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

Отчет по лабораторной работе № 4

“Триггеры”

по дисциплине "Базы данных"

Группа: 43501/3
Студент: Кирпиченков П.С.
Преподаватель: Мяснов А.В.

Санкт-Петербург
2016

1. Постановка задачи

- 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
- Создать триггер в соответствии с индивидуальным заданием, полученным у преподавателя
- Создать триггер в соответствии с индивидуальным заданием, вызывающий хранимую процедуру
- Выложить скрипт с созданными сущностями в svn
- Продемонстрировать результаты преподавателю

Индивидуальное задание:

- Вызов процедур копирования патронов и креплений при создании оружия предыдущего калибра.
- Сделать невозможным множественное добавление одного типа крепления к оружию.

2. Теоретические положения

Триггеры используются для автоматического выполнения действий при возникновении определенных условий. Это может быть необходимо для упрощения работы с базой данных или для поддержания целостности данных в ней. Создание триггеров похоже на создание хранимых процедур с некоторыми отличиями.
Общий синтаксис:

```
CREATE TRIGGER TriggerName
FOR TableName
<Trigger Type Keywords>
AS
<local variable declarations>
BEGIN
  <body of trigger>
END
```

Кроме отличий в объявлении, триггер явно привязывается к таблице. В разделе Trigger Type Keywords указываются дополнительные параметры триггера. Основной параметр – условие срабатывания триггера. Триггер может быть привязан к добавлению, обновлению или удалению записей таблицы, срабатывание на нескольких условиях не поддерживается. Помимо этого можно установить приоритет на случай события, вызывающего выполнение нескольких триггеров. При наличии приоритетов последовательность вызова триггеров будет точно определена, если приоритеты различны. Также можно сделать триггер неактивным при создании с возможностью активации позже.
Для автоматического заполнения первичного ключа были использованы генераторы, получение значений из которого контролировалось триггером. Генератор хранит целочисленное значение и автоматически изменяет его при чтении.

3. Ход выполнения и результаты

Триггер для автоматического заполнения ключевого поля. Создан для таблицы Country.

Был создан и обновлен генератор для первичного ключа:

```
SQL> create generator gen_id_country;
SQL> SELECT GEN_ID( gen_id_country, 0 ) FROM RDB$DATABASE;
            GEN_ID
======================
                 0
SQL> select max(id) from country;

        MAX
============
         7
SQL> set generator gen_id_country to 7;
```

Создание триггера:

```
set term ^;
create or alter trigger trig_id_country for Country
active
before insert
as
begin
            if (new.id is NULL) then
            new.id = gen_id(gen_id_country, 1);
end^
set term ;^
```

Проверка:

```
SQL> insert into Country (name) values ('Canada') returning id, name;

         ID NAME
===============
          8 Canada

SQL> insert into Country (name) values ('Japan') returning id, name;

         ID NAME
============
          9 Japan

SQL> select id, name from country;

         ID NAME
============
          1 Russia
          2 USA
          3 Germany
          4 UK
          5 Belgium
          8 Canada
          7 temp
          9 Japan
```

Триггеры, запрещающие удаление или изменение страны при наличии зависимых записей. Проверка производится при изменении или удалении записи из таблицы Country. Если запись используется, выдается исключение. Исключение было предварительно создано.

```
SQL> create exception dependent_field 'Can''t alter or remove record due to dependent field.';
set term ^;
create trigger delete_check for Country
active
before delete
as
declare variable id_country integer;
begin
	for select id_country
	from Manufacture
	into :id_country
	do begin
		if (old.id = :id_country) then
			exception dependent_field;
	end
end^

create trigger update_check for Country
active
before update
as
declare variable id_country integer;
begin
	for select id_country
	from Manufacture
	into :id_country
	do begin
		if (old.id = :id_country) then
			exception dependent_field;
	end
end^
set term ;^
```

Проверки:

```
SQL> delete from country where name = 'temp';
Statement failed, SQLSTATE = HY000
exception 1
-DEPENDENT_FIELD
-Can't alter or remove record due to dependent field.
-At trigger 'DELETE_CHECK' line: 11, col: 29
SQL> update country set name = 'newtemp' where name = 'temp';
Statement failed, SQLSTATE = HY000
exception 1
-DEPENDENT_FIELD
-Can't alter or remove record due to dependent field.
-At trigger 'UPDATE_CHECK' line: 11, col: 29
SQL> select name from country;
SQL> select name from country;
```

```
NAME
===============
Russia
USA
Germany
UK
Belgium
Canada
temp
Japan

SQL> update country set name = 'JP' where name = 'Japan';
SQL> select name from country;
NAME
===============
Russia
USA
Germany
UK
Belgium
Canada
temp
JP
```

Триггер для копирования боеприпасов при переводе оружия на предыдущий калибр.

Триггер срабатывает при добавлении нового оружия, у которого имя, тип и производитель совпадают с существующим, а калибр предыдущий по отношению к имеющемуся.

```
create or alter trigger downgrade for weapon
active
before insert
as
        declare variable weapon_name varchar(100);
        declare variable cntr integer;
        declare variable id_caliber_req integer;
begin
        id_caliber_req = NULL;
        weapon_name = NULL;
        select first 1 id
        from caliber
        where caliber.diameter >
                (select diameter from caliber where id = new.id_caliber)
        order by diameter asc, length asc
        into :id_caliber_req;
        if (id_caliber_req is not NULL) then
                select name
                from weapon
                where id_caliber = :id_caliber_req and
                    id_type = new.id_type and
                    id_manufacture = new.id_manufacture and
                    name = new.name
                into :weapon_name;
        if (weapon_name is not NULL) then
        begin
                execute procedure downgrade :weapon_name
                returning_values :cntr;
                new.name = :weapon_name || '_' || cast (new.id as varchar(100));
        end
end^
```

Второй триггер копирует крепления при создании оружия предыдущего калибра:

```
create or alter trigger copymounts for weapon
active
after insert
as
        declare variable weapon_name varchar(100);
        declare variable new_name varchar(100);
        declare variable cntr integer;
        declare variable id_caliber_req integer;
begin
        id_caliber_req = NULL;
        weapon_name = NULL;
        select first 1 id
        from caliber
        where caliber.diameter >
                (select diameter from caliber where id = new.id_caliber)
        order by diameter asc, length asc
        into :id_caliber_req;
        if (id_caliber_req is not NULL) then
                select first 1 name
                from weapon
                where id_caliber = :id_caliber_req and
                    id_type = new.id_type and
                    id_manufacture = new.id_manufacture and
                    new.name like name || '%'
                order by id
                into :weapon_name;
        if (weapon_name is not NULL) then
        begin
                select first 1 new.name from weapon into :new_name;
                execute procedure copymounts :weapon_name, :new_name
                returning_values :cntr;
        end
end^
```

Проверка:

```
SQL> select weapon.name as weapon_name, shell.name as shell_name, caliber.diameter as diameter, caliber.length as length
CON> from weapon join caliber on weapon.id_caliber = caliber.id join shell on shell.id_caliber = caliber.id;

WEAPON_NAME SHELL_NAME   DIAMETER    LENGTH
=============== ==================== ======================
temp            5.45x39 b       0.00            0.00
AK-74           5.45x39 b       5.45           39.00
PM              9x18 b          9.00           18.00
PM              9x18 tr         9.00           18.00
MP-5            9x19 b          9.00           19.00
SCAR-L          5.56x45 tr      5.56           45.00
AK-74_6         5.45x39 b       0.00            0.00

SQL> select weapon.name as weapon, mount.name as mount
CON> from weapon join weapon_mount on weapon_mount.id_weapon = weapon.id join mount on weapon_mount.id_mount =
mount.id;

WEAPON                                                                  MOUNT
================================================================================
AK-74                                                           Warsaw Pact mounting dovetail
temp                                                            Warsaw Pact mounting dovetail
AK-74_6                                                         Warsaw Pact mounting dovetail
```

Триггер для запрета добавления одинаковых типов крепления к одному оружию вместо изменения их количества:

```
set term ^;
create or alter trigger mountcontrol for weapon_mount
active
before insert
as
        declare variable id_weapon_mount integer;
begin
        for select id
        from weapon_mount
        where id_weapon = new.id_weapon and id_mount = new.id_mount
        into id_weapon_mount
        do
                exception mount_exists;
end^
set term ;^
```

Проверка:

```
SQL> select * from weapon_mount;

        ID    ID_WEAPON     ID_MOUNT     QUANTITY
============ ============ ============ ============
         1          1          1          1
         2          5          1          1
         3          6          1          1

SQL> insert into weapon_mount (id, id_weapon, id_mount, quantity)
CON> values (4, 1, 1, 1);
Statement failed, SQLSTATE = HY000
exception 2
-MOUNT_EXISTS
-The mount already exists for this weapon. Change amount if needed.
-At trigger 'MOUNTCONTROL' line: 11, col: 2
```

4. Выводы

Триггеры позволяют выполнять автоматически нужную работу по поддержанию целостности данных и их обновлению. При создании триггеров стоит помнить о том, что они будут выполняться при всех действиях того типа, к которому привязаны. Если триггеров будет слишком много, работа с базой замедлится. Выполнение триггеров происходит неявно, в результате их работы клиент может получить не то, чего ждал, если не знает об их существовании. В остальном это удобный способ разгрузки клиентского кода по работе с базой.