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

Отчёт о лабораторной работе №6

Дисциплина: Базы данных

Тема: Триггеры

Выполнил студент гр. 43501/1 Нагорнов А.А.

Руководитель Мяснов А.В.

Санкт -Петербург
2016

1. Цели работы

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

2. Программа работы

- 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

3. Ход работы

Триггер (trigger) — это хранимая процедура особого типа, которая не вызывается непосредственно, а исполнение которой обусловлено наступлением одного из событий, относящегося к одной конкретной таблице (представлению), или наступлению одного из событий базы данных.

Триггер, вызываемый при наступлении события таблицы, связан с одной таблицей или представлением, с одним или более событиями для этой таблицы или представления (INSERT, UPDATE, DELETE) и ровно с одной фазой такого события (BEFORE или AFTER).

Триггер выполняется в той транзакции, в контексте которой выполнялась программа, вызвавшая соответствующее событие. Исключением являются триггеры, реагирующие на события базы данных. Для некоторых из них запускается транзакция по умолчанию.

Для каждой комбинации фаза-событие может быть определено более одного триггера. Порядок, в котором они выполняются, может быть указан явно с помощью дополнительного аргумента POSITION в определении триггера. Максимальная позиция равна 32767. Триггеры с меньшей позицией вызываются первыми.

Существует шесть основных вариантов соотношения событие-фаза для таблицы(представления):

до добавления новой строки	(BEFORE INSERT)
после добавления новой строки	(AFTER INSERT)
до изменения строки	(BEFORE UPDATE)
после изменения строки	(AFTER UPDATE)
до удаления строки	(BEFORE DELETE)
после удаления строки	(AFTER DELETE)

4. Задание

4.1. Триггер для автоматического заполнения ключевого поля

```
SET term ^ ;
create generator increment^
create or alter trigger auto_generation for People
before insert
as
begin
   new.ID = gen_id(increment,1);
end^
SET term ; ^
```

```
SQL> set generator increment to 26;
SQL> insert into People values(null,'ASASA','USSR');
SQL> select * from people where id>20;

          ID NAME                                               COUNTRY
============ ================================================== ==================================================
          21 Dima Bilan                                         Russia
          22 Michail Bojarskij                                  Russia
          23 Vera Brezneva                                      Russia
          24 Butusov                                            Russia
          25 Visockij                                           USSR
          27 ASASA                                              USSR
```

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

```
SET term ^ ;
create exception error1 'Error: cannot delete or update this track'^
create or alter trigger del_up for Tracks
before delete or update
as
begin
If (old.ID in (select ID_track from Performers)) then
exception error1;
end^
SET term ; ^
```

```
SQL> delete from tracks where id = 2;
Statement failed, SQLSTATE = HY000
exception 2
-ERROR1
-Error: cannot delete or update this track
-At trigger 'DEL_UP' line: 5, col: 50
SQL>
```

5. Индивидуальное задание

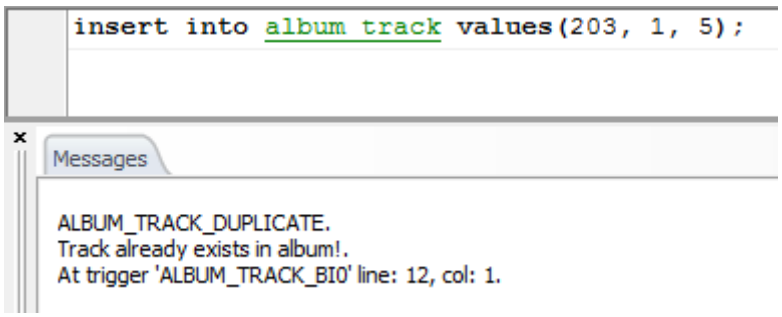
Проверять дубли при добавлении трека к альбому. При дубле - выбрасывать исключение.

а) Триггер без вызывающей хранимой процедуры

```
--ИСКЛЮЧЕНИЯ
SET TERM ^ ;
CREATE or ALTER EXCEPTION ALBUM_TRACK_DUPLICATE 'Track already
present in album!'^
SET TERM ; ^

--ТРИГГЕР
SET TERM ^ ;
CREATE OR ALTER TRIGGER ALBUM_TRACK_BI0 FOR ALBUM_TRACK
ACTIVE BEFORE INSERT POSITION 0
AS
begin
  if((select count(*) from album_track where album_track.track_id = New.track_id
and album_track.album_id = New.album_id) != 0)
  then EXCEPTION ALBUM_TRACK_DUPLICATE;
end
^
SET TERM ; ^
```

Результат:



б) Триггер с вызывающей хранимой процедурой

```
--ИСКЛЮЧЕНИЯ
SET TERM ^ ;
CREATE or ALTER EXCEPTION ALBUM_TRACK_DUPLICATE 'Track already
exists in album!'^
SET TERM ; ^

--ПРОЦЕДУРА
SET TERM ^ ;
CREATE or ALTER procedure check_duplicate(new_track_id int, new_album_id
int)
returns(result int)
as
```

```
declare variable r_count int = 0;
begin
        select count(*)
                from album_track
                        where album_track.track_id = :new_track_id and
album_track.album_id = :new_album_id
                                into :r_count;
        begin
        result = :r_count;
        suspend;
        end
end^
SET TERM ; ^

--ТРИГГЕР
SET TERM ^ ;
CREATE OR ALTER TRIGGER ALBUM_TRACK_BI0 FOR ALBUM_TRACK
ACTIVE BEFORE INSERT POSITION 0
AS
declare variable tmpRes int = 0;
begin
  select * from check_duplicate(new.track_id, new.album_id)
        into :tmpRes;

        if(tmpRes != 0)
        then
        begin
        EXCEPTION ALBUM_TRACK_DUPLICATE;
        end
end^
SET TERM ; ^
```

Результаты совпали.

6. Вывод

Триггеры в основном используются для поддержки целостности базы данных. Это специальный тип хранимой процедуры, который не вызывается непосредственно пользователем. При создании триггера он настраивается на срабатывание при указанном изменении данных в конкретной таблице или столбце.