Delay University');
```

Попробуем добавить для него с помощью следующего запроса «активность», например, призыв:

```
insert into Activities (Id_file, Id_Activity, Activity_date) values (100, 499, '2013-09-11');
```

В итоге вывелось следующее исключение:

```
NOT_COMMIT.
```

```
NOT_COMMIT (DELAY IS RUNNING).
```

```
At trigger 'APPEND_SLUZHBA' line: 15, col: 17.
```

Таким образом, мы проверили, что добавление для призывника «активности» - «призыв» невозможно при действующей отсрочке.

2.2 Проверка наложения мероприятий. Если военнослужащий проходит сборы, и производится регистрация его службы по контракту, завершать прохождение сборов датой начала прохождения службы по контракту. Если военнослужащий проходит службу по призыву или контракту - делать невозможным прохождение сборов.

В данном задании необходимо создать триггер, вызывающий хранимую процедуру.

Начнем с ХП. Создадим ХП, которая будет завершать порождение сборов датой начала прохождения службы по контракту.

```
create or alter procedure CREATE KONTRAKT DATE (  
    OLD_ACT integer,  
    ID_PF integer,  
    NEW_DATE date)  
as  
begin  
    update Activities set activity_date_off = :new_date where id_file = :id_pf and id_activity = :old_act;  
    suspend;  
end
```

ОБУСЛОВИМСЯ, ЧТО ID_АКТИВИТИ = 450 ----- СБОРЫ. В реальной базе можно было бы сделать проверку по названию активности, но так как в база данных заполнена рандомными значениями, приходится поступать таким образом.

```
CREATE OR ALTER TRIGGER WATCHDOG ACTIVITIES FOR ACTIVITIES  
ACTIVE BEFORE INSERT POSITION 0  
AS  
declare variable id_f integer;  
declare variable id_a integer;  
declare variable out_date date;  
begin  
    for select p1.id_file, p1.id_activity, p1.activity_date_off from activities as p1 /* p1.  
    into :id_f, :id_a, :out_date  
    do  
        begin  
            if (new.id_activity = 499) then  
                begin  
                    if (:id_f = new.id_file) then  
                        begin  
                            if (:id_a = 450) then  
                                begin  
                                    if (:out_date is null) then  
                                        begin  
                                            execute procedure create kontrakt date (450, :id_f, new.activity_date);  
                                        end  
                                    end  
                                end  
                            end  
                        end  
                    end  
                end  
            else if (new.id_activity = 450) then  
                begin  
                    if (:id_f = new.id_file) then  
                        begin  
                            if ((:id_a = 498) or (:id_a = 499)) then  
                                begin  
                                    if (:id_f = new.id_file) then  
                                        begin  
                                            if ((:id_a = 498) or (:id_a = 499)) then  
                                                begin  
                                                    if (:out_date is null) then  
                                                        begin  
                                                            exception prizyv run;  
                                                        end  
                                                    end  
                                                end  
                                            end  
                                        end  
                                    end  
                                end  
                            end  
                        end  
                    end  
                end  
            end  
        end  
    end  
end
```

Триггер содержит много if-ов для того, чтобы упростить его структуру для наглядности.

Выполним проверку работы данного триггера. Для того чтобы проверить, находится ли призывник в данный момент на сборах, используется поле activity_date_off, которое определяет дату, когда закончилась данная активность. Соответственно, если это значение равно NULL, то значит, что данный призывник в данный момент на сборах. Данный факт и проверяется в условиях if.

Для примера возьмем призывника с ID_дела = 44643. Как мы видим – этот призывник находится на сборах в данный момент (поле activity_date_off = null).

ID_FILE	ID_ACTIV...	ACTIVITY_DATE	ACTIVITY_DATE_OFF
44 643	450	29.04.2007	<null>

Сделаем запрос на добавление id_activity = 499, что означает призыв по контракту.

```
insert into Activities (Id_file, Id_Activity, Activity_date) values (44643, 499, '2013-09-11');
```

1 record(s) was(were) updated in ACTIVITIES

1 record(s) was(were) inserted into ACTIVITIES

Execute time = 327ms

ID_FILE	ID_ACTIV...	ACTIVITY_DATE	ACTIVITY_DATE_OFF
44 643	450	29.04.2007	11.09.2013
44 643	499	11.09.2013	<null>

Видим, что информация в таблицу была успешно добавлена и поле для даты конца сборов заполнено датой начала службы по контракту.

Теперь проверим возможно ли добавить прохождение сборов при не окончившейся срочной службе или службе по контракту. Возьмем военнослужащего, который в данный момент проходит срочную службу (дата конца активности равна NULL).

ID_FILE	ID_ACTIV...	ACTIVITY_DATE	ACTIVITY_DATE_OFF
89 046	498	08.02.2016	<null>

Попробуем для него добавить активность с id = 450.

```
insert into Activities (Id_file, Id_Activity, Activity_date) values (89046, 450, '2013-09-11');
```

Система выводит нам:

```
PRIZYV_RUN.  
NOT_COMMIT (IN THE ARMY NOW).  
At trigger 'WATCHDOG_ACTIVITIES' line: 33, col: 18.
```

Это значит, что добавление не сработало по причине того, что военнослужащий находится на службе.

Выводы:

Триггер - это SQL процедура, которая срабатывает при каком-нибудь событии (INSERT, DELETE или UPDATE). Триггеры хранятся и управляются СУБД. Триггеры используются для поддержания ссылочной целостности данных в одинаковый манер реагируя на события изменения этих данных. Триггер не может быть вызван или выполнен вручную, СУБД автоматически вызывает его после модификации данных в соответствующей таблице. В этом и есть его отличие от хранимых процедур, которые нужно выполнять вручную вызовом CALL.

Достоинства триггеров:

- обеспечение надежности функционирования системы в условиях использования системы обычным пользователем, который может ошибаться, при изменении базы данных.
- возможность автоматической проверки триггерами DML *корректности данных*, определенные значения которых необходимо хранить в соответствующей таблице.
- создание более гибкой системы поддержки целостности БД по сравнению со стандартными средствами.

Недостатки триггеров

- влияние на производительность: перед выполнением каждой команды по изменению состояния базы данных СУБД должна проверить триггерное условие с целью выяснения необходимости запуска триггера для этой команды. Выполнение подобных вычислений сказывается на общей производительности СУБД