**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Институт информационных технологий и управления**

**Кафедра компьютерных систем и программных технологий**

**ОТЧЕТ**

**по лабораторной работе №6**

**«SQL-программирование: триггеры, вызовы процедур»**

**Информационное обеспечение систем управления**

Студент гр. 43501/1 Васильев А.М.

Преподаватель Мяснов А.В.

**Санкт-Петербург**

**2015**

# 1. Цель работы

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

# 2. 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

# 3. Общие примеры триггеров

Триггер привязан к конкретной таблице и выполняется до или после заданного типа операции над ней. Если триггеров у одной таблицы несколько, порядок их выполнения можно задавать с помощью position.

## 3.1 Триггер для автоматического заполнения ключевого поля таблицы numbers.

Ранее был создан генератор идентификаторов для таблицы:

create sequence gen\_numid;

Триггер:

set term ^ ;

create or alter trigger trg\_gen\_numid for numbers

active before insert position 0

as

begin

if ((new.id is null) or (new.id = 0)) then

begin

new.id = gen\_id(gen\_numid, 1);

end

end ^

set term ; ^

Выполняется проверка на присутствие и неравенство нулю ключевого поля в новой записи, если условие не выполнено, то перед добавлением новой записи в ней заполняется ключевое поле.

Пример использования:

SQL> select count(\*) from numbers;

COUNT

============

110000

**SQL> insert into numbers (number) values (9210010101);**

SQL> select count(\*) from numbers;

COUNT

============

110001

SQL> select first 1 \* from numbers order by -id;

ID NUMBER

============ =====================

110001 9210010101

Триггер сработал, запись с новым id добавлена.

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

Контролируется удаление записей из таблицы subs, которое должно вызывать удаление транзакций и пополнений счета заданного абонента из соответствующих таблиц.

set term ^ ;

create or alter trigger trg\_del\_sub for subs

active before delete position 0

as begin

delete from transactions where sub = old.id;

delete from payments where sub = old.id;

end ^

set term ; ^

Проверка:

SQL> delete from subs where id=1;

SQL> select count(\*) from subs;

COUNT

============

99999

Без триггера эту операцию вообще невозможно было бы совершить, так как она нарушает контроль ссылочной целостности.

# 4. Индивидуальное задание

## 4.1 Не давать производить транзакцию/не создавать действие, если сумма на счету абонента ниже порога отключения по тарифу.

Триггер:

create or alter exception NEG\_BALANCE 'Недостаточно денег на счету';

set term ^ ;

create or alter trigger trg\_check\_balance for transactions

active before insert position 0

as

declare variable balance decimal(10,2);

begin

balance = (select balance from subs where id=new.sub);

if(balance <= 0) then exception NEG\_BALANCE;

end ^

set term ; ^

Создается исключение типа NEG\_BALANCE.

Проверка, выберем абонента с минимальным балансом:

SQL> select first 1 id,balance from subs order by balance;

ID BALANCE

============ =====================

96334 -282.59

SQL> execute procedure call(96334,1);

Statement failed, SQLSTATE = 42000

exception 1

-NEG\_BALANCE

-Недостаточно денег на счету

-At trigger 'TRG\_CHECK\_BALANCE' line: 7, col: 19

At procedure 'CALL' line: 12, col: 2

Проверка результатов операции:

SQL> select count(\*) from transactions;

COUNT

============

100000

SQL> select balance from subs where id=96334;

BALANCE

=====================

-282.59

Операция не выполнена, транзакция не добавлена, баланс не изменился.

## 4.2 При добавлении договора выдавать бонус

Создадим специальный бонус без условий, который нужен для ведения статистики:

insert into bonuses values (4,'Рекрут',NULL,NULL,NULL,NULL,NULL,NULL);

Хранимая процедура, которую вызывает триггер:

create sequence gen\_sbid;

set term ^ ;

create procedure add\_bonus (sub\_id int,bonus\_id int,sms\_amount int,minutes\_amount int,days int) as

declare variable end\_date date;

declare variable start\_date date;

begin

start\_date = current\_date;

if(days is null) then

start\_date = null;

end\_date = dateadd(days day to :start\_date);

insert into subs\_bonuses (id,sub,bonus,amount\_left\_sms,amount\_left\_calls,start\_date,expire\_date) values

(gen\_id(gen\_sbid,1),:sub\_id, :bonus\_id, :sms\_amount, :minutes\_amount,:start\_date,:end\_date);

end ^

Триггер:

create or alter trigger trg\_new\_sub for subs

active after insert position 0 as

begin

execute procedure add\_bonus (new.id,4,NULL,NULL,NULL);

update subs set balance = balance+100 where id = new.id;

end ^

set term ; ^

Проверка:

SQL> insert into subs values (1,1,1,0.0,1);

SQL> select \* from subs where id=1;

ID PERSON NUMBER BALANCE TARIFF

============ ============ ============ ===================== ============

1 1 1 100.00 1

SQL> select \* from subs\_bonuses where sub=1;

ID SUB BONUS AMOUNT\_LEFT\_SMS AMOUNT\_LEFT\_CALLS START\_DATE EXPIRE\_DATE

============ ============ ============ =============== ================= =========== ===========

2 1 4 <null> <null> <null> <null>

Новому абоненту на счет зачислены 100 рублей, и добавлена запись о бонусе.

# 5. Вывод

В ходе данной работы были изучены триггеры. Триггер связан с определенной таблицей и вызывается автоматически до или после определенного действия по изменению записей в таблице. Триггеры – основной способ контроля целостности данных при удалении записи в одной из связанных таблиц. Другой способ – назначение правила каскадного удаления. Триггер можно использовать для автоматического заполнения полей таблицы при добавлении записей или контроля проводимых операций. При этом триггеры усложняют работу БД и негативно влияют на производительность.