Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Базы данных

Отчет по лабораторной работе №6
SQL-программирование: Триггеры, вызовы процедур

Работу выполнила:
Темнова А.С.
Группа: 43501/3
Преподаватель:
Мяснов А.В.

Санкт-Петербург
2016

1 Цель работы

Познакомить студентов 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 Ход выполнения работы

Триггер – это подпрограмма, похожая на процедуру БД, автоматически вызываемая СУБД при изменении, удалении или добавлении записи в таблице.

3.1 Триггер для автоматического заполнения ключевого поля

Создадим тестовую таблицу my_wine с атрибутами my_wine_id, wine_number (тип - integer).

```
1  connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';
2
3  create table my_wine (
4    my_wine_id       integer   not null,
5    wine_number      integer   not null
6  );
7
8  alter table my_wine add constraint pk_my_wine primary key (my_wine_id);
```

Листинг 1: создание тестовой таблицы

Теперь создадим генератор my_gen:

```
1  create sequence my_gen;
2  alter sequence my_gen restart with 0;
```

Листинг 2: создание генератора

Затем создадим триггер, который будет использовать этот генератор. Этот триггер автоматически заполнит поле my_wine_id таблицы my_wine.

```
1  connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';
2
3  drop trigger trig_auto_my_wine_id;
4  create trigger trig_auto_my_wine_id for my_wine
5  active before insert
6  as
7  begin
8    if(new.my_wine_id is null) then
9      new.my_wine_id = gen_id( my_gen, 1 );
10 end;
```

Листинг 3: создание триггера

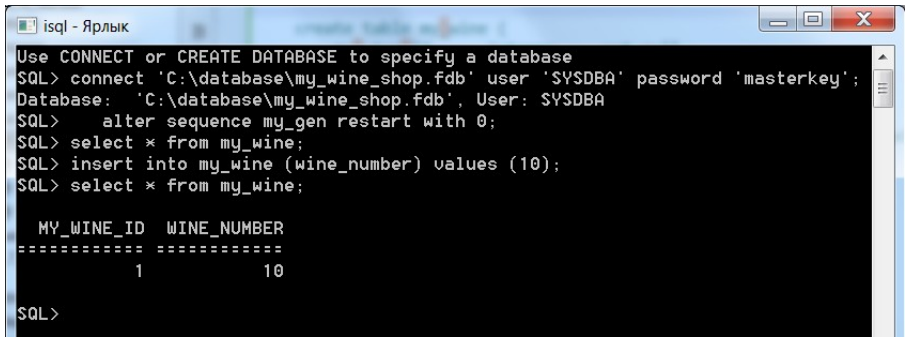

Рис. 1: Демонстрация работы триггера

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

```
1  connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';
2
3  drop trigger modify_wine;
4  drop exception ex_no_modify;
5
6  create exception ex_no_modify 'This wine in other tables';
7  create trigger modify_wine for wine
8  before delete or update
9  as
10 begin
11   if(old.wine_id in
12   (select producer_wine.wine_id from producer_wine) )
13   then
14     exception ex_no_modify;
15 end;
```

Листинг 4: создание триггера

Собственно проверяем, если в producer_wine есть записи о том, что кто-то производит это вино, то удалять инфу о вине нельзя.

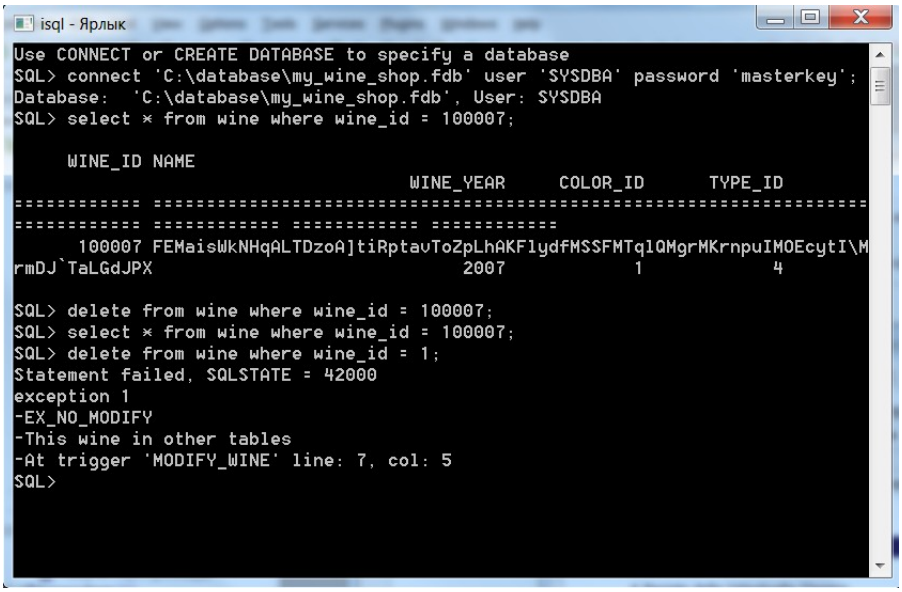

Рис. 2: Демонстрация работы триггера

Вина с id = 100007 нет в таблице продюсер-вино, поэтому эта запись преспокойно удалилась при соответствующем запросе.

Вино с id = 1 присутствует в таблице продюсер-вино, поэтому при запросе об удалении этой записи из таблицы вина, нам вывалился эксепшн.

3.3 Триггер в соответствии с индивидуальным заданием, полученным у преподавателя

При оформлении заказа клиентом проверять наличие на складах магазина достаточного количества вина. При недостатке - выбрасывать исключение.

```
1  connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';
2
3  drop trigger create_order;
4  drop exception ex_no_quantity;
5
6  create exception ex_no_quantity 'There is no necessary quantity of goods';
7
8  set term ^;
9  create trigger create_order for purchase
10 active before insert
11 as
12 declare variable storage_quantity integer;
13 begin
14   select storage.number into :storage_quantity from storage where storage.storage_id = :new.
     ↪ storage_id;
15   if(new.number > storage_quantity) then
16     ex_no_quantity;
17 end^
18 set term ; ^
```

Листинг 5: создание триггера indiv1

Собственно вытаскиваем сколько у нас на складе есть, если просят больше, то выкидываем исключение.

3.4 Триггер в соответствии с индивидуальным заданием, вызывающий хранимую процедуру

При добавлении/изменении проверять дубли в таблице связи производителей и вина. При наличии дублей - выбрасывать исключение.

```
1  connect 'C:\database\my_wine_shop.fdb' user 'SYSDBA' password 'masterkey';
2
3  set term ^;
4
5  create procedure IS_EXIST_RECORD (wine int, producer int)
6  returns (isexist int)
7  as
8  begin
9    select count(*) from producer_wine
10   where producer_id=: producer and wine_id=:wine
11   into : isexist;
12   do suspend;
13 end^
14
15 drop trigger create_producer_wine;
16 drop exception ex_isexist;
17
18 create exception ex_isexist 'The duplicating record';
19
20 create trigger create_producer_wine for producer_wine
21 active before insert
22 declare variable ex integer;
23 as
24 begin
25   execute procedure IS_EXIST_RECORD new.wine_id new.producer_id
26   RETURNING_VALUES ex;
27   if( ex > 0 ) then
28     ex_isexist;
29 end
30 ^
31 set term ; ^
```

Листинг 6: создание триггера indiv2

Процедура вытаскивает количество записей, где продюсер равен чему-то и вино равно чему-то. По факту там либо ноль, либо один (когда такая запись есть).

В триггере вызываем эту процедуру для вина и продюсера, связь которых хотим вставить. Если такая запись уже существует, то выбрасываем исключение.

4 Выводы

В ходе данной работы я познакомилась с реализацией триггеров.

Было создано несколько стандартных триггеров, а так же реализованы триггеры в соответствие с индивидуальным заданием.

Преимущества использования триггеров:

1. целостность данных перемещается с уровня логики на уровень данных, где она и должна быть;
2. автоматизация заполнения таблиц бд (если в одной из таблиц появляется какая-либо запись, то триггер способен создать запись в другой таблице, изменить запись, или изменить данные);
3. предотвращение добавления данных, которые не вписываются в логику созданной БД (контроль корректности вводимых данных непосредственно в момент их добавления).

Недостатки использования триггеров:

1. сложность (размещение некоторых действий над данными в БД усложняет ее проектирование, реализацию и администрирование);
2. скрытность функциональных возможностей от пользователя (трудно производить модернизацию приложения, когда скрыты некоторые возможности);
3. влияние на производительность (при большом кол-ве триггеров производительность стремительно уменьшается).