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

Отчёт по лабораторной работе

Дисциплина: Базы данных

Тема: Триггеры. Вызовы процедур.

Выполнил студент гр. 43501/1 _____ А.О. Перешеин
(подпись)
Руководитель _____ А.В. Мяснов
(подпись)

“__” _____ 2015 г.

Санкт-Петербург
2015

1. Цели работы

Познакомить студентов 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, реализующий следующее задание:

Для любой таблицы БД создать триггер, осуществляющий автоматическое заполнение значения первичного ключа при добавлении новых записей.

- Создать триггер, реализующий следующее задание:

При превышении количества продаж альбома более заданной величины повышать рейтинг альбому на фиксированную величину.

- Выложить скрипт с созданными сущностями в svn.
- Продемонстрировать результаты преподавателю.

3. Реализация триггеров

Для каждой таблицы может быть назначена хранимая процедура без параметров, которая вызывается при выполнении оператора модификации этой таблицы (INSERT, UPDATE, DELETE). Такие хранимые процедуры получили название триггеров. Триггеры выполняются автоматически, независимо от того, что именно является причиной модификации данных - действия человека оператора или прикладной программы. Синтаксис оператора создания триггера:

```
CREATE TRIGGER <имя_триггера>
        ON <имя_таблицы>
        FOR { INSERT | UPDATE | DELETE }
             [, INSERT | UPDATE | DELETE ] ...
        AS <SQL_оператор>
```

Ключевое слово `ON` задает имя таблицы, для которой определяется триггер, ключевое слово `FOR` указывает какая команда (команды) модификации данных активирует триггер. Операторы `SQL` после ключевого слова `AS` описывают действия, которые выполняет триггер и условия выполнения этих действий. Здесь может быть перечислено любое число операторов `SQL`, вызовов хранимых процедур и т.д.

1) Триггер для автоматического заполнения ключевого поля

Для автоматического заполнения ключевого поля сначала создадим последовательность, которую будем использовать при генерации первичного ключа для добавляемой записи. Так как записей на настоящий момент в таблице Artists 5, то используем следующий синтаксис:

```
CREATE SEQUENCE ART_SEQ RESTART WITH 5
```

Для создания триггера будем использовать блок if (<…>) then <...> для проверки того, что первичный ключ для записи не задан и команду next value for <sequence_name> для генерации следующего по порядку id. Триггер будет активироваться перед командой insert:

```
CREATE OR ALTER trig1 artists_bi0 for artists
active before insert position 0
```

```
AS
begin
  if(new.art_id is null) then
  new.art_id = next value for art_seq;
end
```

Проверка работоспособности триггера:

```
SQL> select * from artists;

    ART_ID ARTIST
========== ========================
         1 Pantera
         2 Amorphis
         3 Slayer
         4 Ice Cube
         5 Edge of Sanity

SQL> insert into artists values (null, 'Bad Brains');
SQL> select * from artists;

    ART_ID ARTIST
========== ========================
         1 Pantera
         2 Amorphis
         3 Slayer
         4 Ice Cube
         5 Edge of Sanity
         6 Bad Brains

SQL> insert into artists values (null, 'Converge');
SQL> select * from artists;

    ART_ID ARTIST
========== ========================
         1 Pantera
         2 Amorphis
         3 Slayer
         4 Ice Cube
         5 Edge of Sanity
         6 Bad Brains
         7 Converge
```

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

При удалении плейлиста, все записи из таблицы pl_tr, содержащей все треки этого плейлиста, также должны быть удалены. При изменении имени плейлиста, в записях таблицы pl_tr также должно измениться имя. Узнавать из-за удаления или из-за изменения данных был вызван триггер будем через булевые переменные deleting и updating (в блоке if-else достаточно использовать один из них).

```
CREATE OR ALTER trigger trig2 for playlists
active before update or delete position 0
AS
begin
  if(deleting)
  then delete from pl_tr where pl_tr.name = old.name;
  else update pl_tr set pl_tr.name = new.name where pl_tr.name = old.name;
end
```

Проверка работоспособности триггера:

[исходная таблица с плейлистами]

```
SQL> select * from playlists;

NAME                 DATE_OF_CR
==================== ==========
list1                2013-01-09
list3                2014-09-30
mp3                  2012-05-13
list2                2014-03-21
home                 2013-01-30
list4                2014-12-15

SQL> select * from pl_tr;

NAME                    TR_ID
==================== ==========
home                    21
home                    22
home                    23
home                    24
home                    25
home                    26
home                    27
home                    32
home                    33
home                    34
list1                   1
list1                   2
list2                   3
list2                   4
list2                   5
mp3                     35
mp3                     36
mp3                     37
mp3                     38
mp3                     50

NAME                    TR_ID
==================== ==========
mp3                     51
mp3                     52
list4                   13
```

[изменение имени плейлиста]

```
SQL> update playlists set playlists.name = 'changed' where playlists.name = 'list4';
SQL> select * from playlists;

NAME                 DATE_OF_CR
==================== ==========
list1                2013-01-09
list3                2014-09-30
mp3                  2012-05-13
list2                2014-03-21
home                 2013-01-30
changed              2014-12-15
```

SQL> select * from pl_tr;

NAME	TR_ID
home	21
home	22
home	23
home	24
home	25
home	26
home	27
home	32
home	33
home	34
list1	1
list1	2
list2	3
list2	4
list2	5
mp3	35
mp3	36
mp3	37
mp3	38
mp3	50

NAME	TR_ID
mp3	51
mp3	52
changed	13

[удаление плейлиста]

SQL> delete from playlists where playlists.name = 'mp3';
SQL> select * from playlists;

NAME	DATE_OF_CR
list1	2013-01-09
list3	2014-09-30
list2	2014-03-21
home	2013-01-30
list4	2014-12-15

SQL> select * from pl_tr;

NAME	TR_ID
home	21
home	22
home	23
home	24
home	25
home	26
home	27
home	32
home	33
home	34
list1	1
list1	2
list2	3
list2	4

list2	5
list4	13

3) Создать триггер, реализующий следующее задание:
   При превышении количества продаж альбома более заданной величины повышать рейтинг альбому на фиксированную величину.

Данный триггер будет вызывать хранимую процедуру. Количество продаж, превышение которого говорит о необходимости изменения рейтинга, будем задавать в самом триггере, а значение на которое повышать рейтинг будем задавать как один из аргументов ХП.

```
CREATE OR ALTER trigger trig3 for cds
active after update position 0
AS
begin
  if(new.sold > old.sold)
  then begin
        if(new.sold >= 10000000)
           then execute procedure for_trig(1, new.code);
    end
end

create or alter procedure FOR_TRIG (
    VAL integer not null,
    CODE type of column CDS.CODE not null)
as
begin
  if((select rating from albums, cds_alb, cds
        where albums.alb_id = cds_alb.alb_id and
        cds_alb.code = cds.code and cds.code = :code) <= (5 - :val))
  then begin
        update albums set albums.rating = (albums.rating + :val)
        where albums.alb_id = (select albums.alb_id from
        albums,  cds_alb,  cds where albums.alb_id = cds_alb.alb_id and
        cds_alb.code = cds.code and cds.code = :code);
        end
end
```

Проверка работоспособности:

[исходные таблицы]

SQL> select * from albums;

ALB_ID	ALBUM	RELEASE_YEAR	LABEL	RATING	NUM_OF_RAT	ST_ID	TYPE	GENRE
1	Cowboys from Hell	1990	Atco	5	54	1	album	1
2	Tales From The Thousand Lakes	1994	Relapse	4	37	2	album	1
3	Black Winter Day	1994	Relapse	4	23	2	EP	1
4	Reign In Blood	1986	Def Jam Recordings	5	78	3	album	1
5	Icon	2013	Priority Records	3	48	4	compilation	2
6	Crimson	1996	Black Mark Production	5	18	5	album	1

SQL> select * from cds;

CODE	RELEASE_YEAR	SOLD	STATUS	AWARD
679137227	1990	100000000	3	grammy
781676650	2000	10000000	3	grammy

329751487	1986	200000000	3	grammy
105520746	2013	1700000	3	MTV award
714679272	1996	600000	1	MTV award

[для рейтинга = 3]

SQL> update cds set sold = 10000000 where code = 105520746;
SQL> select * from cds;

CODE	RELEASE_YEAR	SOLD	STATUS	AWARD
679137227	1990	100000000	3	grammy
781676650	2000	10000000	3	grammy
329751487	1986	200000000	3	grammy
105520746	2013	10000000	3	MTV award
714679272	1996	600000	1	MTV award

SQL> select * from albums;

ALB_ID	ALBUM	RELEASE_YEAR	LABEL	RATING	NUM_OF_RAT	ST_ID	TYPE	GENRE
1	Cowboys from Hell	1990	Atco	5	54	1	album	1
2	Tales From The Thousand Lakes	1994	Relapse	4	37	2	album	1
3	Black Winter Day	1994	Relapse	4	23	2	EP	1
4	Reign In Blood	1986	Def Jam Recordings	5	78	3	album	1
5	Icon	2013	Priority Records	4	48	4	compilation	2
6	Crimson	1996	Black Mark Production	5	18	5	album	1

[для максимального значения рейтинга (= 5)]

SQL> update cds set sold = 11000000 where code = 714679272;
SQL> select * from cds;

CODE	RELEASE_YEAR	SOLD	STATUS	AWARD
679137227	1990	100000000	3	grammy
781676650	2000	10000000	3	grammy
329751487	1986	200000000	3	grammy
105520746	2013	1700000	3	MTV award
714679272	1996	11000000	1	MTV award

SQL> select * from albums;

ALB_ID	ALBUM	RELEASE_YEAR	LABEL	RATING	NUM_OF_RAT	ST_ID	TYPE	GENRE
1	Cowboys from Hell	1990	Atco	5	54	1	album	1
2	Tales From The Thousand Lakes	1994	Relapse	4	37	2	album	1
3	Black Winter Day	1994	Relapse	4	23	2	EP	1
4	Reign In Blood	1986	Def Jam Recordings	5	78	3	album	1
5	Icon	2013	Priority Records	3	48	4	compilation	2
6	Crimson	1996	Black Mark Production	5	18	5	album	1

[для количества продаж меньше заданного]

SQL> update cds set sold = 1000000 where code = 105520746;
SQL> select * from cds;

CODE	RELEASE_YEAR	SOLD	STATUS	AWARD
679137227	1990	100000000	3	grammy
781676650	2000	10000000	3	grammy
329751487	1986	200000000	3	grammy

105520746	2013	1000000	3 MTV award
714679272	1996	600000	1 MTV award

SQL> select * from albums;

ALB_ID	ALBUM	RELEASE_YEAR	LABEL	RATING	NUM_OF_RAT	ST_ID	TYPE	GENRE
1	Cowboys from Hell	1990	Atco	5	54	1	album	1
2	Tales From The Thousand Lakes	1994	Relapse	4	37	2	album	1
3	Black Winter Day	1994	Relapse	4	23	2	EP	1
4	Reign In Blood	1986	Def Jam Recordings	5	78	3	album	1
5	Icon	2013	Priority Records	3	48	4	compilation	2
6	Crimson	1996	Black Mark Production	5	18	5	album	1

Как видим, в первом случае было произведено увеличение рейтинга альбома с заданного диска на единицу. В двух других случаях, как неудовлетворяющих условию, изменения записей не произошло.

4. Выводы

В ходе выполнения работы было изучено создание и использование триггеров. Использование триггеров очень удобно для выполнения контроля целостности, т.е. соблюдения условия непротиворечивости данных (отсутствия дубликатов), а также для автоматизированной модификации необходимых данных, что может подстраховать от пользовательских ошибок.

В работе было показано, что с помощью триггеров можно автоматизировать заполнение некоторых полей (например, численное значение первичного ключа), выполнять контроль целостности данных (например, модификация данных в подчиненной таблице при удалении или изменении записи в главной таблице), а также автоматически изменять значения атрибутов записи, которые логически зависят от изменения других атрибутов.