

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе №6

по дисциплине «Базы данных»

SQL-программирование: Триггеры, вызовы процедур

Выполнил

студент гр. 43501/3

Е.А. Никитин

Преподаватель

А.В. Мяснов

«__»_____2015г.

Санкт-Петербург

2015

Цели работы

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

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

Ход работы:

1) Триггер — это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено действием по модификации данных: добавлением `INSERT`, удалением `DELETE` строки в заданной таблице, или изменением `UPDATE` данных в определенном столбце заданной таблицы реляционной базы данных. Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики. Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера. Соответственно, в случае обнаружения ошибки или нарушения целостности данных может произойти откат этой транзакции.

Момент запуска триггера определяется с помощью ключевых слов `BEFORE` (триггер запускается до выполнения связанного с ним события; например, до добавления записи) или `AFTER` (после события). В случае, если триггер вызывается до события, он может внести изменения в модифицируемую событием запись (конечно, при условии, что событие — не удаление записи). Некоторые СУБД накладывают ограничения на операторы, которые могут быть использованы в триггере (например, может быть запрещено вносить изменения в таблицу, на которой «висит» триггер, и т. п.).

Кроме того, триггеры могут быть привязаны не к таблице, а к представлению (VIEW). В этом случае с их помощью реализуется механизм «обновляемого представления». В этом случае ключевые слова `BEFORE` и `AFTER` влияют лишь на последовательность вызова триггеров, так как собственно событие (удаление, вставка или обновление) не происходит.

В некоторых серверах триггеры могут вызываться не для каждой модифицируемой записи, а один раз на изменение таблицы. Такие триггеры называются табличными.

2) Создать триггер для автоматического заполнения ключевого поля.

Был создан триггер для заполнения ключевого поля:

```
create generator gen;  
  
set generator gen to 2147483306;
```

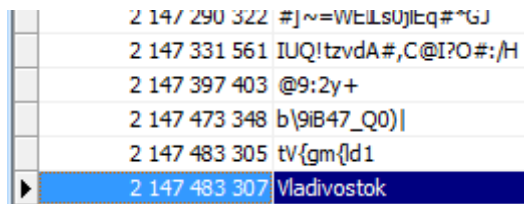
```
create trigger aut_trig for station  
  
before insert  
  
as begin  
  
    NEW.st_id = GEN_ID(gen, 1);  
  
end;
```

Начальная инициализация генератора числом 2147483306 из-за заполнений таблицы в предыдущих экспериментах.

Проверка работы скрипта:

```
insert into station (station.name) values ('Vladivostok');
```

Результат:



2147290322	#J~=WELsUjEq#*GJ
2147331561	IUQ!tzvdA#,C@I?O#:/H
2147397403	@9:2y+
2147473348	b\9IB47_Q0)
2147483305	tV{gm{ld1
2147483307	Vladivostok

3) Создать триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице.

Был создан триггер контроля целостности данных в таблице route_station, которая связывает станции с маршрутами, при модификации или удалении данных из таблицы station, которая хранит набор всех станций и их id: при попытке удаления или изменения названия станции, которая используется, выдается сообщение об особой ситуации.

```
create exception del_mod_err 'Wrond modify! This station is using in table  
route_station!';
```

```
create trigger proverka for station
```

```
before delete or update
```

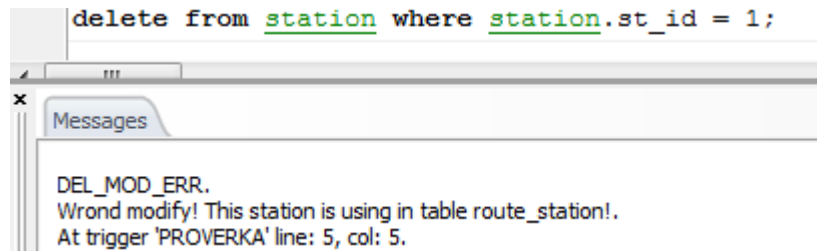
```
as begin
```

```
if(old.st_id in (select route_station.st_id from route_station))

then exception del_mod_err;

end;
```

Проверка работы (попытка удалить из словаря станцию с id=1 – Москва):



4) Создать триггер для проверки участия данного билета в других заказах. Если есть - не добавлять.

Был создан триггер:

```
create exception tick_control 'This ticket already using!';

create trigger add_tick for t_order

before insert

as begin

    if(new.t_num in (select t_order.t_num from t_order

        where new.t_num = t_order.t_num))

    then

        exception tick_control;

    end
```

Для проверки было создано специальное место, которого раньше не было – место в поезде маршрута Москва-СПБ:

PLACE_ID	TRAIN_ID	CARRIAGE	NUM_IN_...	CAR_TYPE...	PLACE_TYPE...
25	75	1	25	0	0
26	75	1	26	0	1
36	78	1	36	0	2

И был создан билет на основе этого места:

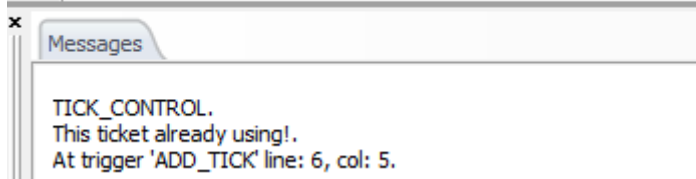
T_NUM	PLACE_ID	STATUS	START_ST	STOP_ST
28	25	buy	Moscow	SPB
29	26	buy	Moscow	SPB
35	128	buy	Moscow	N_Novgorod

До этого в таблице t_order не было записи с t_num=29, попытаемся добавить данный билет в заказы:

ORDER_ID	T_NUM	TICKETS_N...
1	29	1

Билет был добавлен. Теперь попробуем добавить новый заказ, изменим номер заказа на order_id = 2, но номер билета оставим прежний:

```
insert into t_order values (2, 29, 1);
```



В результате вышло сообщение о ситуации ошибочного добавления заказа (данный билет уже используется) и заказ не был добавлен:

ORDER_ID	T_NUM	TICKETS_N...
1	29	1
1 234	44	1
4 523	28	1
5 534	35	1
7 546	88	1
7 675	59	1
7 676	89	1
8 954	75	1

5) Создать триггер для проверки валидности станций отправления и назначения (принадлежат ли маршруту и в правильном ли порядке указаны). Если есть ошибки - не добавлять.

Была написана ХП, которая возвращает номер маршрута, которому принадлежит место в создаваемом билете:

```

create procedure find_route(p_id int)

returns(r_id int)

as begin

for

    select shedule.route_id from shedule, place where

        shedule.train_id = place.train_id and place.place_id = :p_id

into :r_id do suspend;

end;

```

Был создан триггер:

```

create exception valid_warning 'Wrong stations in ticket!';

create trigger valid_st for ticket

before insert

as

declare variable r_id int;

declare variable sstop_id int;

declare variable sstart_id int;

begin

    execute procedure find_route(new.place_id)

    returning_values r_id;

    select station.st_id from station where station.name = new.start_st

```

```

into :sstart_id;

select station.st_id from station where station.name = new.stop_st

into :sstop_id;

if (

(:sstop_id not in (select route_station.st_id from route_station where

route_station.route_id = :r_id))

or

(:sstart_id not in (select route_station.st_id from route_station where

route_station.route_id = :r_id))

or

((select route_station.num_in_route from route_station where

route_station.st_id = :sstart_id and route_station.route_id = :r_id) >

(select route_station.num_in_route from route_station where

route_station.st_id = :sstop_id and route_station.route_id = :r_id))

) then exception valid_warning;

end;

```

Для проверки был выбран маршрут с id=29 Москва-СПБ.

Станции Москва (st_id=1) и СПб (st_id=2) в таблице route_station, связывающей id станций и маршруты, и хранящая функцию станции и номер в маршруте.

	NUM_IN_RO...	ROUT... ▲	ST_ID	ST_FUNC
▶	1	29	1	start
	2	29	2	finish

Для него создано место, которого раньше не было:

PLACE_ID	TRAIN_ID	CARRIAGE	NUM_IN_...	CAR_TYPE...	PLACE_TYPE...
24	75	1	24	0	3

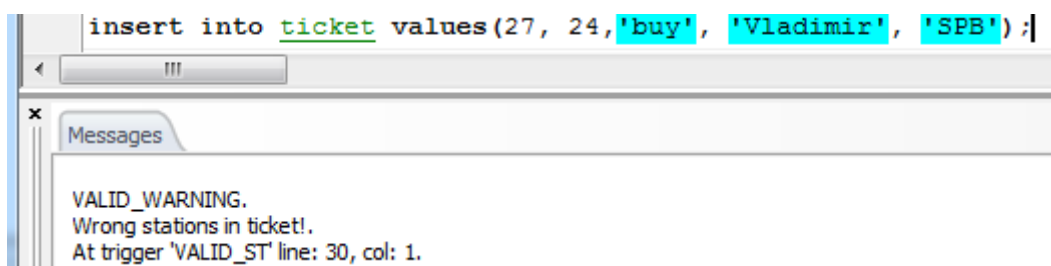
Попробуем создать билет:

```
insert into ticket values(27, 24,'buy', 'Moscow', 'MSPB');
```

T_NUM	PLACE_ID	STATUS	START_ST	STOP_ST
27	24	buy	Moscow	SPB
28	25	buy	Moscow	SPB
29	26	buy	Moscow	SPB

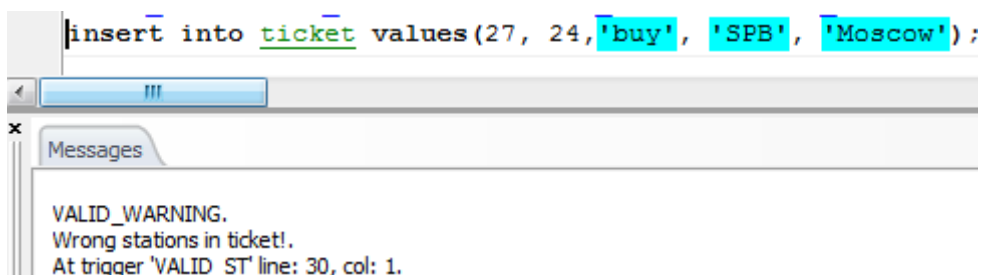
Как видно, билет с номером 27 успешно создан.

Теперь попробуем создать билет, где для данного маршрута неправильная станция (вместо Москвы – Владимир):



Билет не добавился и получено сообщение о неправильных станциях.

Теперь попробуем создать билет, где станции верные, но стоят в неправильном порядке:



Билет не добавился и получено сообщение о неправильных станциях.

Вывод.

В ходе лабораторной работы были изучены триггеры в SQL

Триггером называлась хранящаяся в базе данных процедура, автоматически вызываемая СУБД при возникновении соответствующих условий. При определении триггера указывались два условия его применимости – общее

условие (имя отношения и тип операции манипулирования данными) и конкретное условие (логическое выражение, построенное по правилам, близким к правилам ограничений целостности), а также действие, которое должно быть выполнено над БД при наличии условий применимости. Триггеры - инструмент, позволяющий выполнять различные проверки автоматически, что необходимо в больших реальных БД. Триггеры позволяют предотвратить добавление данных, которые не удовлетворяют условию.