

Санкт-Петербургский государственный политехнический университет  
Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №5**

**Дисциплина:** Базы данных

**SQL-программирование: Триггеры, вызовы процедур**

Выполнила студентка гр. 43501/4

Перетяцько Е. В.

Преподаватель:

Мяснов А. В.

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

## Индивидуальное задание:

Реализовать триггеры:

1. При добавлении экскурсии к туру проверять нет ли уже такой экскурсии. Если есть - не добавлять.
2. При покупке тура стоимостью выше порогового значения добавлять самую популярную для данного тура экскурсию бесплатно.

**Триггер** — это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением INSERT, удалением DELETE строки в заданной таблице, или изменением UPDATE данных в определенном столбце заданной таблицы реляционной базы данных. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики. Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

Момент запуска триггера определяется с помощью ключевых слов BEFORE (триггер запускается до выполнения связанного с ним события; например, до добавления записи) или AFTER (после события). В случае, если триггер вызывается до события, он может внести изменения в модифицируемую событием запись (конечно, при условии, что событие — не удаление записи). Некоторые СУБД накладывают ограничения на операторы, которые могут быть использованы в триггере (например, может быть запрещено вносить изменения в таблицу, на которой «висит» триггер, и т. п.).

Кроме того, триггеры могут быть привязаны не к таблице, а к представлению (VIEW). В этом случае с их помощью реализуется механизм «обновляемого представления». В этом случае ключевые слова BEFORE и AFTER влияют лишь на последовательность вызова триггеров, так как собственно событие (удаление, вставка или обновление) не происходит.

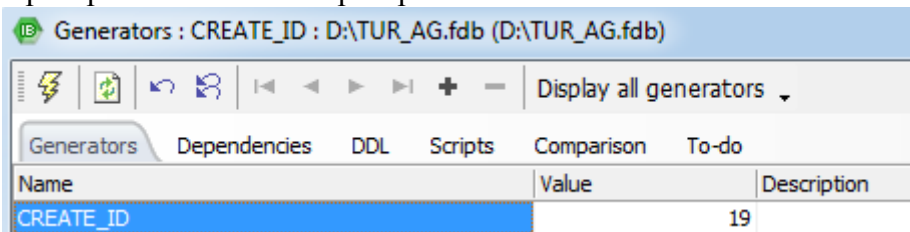
## Выполнение работы:

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.
- Создан триггер для автоматического заполнения поля contract\_ID таблицы(Contracts) (имитация автоинкремента).

### Листинг 1. Триггер автоинкремента

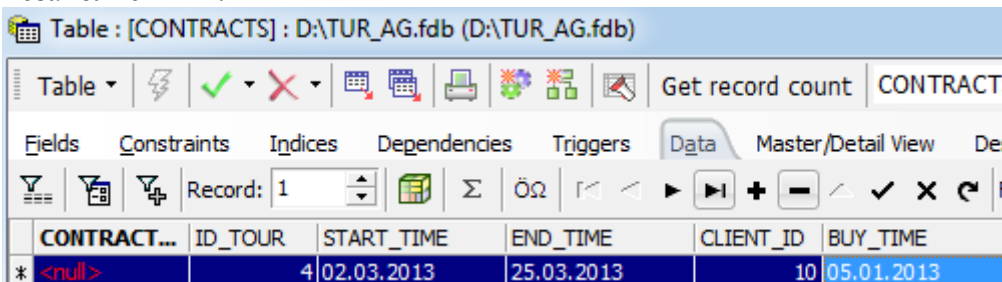
```
create generator create_id;  
set generator create_id to 19;  
create trigger auto_create_id for contracts before insert  
as  
begin  
    new.contract_ID=gen_id(create_id, 1);  
end
```

Проверим значение генератора:



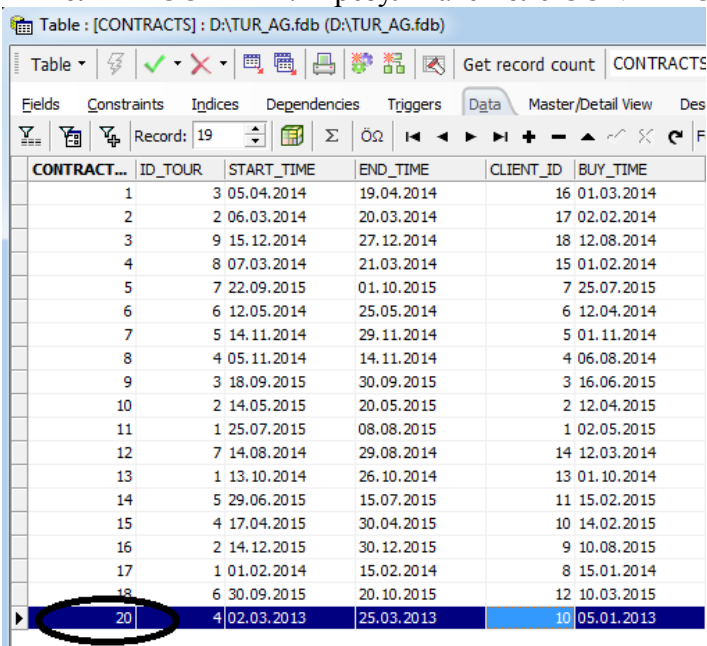
Name	Value	Description
CREATE_ID	19	

Добавим новую запись в таблицу CONTRACT, поле CONTRACT\_ID оставим незаполненным.



CONTRACT...	ID_TOUR	START_TIME	END_TIME	CLIENT_ID	BUY_TIME
* <null>	4	02.03.2013	25.03.2013	10	05.01.2013

Выполним COMMIT. В результате поле CONTRACT\_ID было заполнено автоматически



CONTRACT...	ID_TOUR	START_TIME	END_TIME	CLIENT_ID	BUY_TIME
20	4	02.03.2013	25.03.2013	10	05.01.2013

- Создан триггер, который проверяет данные на целостность: при попытке удаления или изменения записи в таблице предметов, на которую присутствуют внешние ссылки, он выдает ошибку:

#### Листинг 2. Триггер контроля целостности

```
create exception error_client 'Error. Cant delete or update in table client';
create trigger control_client for client before delete or update
as
begin
    if(old.client_id in (select client_id from contracts)) then
        exception error_client;
    end
```

Попытаемся удалить запись в таблице CLIENT, на которую ссылается запись из таблицы CONTRACT. При попытке удаления, возникает ошибка. Триггер работает верно.

Table : [CLIENT] : D:\TUR\_AG.fdb (D:\TUR\_AG.fdb)

Table | Fields | Constraints | Indices | Dependencies | Triggers | Data | Master/Detail View | Description | DDL | Grants

Record: 1

CLIENT_ID	NAME	NUMBER_PHONE
1	Dariya	7 412 582
2	Artem	7 125 852
3	Andrey	7 415 852
4	Nikita	7 412 585
5	Vlad	7 458 632
6	Ruslan	7 258 632
7	Anastasiya	7 455 896
8	Kristina	7 582 369
9	Katya	7 258 932
10	ElenUlyaa	7 458 962
11	Igor	7 412 586
12	Pavel	7 258 314
13	Alex	7 951 364
14	Sasha	7 562 344
15	Masha	7 856 314
16	Marina	7 215 634
17	Svatlana	7 852 146
18	Nataly	7 896 235

Error

ERROR\_CLIENT.  
Error. Cant delete or update in table client.  
At trigger 'CONTROL\_CLIENT' line: 5, col: 60.

OK

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

При добавлении экскурсии к туру(контракту) проверять нет ли уже такой экскурсии. Если есть - не добавлять.

#### Листинг 3. Триггер проверки наличия существующей экскурсии.

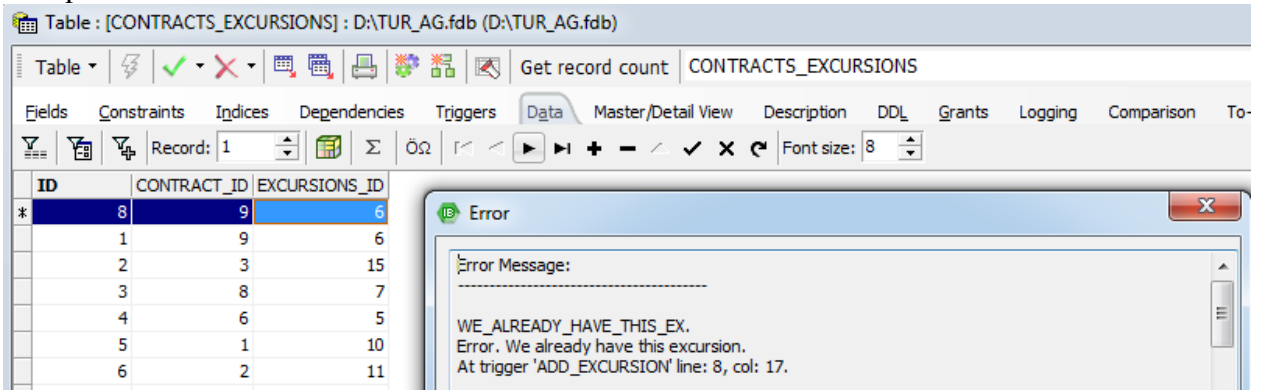
```
create exception we_already_have_this_ex 'Error. We already have this excursion';
create trigger add_excursion for contracts_excursions before insert
as
declare variable ex integer;
begin
    ex=-1;
    select excursions_id from contracts_excursions
    where(excursions_id=new.excursions_id)and(contract_id=new.contract_id) into :ex;
    if (ex<>-1) then
```

```

exception we_already_have_this_ex;
end

```

При попытке привязать с контракту существующую экскурсию, срабатывает триггер, который сообщает об ошибке.



3. Создать триггер в соответствии с **индивидуальным заданием**, вызывающий хранимую процедуру

При покупке тура стоимостью выше порогового значения добавлять самую популярную для данного тура экскурсию бесплатно.

#### Листинг 4. Добавление популярной экскурсии бесплатно при покупке тура стоимостью выше порогового значения.

```

create trigger add_free_excursion for contracts after insert
as
declare variable price_t integer;
declare variable ex_id integer;
declare variable place_id integer;
declare variable max_c integer;
declare variable new_id integer;
begin
    select price from tour where(tour_id=new.id_tour) into :price_t;
    if (price_t<40000) then
        exit;
    select first 1 excursions.place_of_interest_id, count(contracts_excursions.excursions_id) as
maxcount
    from contracts, contracts_excursions, excursions
    where(id_tour=new.id_tour)and(contracts.contract_id=contracts_excursions.contract_id)
    and(contracts_excursions.excursions_id=excursions.excursions_id)
    group by excursions.place_of_interest_id order by maxcount desc into :place_id, :max_c;
    ex_id=-1;
    select excursions_id from excursions
    where (place_of_interest_id=:place_id)and(price=0) into :ex_id;
    if(ex_id=-1) then
        begin
            select max(excursions_id) from excursions into :new_id;
            new_id=new_id+1;
            insert into excursions values (:new_id, :place_id, 0);
            ex_id=new_id;
            select max(id) from contracts_excursions into :new_id;
            new_id=new_id+1;
            insert into contracts_excursions values (:new_id, new.contract_id, :ex_id);
        end
    end

```

```

end
else
begin
    select max(id) from contracts_excursions into :new_id;
    new_id=new_id+1;
    insert into contracts_excursions values (:new_id, new.contract_id, :ex_id);
end
end

```

Посмотрим содержимое таблицы CONTRACTS\_EXCURSION, до добавления новой записи.

Table : [CONTRACTS\_EXCURSIONS] : D:\T...

ID	CONTRACT_ID	EXCURSIONS_ID
1	9	6
2	3	15
3	8	7
4	6	5
5	1	10
6	2	11
7	5	14

Была добавлена новая запись в таблицу CONTRACTS, содержащая тур, стоимостью больше заданного(40000).

Table : [CONTRACTS] : D:\T...

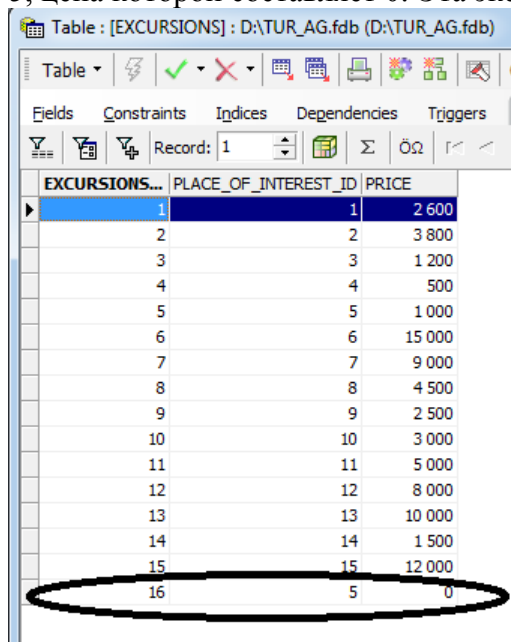
CONTRACT...	ID_TOUR
1	30
2	20
3	90
4	80
5	70
6	60
7	50
8	40
9	30
10	20
11	10
12	70
13	10
14	50
15	40
16	20
17	10
18	60
25	30

В таблице CONTRACTS\_EXCURSION автоматически была добавлена запись с созданным контрактом, которому была присвоена экскурсия бесплатно.

Table : [CONTRACTS\_EXCURSIONS] : D:\T...

ID	CONTRACT_ID	EXCURSIONS_ID
1	9	6
2	3	15
3	8	5
4	6	5
5	1	5
6	2	11
7	5	14
8	25	16

В таблице EXCURSIONS, была создана запись с самой популярной экскурсией под номер 5, цена которой составляет 0. Эта экскурсия была добавлена к выбранному контракту.



EXCURSIONS...	PLACE_OF_INTEREST_ID	PRICE
1	1	2 600
2	2	3 800
3	3	1 200
4	4	500
5	5	1 000
6	6	15 000
7	7	9 000
8	8	4 500
9	9	2 500
10	10	3 000
11	11	5 000
12	12	8 000
13	13	10 000
14	14	1 500
15	15	12 000
16	5	0

### Выводы:

В данной работе были изучены триггеры и генераторы.

Триггер – это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных.

Триггеры предназначены для выполнения простых операций по поддержанию целостности базы. Например, они могут генерировать индексы для таблицы, что является самым безопасным способом получения индекса. Также они позволяют каскадно удалять или изменять поля, зависящие от других таблиц.

Генератор – это специальный объект базы данных, который генерирует уникальные последовательные числа. Одним из применений генераторов является их использование в триггерах автоинкрементирования ключей. В таких триггерах необходимо использовать генераторы, так как они обеспечивают уникальность генерируемых значений даже при параллельной обработке нескольких запросов.