Санкт-Петербургский политехнический университет
Петра Великого
Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

Дисциплина: «Базы данных»

Тема: «SQL-программирование: Триггеры, вызов процедур»

Выполнил студент гр. 43501/3 _____ М.Ю. Попсуйко
(подпись)

Преподаватель _____ А.В. Мяснов
(подпись)

“___” _____ 2016 г.

Санкт-Петербург

2016

Оглавление

Цель 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:

Триггер для автоматического заполнения ключевого поля

В начале работы был создан генератор CAR_GEN:

```
CREATE sequence CAR_GEN ;
ALTER SEQUENCE CAR_GEN RESTART WITH 100001;
```

Затем создадим триггер, использующий данный генератор. Если id был введен вручную, то автоматически изменяется значения генератора для последующего использования.

```
create trigger car_id_autoinc for car
active before insert position 0
as
declare variable tmp DECIMAL (18 ,0) ;
begin
if (new.id is null ) then
new.id = gen_id ( CAR_GEN ,1) ;
else
begin
tmp = gen_id ( CAR_GEN ,0) ;
if ( tmp < new.id ) then
tmp = gen_id ( CAR_GEN , new.id - tmp ) ;
end
end
```

Выполним проверку работы триггера:

Добавим в таблицу CAR следующие строки:

```
insert into car values(100500,23,12,1,2011);
insert into car values(null,34,23,2,2012);
```

Теперь откроем и посмотрим таблицу CAR:

ID	ID_FACTORY	ID_MODIFICATION	ID_EQUIPMENT	CAR_YEAR
100 501	34	23	2	2 012
100 500	23	12	1	2 011
100 000	67	29	3	2 015
99 999	71	98	3	2 007

Рис. 1 Часть содержимого таблицы car

Как видно, автоматически создалась запись с id = 100501.

Триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице

Создадим исключение и триггер для контроля целостности данных в подчиненных таблицах при удалении/изменении записей в главной таблице CAR:

```
create exception ex_modify_car 'This car in other tables !';

create trigger car_delete_update for car
before delete or update
as
begin
if (old.id in( select id_car from sells))
then exception ex_modify_car ;
end
```

Выполним проверку работы триггера:

Попробуем удалить существующую запись из таблицы CAR:

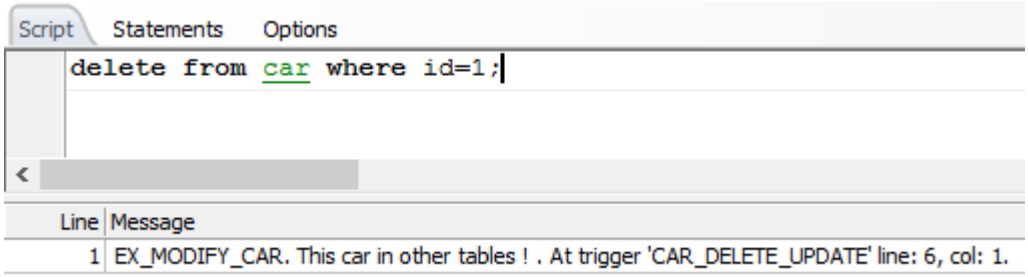

Рис. 2 Попытка удалить запись

Как видно, сделать этого нам не удалось, так как сработал триггер.

Индивидуальные задание: триггер 1

Задание: Сделать невозможным добавление дублирующих опций в заказ и в комплектацию.

Необходимо включить триггер TYPE_CHECKER_2

Часть 1: создание триггера, который проверяет существующие опции в заказе, перед добавлением новой.

```
create exception ex_type_exists 'Type already exists.';

create trigger type_checker for additional_options
active before insert or update
as
declare variable temp integer;
begin
for
select additional_options.id_options from additional_options
where additional_options.id_sell=new.id_sell
into :temp
do
begin
if(:temp=new.id_options)
then exception ex_type_exists;
end
end
```

Проверим работу триггера:

Попробуем создать новую запись с уже существующими id_sells и id_options в таблицу additional_options:

ID_SELL	ID_OPTIONS
1	30 716
2	50 120
3	27 156
4	10 867
7	14 154

Рис. 3 Часть содержимого таблицы additional_options

Script | Statements | Options

```
insert into additional_options values(1,30716);
```

Line	Message
1	EX_TYPE_EXISTS. Type already exists. At trigger 'TYPE_CHECKER' line: 13, col: 1.

Рис. 4 Попытка добавить уже существующую опцию

Сделать этого не удалось, так как сработал триггер.

Часть 2: Создание триггера, который проверяет существующие опции в комплектации, перед добавлением новой.

```
create trigger type_checker_2 for "CAR_DEFAULT-OPTIONS"
active before insert or update
as
declare variable temp integer;
begin
for
select "CAR_DEFAULT-OPTIONS".id_options from "CAR_DEFAULT-OPTIONS"
where "CAR_DEFAULT-OPTIONS".id_equipment=new.id_equipment
into :temp
do
begin
if(:temp=new.id_options)
then exception ex_type_exists;
```

```
        end
        end
```

Выполним проверку работы триггера:

Создадим новую запись в CAR_DEFAULT-OPTIONS с уже имеющимися ID_OPTIONS и ID_EQUIPMENT:

ID_OPTIONS	ID_EQUI...
99 995	1
3	1
99 989	1
5	1
99 978	1
99 977	1

Рис. 5 Часть содержимого таблицы car_default_options

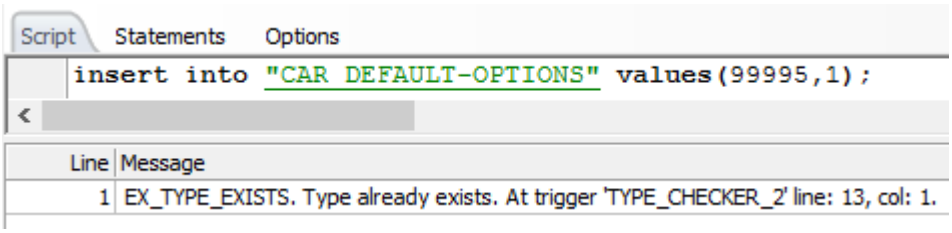

Рис. 6 Попытка добавить уже существующую опцию

Новую запись создать не удалось, триггер сработал.

Индивидуальные задание: триггер 2

Задание: При формировании заказа общей стоимостью выше заданного значения добавлять в заказ наиболее популярную опцию за последний год с нулевой стоимостью.

Примечание: в таблицу additional_options был добавлен столбец price, значению которого равно значению price из таблицы options или же 0, согласно постановке задания.

```
        ALTER TABLE additional_options ADD PRICE FLOAT;

        update additional_options t1
        set
        t1.price=(select price from options t2 where t2.id=t1.id_options);
```

Для начала необходимо создать хранимую процедуру, которая будет возвращать наиболее популярную опцию за последний год.

```
        create procedure get_popular_option_at_last_year
        returns(output_id integer)
        as
        begin
        select id from(select first 1 options.id, count(options.id) as
total_count from options
        join additional_options on additional_options.id_options=options.id
        join sells on sells.id=additional_options.id_sell
        where sells.sell_date between (dateadd(year, -1, current_timestamp)) and
current_date
        group by options.id
```

```
order by total_count desc)
into :output_id;
end
```

При сумме более 600000 будет бесплатно добавлена наиболее популярная опция за последний год.

```
create trigger extra_option for sells
active after insert
as
declare variable temp integer;
begin
if (new.price>600000) then
begin
select * from  get_popular_option_at_last_year into :temp;
insert into additional_options values(
new.id,
:temp,
0);
end
end
```

Выполним проверку работы:

Для этого добавим в таблицу SELLS следующую запись:

```
insert into sells values(100001,12345,12345,12345,650000,current_date);
```

Рассмотрим созданную нами запись в таблицах SELLS и ADDITIONAL_OPTIONS:

ID	ID_CAR	ID_CLIENT	ID_EMPLOYEE	PRICE	SELL_DATE
100 002	12 345	12 345	12 345	550 000,00	10.12.2016
100 001	12 345	12 345	12 345	650 000,00	10.12.2016
100 000	19 516	19 869	92 154	623 987,42	24.09.2016
99 999	85 965	61 176	79 719	603 409,21	22.04.2012

Рис. 7 Часть содержимого таблицы sells

ID_SELL	ID_OPTIONS	PRICE
100 001	94 804	0,000
100 000	40 556	7 321,000
100 000	36 580	2 958,000
99 999	87 875	9 005,000

Рис. 8 Часть содержимого таблицы additional_options.

При добавлении записи, сумма которой более 600000, бесплатно добавляется наиболее популярная опция за последний год.

Выводы:

В результате выполнения данной лабораторной работы ознакомился со сложной обработкой данных на стороне сервера при помощи хранимых процедур и триггеров.

В ходе работы были созданы триггера в соответствии с общими и индивидуальными заданиями.

Можно заметить, что триггер очень похож на хранимую процедуру, но запуск его, в отличии от ХП, производится автоматически сервером при модификациях данных таблицы. Триггер очень полезен, так как он ограждает БД от пользователей, пользователь не имеет доступа к таблицам, а значит не может их повредить. Триггер не может навредить БД так как выполняется в виде транзакции и при обнаружении ошибки, порчи таблицы произойдет отказ этой транзакции.

Триггер запускается с помощью ключей BEFORE или ALTER, что определяет момент его запуска – до выполнения, связанного с ним события или после, соответственно.