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, вызовы процедур»

Работу выполнила:
Мартюшева Надежда
Группа: 43501/3
Преподаватель:
Мяснов Александр Владимирович

Санкт-Петербург
2017

1 Цели работы

Познакомиться 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.
4. Выложить скрипт с созданными сущностями в svn.
5. Продемонстрировать результаты преподавателю.

3 Индивидуальное задание

1. При добавлении/изменении данных в таблицу результатов проверять на дубль места и соответствия количества очков заносимому месту. При дублях или несоответствии - выбрасывать исключение.
2. При попытке внесения спонсорской помощи для соревнования проверять не вносились ли в течение последнего месяца средства клубам участников соревнования. Если вносились - выбрасывать исключение.

4 Ход работы

Триггрер для автоматического заполнения ключевого поля.

```
1  CREATE SEQUENCE seq_judge_id;
2  ALTER SEQUENCE seq_judge_id RESTART WITH 20;
3
4  set term ^;
5  CREATE OR ALTER TRIGGER trg_judge_id
6  before insert position 0 on judge
7  as
8  begin
9    if ((new.judge_id is null) or (new.judge_id = 0)) then
10   begin
11     new.judge_id = next value for seq_judge_id;
12   end
13 end^
14
15 set term ;^
```

Листинг 1: Триггер для автоматического заполнения ключевого поля

Пример срабатывания триггера: до вставки записи (рисунок 1), после вставки записи (рисунок 2).
Триггрер для контроля целостности в подчиненной таблице при удалении ключа в главной таблице.

```
1  set term ^;
2  CREATE OR ALTER TRIGGER trg_judge_del
3  before delete position 1 on judge
4  as
5  begin
6    delete from competition_judge
7    where judge_id = old.judge_id;
8  end^
9
10 set term ;^
```

Листинг 2: Триггер для контроля целостности данных в подчиненной таблице при удалении записей в главной таблице

```
SQL> select * from judge;

    JUDGE_ID       ASH_ID         RANK  DATE_VALID
============ ============ ============ ==========
           1          627            1 2009-04-04
           2         5993            3 2016-01-29
           3          502            1 2009-01-10
           4         1185            1 2009-07-21
           5         1161            3 2012-09-30
           6         2207            1 2011-12-12
           7         2767            3 2016-02-14

SQL>
```

Рис. 1: Состояние таблицы JUDGE до добавления данных

```
SQL> insert into judge (ASH_ID, RANK, DATE_VALID) values
CON> (10010, 3, '10.12.2016');
SQL> select * from judge;

    JUDGE_ID       ASH_ID         RANK  DATE_VALID
============ ============ ============ ==========
           1          627            1 2009-04-04
           2         5993            3 2016-01-29
           3          502            1 2009-01-10
           4         1185            1 2009-07-21
           5         1161            3 2012-09-30
           6         2207            1 2011-12-12
           7         2767            3 2016-02-14
          21        10010            3 2016-12-10

SQL>
```

Рис. 2: Состояние таблицы JUDGE после добавления данных

```
SQL> select * from competition_judge;

   COMP_H_ID     JUDGE_ID
============ ============
           1           21

SQL> delete from judge where judge_id = 21;
Statement failed, SQLSTATE = 23000
violation of FOREIGN KEY constraint "INTEG_78" on table "COMPETITION_JUDGE"
-Foreign key references are present for the record
-Problematic key value is ("JUDGE_ID" = 21)
```

Рис. 3: Состояние таблицы JUDGE до удаления данных

```
SQL> delete from judge where judge_id = 21;
SQL> select * from competition_judge;
SQL>
```

Рис. 4: Состояние таблицы JUDGE после удаления данных

Пример срабатывания триггера: до удаления записи (рисунок 3), после удаления записи (рисунок 4).

Создать триггрер для контроля целостности в подчиненной таблице при модификации ключа в главной таблице.

```
1  set term ^;
2  CREATE OR ALTER TRIGGER trg_judge_upd
3  after update position 1 on judge
4  as
5  begin
6    update competition_judge
7    set judge_id = new.judge_id
8    where judge_id = old.judge_id;
9  end^
10
11 set term ;^
```

Листинг 3: Триггер для контроля целостности данных в подчиненной таблице при изменении записей в главной таблице

Пример срабатывания триггера: до модификации записи (рисунок 5), после модификации записи (рисунок 6).

```
SQL> select * from judge;

    JUDGE_ID       ASH_ID         RANK  DATE_VALID
============ ============ ============ ==========
           1          627            1 2009-04-04
           2         5993            3 2016-01-29
           3          502            1 2009-01-10
           4         1185            1 2009-07-21
           5         1161            3 2012-09-30
           6         2207            1 2011-12-12
           7         2767            3 2016-02-14

SQL> select * from competition_judge;

   COMP_H_ID     JUDGE_ID
============ ============
           1            7
           1            4
           2            1
           2            6
```

Рис. 5: Состояние таблицы JUDGE до модификации данных

```
SQL> update judge set judge_id = 10 where judge_id = 7;
SQL> select * from judge;

    JUDGE_ID       ASH_ID         RANK  DATE_VALID
============ ============ ============ ==========
           1          627            1 2009-04-04
           2         5993            3 2016-01-29
           3          502            1 2009-01-10
           4         1185            1 2009-07-21
           5         1161            3 2012-09-30
           6         2207            1 2011-12-12
          10         2767            3 2016-02-14

SQL> select * from competition_judge;

   COMP_H_ID     JUDGE_ID
============ ============
           1           10
           1            4
           2            1
           2            6
```

Рис. 6: Состояние таблицы JUDGE после модификации данных

Создать триггер, который при добавлении/изменении данных в таблице результатов проверяет на дубль места и соответствие количества очков заносимому месту. При дублях или несоответствии - выбрасывает исключение.

```
1  CREATE EXCEPTION duplicated 'Plase of dancers is duplicated';
2  CREATE EXCEPTION not_corresponded_points 'The number of points does not correspond to the
       ↪ place occupied by';
3  CREATE EXCEPTION not_raiting_nomination 'Nomination is not rating';
4
5  set term ^;
6  CREATE OR ALTER TRIGGER trg_plase_and_point
7  before insert or update position 0 on competition_result
8  as
9  declare variable cnt_plase int;
10 declare variable right_point int not null;
11 declare variable isRating dom_rating not null;
12 begin
13   select count(*)
14   from competition_result
15   where comp_h_id = new.comp_h_id and nomination_id = new.nomination_id and plase = new.
       ↪ plase
16   into :cnt_plase;
17
18   if (cnt_plase>0) then
19   exception duplicated;
20
21   select is_rating
22   from nomination
23   where nomination_id = new.nomination_id
24   into :isRating;
25
```

```
26    if (new.point>0 and isRating='no') then
27    exception not_raiting_nomination;
28
29    select point
30    from lib_points
31    where num_of_part = new.num_of_part and plase = new.plase
32    into :right_point;
33
34    if (new.point != right_point) then
35    exception not_corresponded_points;
36
37  end^
38
39  set term ;^
```

Листинг 4: Триггер №1

Пример срабатывания триггера: дублирование места (рисунок 7), несоответствие начисленных баллов занимаемому месту (рисунок 8), номинация нерейтинговая (рисунок 9).

```
SQL> insert into competition_result values
CON>    (4, 7, 70, 1, 6, 5993, 6004);
Statement failed, SQLSTATE = HY000
exception 2
-DUPLICATED
-Plase of dancers is duplicated
-At trigger 'TRG_PLASE_AND_POINT' line: 13, col: 20
SQL>
```

Рис. 7: Исключение: дублирование места

```
SQL> insert into competition_result values
CON>    (4, 7, 70, 2, 0, 5993, 6004);
Statement failed, SQLSTATE = HY000
exception 3
-NOT_CORRESPONDED_POINTS
-The number of points does not correspond to the place occupied by
-At trigger 'TRG_PLASE_AND_POINT' line: 29, col: 33
SQL>
```

Рис. 8: Исключение: несоответствие начисленных баллов занимаемому месту

```
SQL> insert into competition_result values
CON>    (2, 1, 70, 1, 3, 9077, 8595);
Statement failed, SQLSTATE = HY000
exception 4
-NOT_RAITING_NOMINATION
-Nomination is not rating
-At trigger 'TRG_PLASE_AND_POINT' line: 21, col: 38
```

Рис. 9: Исключение: номинация нерейтинговая

Создать триггер, который при попытке внесения спонсорской помощи для соревнования проверяет не вносились ли в течение последнего месяца средства клубам участников соревнования. Если вносились - выбрасывает исключение.

```
1  CREATE EXCEPTION exist_subsidy_for_club 'Sponsor aid for competitors clubs was entered
       ↪ during the last month prior to the competition';
2
3  set term ^;
4  CREATE OR ALTER TRIGGER trg_subsidy_for_competition
5  before insert position 0 on sponsor_competition
6  as
7  declare variable club_id int not null;
8  declare variable comp_date date not null;
9  declare variable cnt_sub int;
10 begin
11   select date_start
12   from competition_history
13   where comp_h_id = new.comp_h_id
14   into :comp_date;
15
16   for
17   select club_id
18   from competition_result r
```

```
19    join dancer d on d.ash_id = r.leader_id or d.ash_id = r.partner_id
20    join dancer_club dc on dc.dancer_id = d.dancer_id
21    where r.comp_h_id = new.comp_h_id
22    group by club_id
23    into :club_id
24    do
25    begin
26    select count(*)
27    from sponsor_club sc
28    join subsidy_for_club subc on subc.subsidy_id = sc.subsidy_id
29    where sc.sponsor_id = new.sponsor_id and sc.club_id = :club_id and subc.sub_date between :
        ↪ comp_date-30 and :comp_date
30    into :cnt_sub;
31
32    if (cnt_sub>0) then exception exist_subsidy_for_club;
33    end
34  end^
35
36  set term ;^
```

Листинг 5: Триггер №2

Пример срабатывания триггера: исключение при попытке внести данные в таблицу SPONSOR_COMPETITION (рисунок 10).

```
SQL> select * from subsidy_for_club;

   SUB_DATE      SUBSIDY   SUBSIDY_ID SUBSIDY_H_ID
========== ============ ============ ============
2014-12-05     150000.00            1            1
2015-08-15     240000.00            1            2

SQL> select * from sponsor_club;

  SUBSIDY_ID   SPONSOR_ID      CLUB_ID
============ ============ ============
           1            2          144
           2            1          144

SQL> insert into subsidy_for_club values ('18.11.2016', 5000, 2, 3);
SQL> insert into sponsor_competition values (4, 1, 6);
Statement failed, SQLSTATE = HY000
exception 1
-EXIST_SUBSIDY_FOR_CLUB
-Sponsor aid for competitors clubs was entered during the last month prior to th
e competition
-At trigger 'TRG_SUBSIDY_FOR_COMPETITION' line: 29, col: 17
SQL>
```

Рис. 10: Состояние таблицы JUDGE до модификации данных

5 Вывод

В ходе работы я ознакомиться с возможностями реализации более сложной обработки данных на стороне сервера с помощью триггеров.

Триггер является программой, которая хранится в области метаданных базы данных и выполняется на стороне сервера. Напрямую обращение к триггеру невозможно. Он вызывается автоматически при наступлении события таблицы, связан с одной таблицей/представлением, с одним или более событиями для этой таблицы/представления (INSERT, UPDATE, DELETE) и ровно с одной фазой такого события (BEFORE или AFTER).

С помощью триггеров достигаются следующие цели:

- проверка корректности введенных данных и выполнение сложных ограничений целостности данных, которые трудно, если вообще возможно, поддерживать с помощью ограничений целостности, установленных для таблицы;
- выдача предупреждений, напоминающих о необходимости выполнения некоторых действий при обновлении таблицы, реализованном определенным образом;
- накопление аудиторской информации посредством фиксации сведений о внесенных изменениях и тех лицах, которые их выполнили;
- поддержка репликации.

При условии правильного использования триггеры могут стать очень мощным механизмом. Основное их преимущество заключается в том, что стандартные функции сохраняются внутри базы данных и согласованно активизируются при каждом ее обновлении. Это может существенно упростить приложения. Тем не менее следует упомянуть и о присущих триггеру недостатках:

- сложность: при перемещении некоторых функций в базу данных усложняются задачи ее проектирования, реализации и администрирования;
- скрытая функциональность: перенос части функций в базу данных и сохранение их в виде одного или нескольких триггеров иногда приводит к сокрытию от пользователя некоторых функциональных возможностей, что может стать причиной незапланированных, потенциально нежелательных и вредных побочных эффектов, поскольку в этом случае пользователь не в состоянии контролировать все процессы, происходящие в базе данных;
- влияние на производительность: перед выполнением каждой команды по изменению состояния базы данных СУБД должна проверить триггерное условие с целью выяснения необходимости запуска триггера для этой команды. Выполнение подобных вычислений сказывается на общей производительности СУБД, а в моменты пиковой нагрузки ее снижение может стать особенно заметным. Очевидно, что при возрастании количества триггеров увеличиваются и накладные расходы, связанные с такими операциями.

Неправильно написанные триггеры могут привести к серьезным проблемам, таким, например, как появление "мертвых"блокировок. Триггеры способны длительное время блокировать множество ресурсов, поэтому следует обратить особое внимание на сведение к минимуму конфликтов доступа.