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 О.В. Горемыкина

Руководитель А.В. Мяснов

“ ” 2016 г.

Санкт -Петербург

2016

1. Цели работы

Познакомить студентов с возможностями реализации более сложной обработки данных на стороне сервера с помощью 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 работы

3.1. Триггер для автоматического заполнения поля id_op в таблице ExtraOption

```
CREATE generator increment^
CREATE OR ALTER TRIGGER auto_gen FOR ExtraOption BEFORE INSERT
AS
BEGIN
 new.id_op = gen_id(increment,1);
END^
```

Установим генератор на 7, так как в таблице уже есть 7 записей. Добавим запись в таблицу и посмотрим на результат:

```
SQL> set generator increment to 7;
SQL> insert into extraoption values (777,'testop','0');
SQL> select * from extraoption;

       ID_OP OPTIONTYPE                                          OPTIONPRICE
============ ================================================== ===============
           1 Winter set                                            500.00000
           2 Fog lights                                            200.00000
           3 Audiosystem                                           150.00000
           4 Center armrest                                        50.000000
           5 Rear Parking Sensors                                  300.00000
           6 Mazda Navigation System                               500.00000
           7 Remote Engine Start                                   700.00000
           8 testop                                                0.0000000
```

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

```
CREATE EXCEPTION ERROR_STAGE 'ERROR: CANNOT DELETE STAGE TYPE'^
CREATE OR ALTER TRIGGER check_stage FOR optionlist BEFORE DELETE OR UPDATE
AS
BEGIN
 IF (OLD.id_sale IN (SELECT id_sale FROM sale)) THEN
 EXCEPTION ERROR_STAGE;
END^
```

Попробуем удалить данные из главной таблицы:

```
SQL> delete from optionlist where id_oplist = 2;
Statement failed, SQLSTATE = HY000
exception 3
-ERROR_STAGE
-ERROR: CANNOT DELETE STAGE TYPE
-At trigger 'CHECK_STAGE' line: 4, col: 49
```

4. Индивидуальное задание

4.1. При добавлении опции в заказ проверять наличие опции в комплектации. Если есть - не добавлять.

```
CREATE EXCEPTION WARNING 'OPTION IS ALREADY EXIST'^
CREATE OR ALTER TRIGGER add_op FOR optionlist BEFORE insert
AS
BEGIN
 IF (new.id_op IN (
        SELECT id_op
        FROM comp_option as co
        join complectation using (id_comp)
        join modelcomplectation using (id_comp)
        join car_available using (id_mc)
        join sale using (id_car)
        where new.id_sale = sale.id_sale
        )
 ) THEN
 EXCEPTION WARNING;
END^
```

Опция 6 успешно добавлена во второй заказ, так как у данного автомобиля нет дополнительных опций в комплектации.

```
SQL> insert into optionlist values (131,6,2);
SQL> select id_op from comp_option
    join complectation using (id_comp)
    join modelcomplectation using (id_comp)
    join car_available using (id_mc)
    join sale using (id_car)
CON> where id_sale = 2;
SQL>
```

При попытке добавить опцию 6 в 1 заказ вызывается исключение:

```
SQL> insert into optionlist values (130,6,1);
Statement failed, SQLSTATE = HY000
exception 22
-WARNING
-OPTION IS ALREADY EXIST
-At trigger 'ADD_OP' line: 15, col: 5

SQL> select id_op from comp_option
    join complectation using (id_comp)
    join modelcomplectation using (id_comp)
    join car_available using (id_mc)
    join sale using (id_car)
CON> where id_sale = 1;

       ID_OP
============
           6
           7
```

4.2. При заказе на суммарную стоимость более заданной добавлять в заказ бесплатно наиболее популярную доп. опцию за предыдущий месяц. Если такая опция уже есть в комплектации - добавлять следующую по популярности и т.д.

Для реализации задачи были созданы следующие структуры:

Хранимая процедура выбора month_options(id_sale int) возвращает список дополнительных опций, заказанных в течение месяца, отсортированный по количеству заказов каждой из опций. Если за месяц не было продано ни одной опции, вызывается исключение `ex_monthOption`.

```
-- Getting popular month options
CREATE OR ALTER EXCEPTION ex_monthOption 'not found any options for this month
'^
CREATE OR ALTER PROCEDURE month_options(id_sale int)
returns (id_opt int)
as
    declare variable mon int;
    declare variable yea int;
begin
    id_opt = null;
    -- sale month and year
    select extract(MONTH from sale.dat),extract(YEAR from sale.dat)
    from sale
    where id_sale = :id_sale
    into :mon, :yea;
    mon = mon - 1;
```

```
-- all options for month
    for select * from
        (
            -- comp options for month
            select id_op
            from comp_option
            join complectation using (id_comp)
            join modelcomplectation using (id_comp)
            join car_available using (id_mc)
            join sale using (id_car)
            where extract(MONTH from sale.dat) >= :mon
            and extract(YEAR from sale.dat) = :yea

            union all

            -- sale options for month
            select id_op
            from optionlist
            join sale using (id_sale)
            where extract(MONTH from sale.dat) >= :mon
            and extract(YEAR from sale.dat) = :yea
        )
        group by id_op
        order by sum(id_op) desc
        into :id_opt
    do begin
        suspend;
    end
    if (id_opt = null) then exception ex_monthOption;
end^
```

ХП выбора sales_options(id_sale int) возвращает список опций, уже имеющихся в заказе и комплектации автомобиля.

```
-- Getting options of purchased car
CREATE OR ALTER PROCEDURE sales_options(id_sale int)
returns (id_opt int)
as
begin
    for
        -- comp options
        select id_op
        from comp_option
        join complectation using (id_comp)
        join modelcomplectation using (id_comp)
        join car_available using (id_mc)
        join sale using (id_car)
        where id_sale = :id_sale
        union
        -- sale options
        select id_op
        from optionlist
        join sale using (id_sale)
        where id_sale = :id_sale
        into :id_opt
    do begin
        suspend;
    end
end ^
```

Исполняемая ХП sales_options(id_sale int) получает список популярных за месяц опций и список опций купленного авто, после чего добавляет в список заказанных опций первую популярную опцию, которая еще не была заказана. Если все опции, купленные за прошлый месяц, уже присутствуют в заказе, вызывается исключение ex_carOption 'unable to add an option in order '.

```
CREATE OR ALTER EXCEPTION ex_carOption 'unable to add an option in order '^
-- Adding extra option
create or alter procedure add_free_op(id_sale int)
as
    declare variable added_option int;
begin
    added_option = -1;

    select first 1 id_opt
    from month_options (:id_sale)
    where id_opt not in
    (select * from sales_options(:id_sale))
    into :added_option;

    if (:added_option = -1) then exception ex_carOption;

    INSERT INTO optionlist (id_oplist, id_op, id_sale)
    select max(optionlist.id_oplist)+1, :added_option, :id_sale
    from optionlist;
end^
```

Триггер add_free_op проверяет общую стоимость нового заказа, и, если она больше заданной, вызывает процедуру добавления подарочной опции.

```
create or alter trigger add_free_op for sale after insert
as
declare variable lim_price int;
declare variable summ int;
begin
lim_price = 10000;
if (new.totalprice > lim_price) then
    execute procedure add_free_op(new.id_sale);
end^
```

Получим текущий список связей заказов с опциями:

```
SQL> select * from optionlist;

   ID_OPLIST        ID_OP      ID_SALE
============ ============ ============
...
         124            4            9
         125            3           23
         126            2            9
         127            2           25
         128            2          128
         129            2           65
```

Добавим новый заказ на сумму более 10000 и снова получим список:

```
SQL> insert into sale values (130, '22.12.2016',13,8,12000,'22.12.2020');
SQL> select * from optionlist;

   ID_OPLIST        ID_OP      ID_SALE
============ ============ ============
...
         124            4            9
         125            3           23
         126            2            9
         127            2           25
         128            2          128
         129            2           65
         130            6          130
```

В список была автоматически добавлена опция 6.

Получим список популярных за месяц опций:

```
SQL> select * from month_options (130);

      ID_OPT
============
           6
           7
           5
           1
```

Получим полный список опций из нового заказа:

```
SQL> select * from sales_options (130);

      ID_OPT
============
           1
           6
```

Здесь видим добавленную автоматически опцию 6 и опцию 1 из комплектации.

Будем вызывать процедуру добавления подарочной опции, пока не получим исключение:

```
SQL> execute procedure add_free_op(130);
SQL> execute procedure add_free_op(130);
SQL> execute procedure add_free_op(130);
Statement failed, SQLSTATE = HY000
exception 253
-EX_CAROPTION
-unable to add an option in order
-At procedure 'ADD_FREE_OP' line: 19, col: 1
```

Получим текущий список опций заказа:

```
SQL> select * from sales_options (130);

      ID_OPT
============
           1
           5
           6
           7
```

Таким образом, были добавлены все популярные за месяц опции.

5. Выводы

Триггеры - ключевые элементы среди возможностей, предоставляемых Firebird для централизованной реализации бизнес-правил внутри системы управления базой данных. Триггер является автономным модулем, который выполняется автоматически, когда выполняется запрос, который будет изменять состояние данных в таблице.

С помощью триггеров выдавались сообщения (предупреждения) о том, что необходимо выполнить некоторые действия при изменении таблиц. Также триггеры удобно использовать для оповещения об изменении данных в таблицах. С помощью триггеров можно накладывать ограничения на вносимые данные согласно требованиям предметной области БД.