Санкт-Петербургский Государственный Политехнический Университет

Институт Информационных Технологий и Управления

Кафедра Компьютерных Систем и Программных Технологий

**ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №6**

Дисциплина: **Базы данных**

Тема: **Триггеры**

Н.В. Тышковец

А.В. Мяснов

Выполнил студент гр. № 43501/1

Преподаватель

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

*Индивидуальное задание:*

Реализовать триггеры:

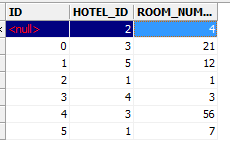
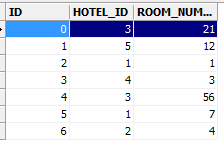
1. При добавлении в заказ нового человека проверять наличие такого человека в заказе. Если есть - не добавлять.
2. Если человек за последний заданный период времени принес более заданной суммы - дарить ему бесплатно наименее популярный тур.

**Ход работы**

1. Был создан триггер для автоматического заполнения поля id в таблице ROOM

|  |
| --- |
| CREATE OR ALTER trigger AUTOINCR for room active before insert AS  begin  select max(room.id) from room into NEW.id;  NEW.id = NEW.id + 1;  end; |

Результат работы триггера:

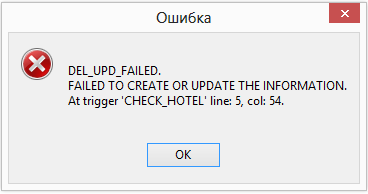
 

При добавлении строчки в таблицу ROOM, без указания ID, в итоге в любом случае ID заполняется значением, равным максимальному значению ID+1.

Затем был создан триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице ROOM

|  |
| --- |
| create or alter exception DEL\_UPD\_FAILED 'FAILED TO CREATE OR UPDATE THE INFORMATION';  CREATE OR ALTER trigger check\_hotel for room  active before delete or update  AS  begin  if (OLD.ID in (select room\_id from hotel)) then  exception DEL\_UPD\_FAILED;  end; |

При попытке удалить строку в ROOM выбрасывается исключение:



1. При добавлении в заказ нового человека проверять наличие такого человека в заказе. Если есть - не добавлять.

Для создания данного триггера предварительно создадим исключение:

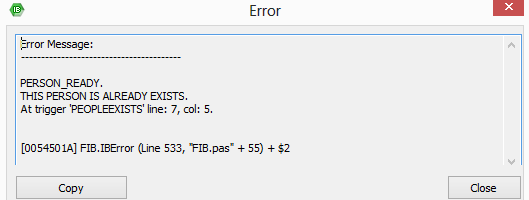
|  |
| --- |
| CREATE EXCEPTION PERSON\_READY 'THIS PERSON IS ALREADY EXISTS'; |

Триггер:

|  |
| --- |
| CREATE OR ALTER TRIGGER PEOPLEEXISTS FOR ORDER\_TOUR  ACTIVE BEFORE INSERT OR UPDATE  AS  begin  IF(NEW.cl\_id in (select cl\_id from order\_tour  where cl\_id = NEW.cl\_id and new.date\_in between date\_in and date\_in+(select duration from tour where id=new.tour\_id)))  then  exception PERSON\_READY;  END; |

При попытке создать очередной заказ в таблице ORDER\_TOUR с CL\_ID уже находящегося в туре клиента, выскакивает исключение.





1. Если человек за последний заданный период времени принес более заданной суммы - дарить ему бесплатно наименее популярный тур.

Была дополнительно создана таблица TOUR\_ACTION, содержащая туры и их стоимость по акциям. Также в таблице CLIENT создано поле, содержащее ссылку на TOUR\_ACTION для данного клиента.

При добавлении записи в заказ клиента, который принес меньше 10000, запись успешно добавляется.

При добавлении записи в заказ клиента, который принес больше 10000, последовательно проверяем, есть ли у клиента возможный тур по акции. Если нет, то дарим клиенту бесплатно наименее популярный тур (добавляем в таблицу TOUR\_ACTION номер наименее популярного тура и стоимость заказа по акции) если он еще не подарен.

Если есть, то проверяем, совпадает ли тур, который заказывает клиент, туру по акции.

Если совпадает, то обнуляем стоимость заказа клиента.

Далее удаляем тур по акции для клиента из таблицы TOUR\_ACTION.

Если не совпадает – оставляем все без изменений.

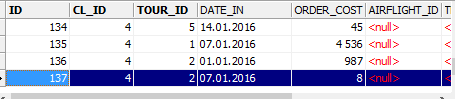
Листинг триггера:

|  |
| --- |
| CREATE OR ALTER TRIGGER FREE\_TOUR FOR ORDER\_TOUR  ACTIVE AFTER INSERT POSITION 0  as  declare variable porog int;  declare variable summa int;  begin  porog = 10000;  summa = 0;  --Если сумма больше порогового значения, то вызываем проц.  select sum(order\_tour.order\_cost) from order\_tour  where order\_tour.cl\_id = new.cl\_id and  date\_in between '2014-09-01' and new.date\_in  group by cl\_id into :summa;  if (summa > porog) then  execute procedure temp(new.id, new.cl\_id );  end |

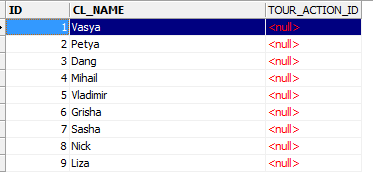
Листинг процедуры:

|  |
| --- |
| create or alter procedure TEMP (  ORDER\_ID integer,  CLIENT\_ID integer)  as  declare variable PER integer; --наименее поп. тур  declare variable CL\_ID integer; --id в TOUR\_ACTION  declare variable PER2 integer;  declare variable COST integer; --стоимость по акции  declare variable TA\_ID integer;  declare variable TA\_T\_ID integer;  declare variable TA\_C integer;  begin  per=0;  cost=0;  --Вычисляем ID для добавления записи в TOUR\_ACTION  select max(tour\_action.id) from tour\_action into :cl\_id;  if(cl\_id is null)  then cl\_id=1;  else  begin  cl\_id=cl\_id+1;  end  --Вычисляем наименее популярный тур  select first 1 tour.id, count(order\_tour.id) from tour, order\_tour where order\_tour.tour\_id=tour.id  group by tour.id order by count(order\_tour.id) into :per, :per2 ;  /\*\*/  /\*Uznaem, est li y clienta touri po akcii\*/  select first 1 tour\_action.id, tour\_action.tour\_id,tour\_action.cost from tour\_action, client, order\_tour  where  client.id=:client\_id and  client.tour\_action\_id=tour\_action.id  into :ta\_id, :ta\_t\_id, :ta\_c;  per2=null;  --  select order\_tour.tour\_id from order\_tour where order\_tour.id=:order\_id and order\_tour.tour\_id=:ta\_t\_id into :per2;  --Если тур в заказе совпадает с туром по акции, то обнуляем --стоимость заказа, удаляем акцию из TOUR\_ACTION  if(per2 is not null) then  begin  update order\_tour set order\_cost=:ta\_c where order\_tour.id=:order\_id;  update client set tour\_action\_id=null where client.id=:client\_id;  delete from tour\_action where tour\_action.id=:ta\_id;  end  else  --Иначе, если в TOUR\_ACTION нет акции, добавим ее  begin  select tour\_action.id from tour\_action where id=1 into per2;  if(per2 is null) then  begin  insert into tour\_action values ( :cl\_id,:per,:cost);  update client set tour\_action\_id=:cl\_id where id=:client\_id;  end  end  end |

Данные в таблицах перед добавлением заказа:



*Рис. 3.1. Таблица ORDER\_TOUR*

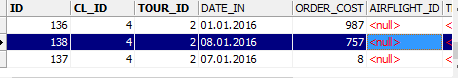


*Рис. 3.2. Таблица CLIENT*



*Рис. 3.3. Таблица TOUR\_ACTION*

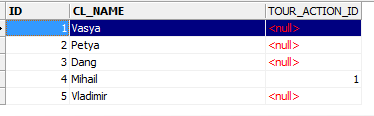
Добавим заказ клиента, принесшего более 10000 (например, id=4)



*Рис. 3.4. Таблица ORDER\_TOUR*

После добавления, срабатывает триггер.

Таблицы после срабатывания триггера:



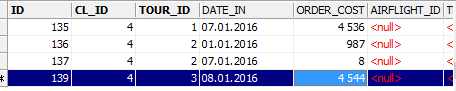
*Рис. 3.5. Таблица CLIENT после срабатывания триггера*



*Рис. 3.6. Таблица TOUR\_ACTION после срабатывания триггера*

В таблицу TOUR\_ACTION добавлен номер наименее популярного тура и стоимость его по акции.

После добавления заказа тем же клиентом, бесплатный тур не добавляется:

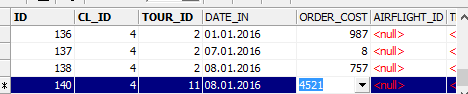


*Рис. 3.7. Таблица ORDER\_TOUR*

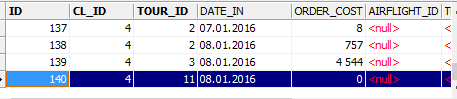


*Рис. 3.8. Таблица TOUR\_ACTION*

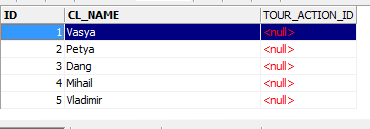
Заказ клиентом тура по акции:



*Рис. 3.9. Таблица ORDER\_TOUR перед подтверждением транзакции*



*Рис. 3.10. Таблица ORDER\_TOUR после добавления записи*



*Рис. 3.11. Таблица CLIENT*



*Рис. 3.12. Таблица TOUR\_ACTION*

Как видно из таблиц стоимость заказа обнулилась, а самого тура по акции у клиента больше нет.

**Вывод**

В данной работе мы познакомились с реализацией триггеров. Было создано несколько стандартных триггеров, а так же реализованы триггеры в соответствие с индивидуальным заданием.

Триггеры предоставляют дополнительные возможности по взаимодействию с данными, позволяя дополнительно контролировать операции создания, изменения и удаления записей в таблицах базы автоматически.

Одним из главных преимуществ триггеров, является возможность контроля целостности базы данных любой сложности. Чем сложнее база, тем применение триггеров более удобно и оправдано.

К недостаткам триггеров можно отнести загруженность системы при вставке/обновлении большого числа записей. Так как, например, при вставке 1000 записей в таблицу, для которой определен триггер BEFORE и AFTER INSERT, они выполнятся соответствующее число раз. А если них содержится еще и обновление различных полей, то процедура вставки может занять много времени.