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»

«Триггеры»

Работу выполнил студент группы №43501/1 Кричевский В.А. \_\_\_\_\_\_\_\_\_

Работу принял 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.

1. Программа работы
2. 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
3. Создать триггер в соответствии с **индивидуальным заданием**, полученным у преподавателя
4. Создать триггер в соответствии с **индивидуальным заданием**, вызывающий хранимую процедуру
5. Выложить скрипт с созданными сущностями в svn
6. Продемонстрировать результаты преподавателю
7. Ход работы
8. Триггер для автоматического заполнения ключевого поля в таблице client

|  |
| --- |
| CREATE SEQUENCE AUTOINC;  ALTER SEQUENCE AUTOINC RESTART WITH 705;  SET TERM ^ ;  CREATE OR ALTER TRIGGER AUTOID\_CLI FOR CLIENT  ACTIVE BEFORE INSERT POSITION 0  AS  begin  if(new.id is null) then  new.id=gen\_id(autoinc,1);  end  ^  SET TERM ; ^ |

При внесении новой записи в таблицу CLIENT, не задав уникального ключа ID, триггер автоматически устанавливает новый ID, используя генератор. В случае, если создаваемый ID уже существует в таблице, триггер создаёт новый до тех пор, пока не достигнет уникального.

1. Триггер для контроля целостности данных в таблицах room и client при удалении/изменении записей в таблице worker.

|  |
| --- |
| CREATE EXCEPTION modify\_no 'This client in other tables';  CREATE TRIGGER tr\_modif\_client\_id FOR client  BEFORE delete or update  AS  BEGIN  if(old.id  in (select room.id from room  UNION  select client.id from client))  then exception modify\_no;  END; |

1. Сделать невозможным бронирование заказа с дублем: с одинаковым клиентом и туром

|  |
| --- |
| CREATE EXCEPTION dupZakaz 'Duplicate order detected!';  SET TERM ^;  CREATE TRIGGER noDupZakaz BEFORE INSERT ON Zakaz AS  DECLARE VARIABLE cnt int;  BEGIN  SELECT count(\*) FROM Zakaz WHERE (Tour\_id = NEW.Tour\_id) AND (Client\_id = NEW.Client\_id)  INTO cnt;  if(cnt <> 0) then EXCEPTION dupZakaz;  END^  SET TERM ;^ |

Проведен эксперимент по проверке работоспособности триггера

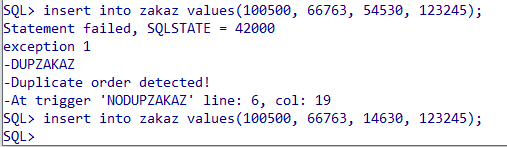


Рис.3. Проверка возможности добавления заказов с дублем

1. Не давать возможности клиенту купить тур более 3х раз.

Код триггера:

|  |
| --- |
| CREATE EXCEPTION limZakaz 'This client has reached maxium order limit!';  SET TERM ^;  CREATE TRIGGER limZakaz BEFORE INSERT ON Zakaz AS  DECLARE VARIABLE cnt int;  BEGIN  SELECT count(\*) FROM Zakaz WHERE Client\_id = NEW.Client\_id INTO cnt;  if(cnt = 3) then EXCEPTION limZakaz;  END^  SET TERM ;^ |

Попытаемся добавить заказ клиенту, у который уже купил три тура:

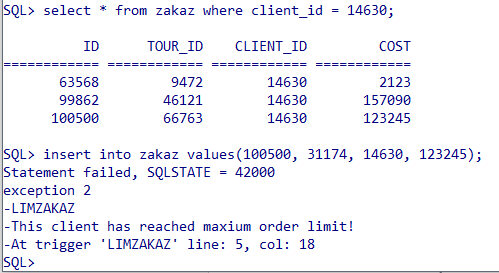


Рис.6. Проверка триггера

Выводы

В данной работе мы познакомились с триггерами. На мой взгляд, триггеры очень сильный инструмент для управления базами данных. Триггеры позволяют автоматизировать процесс заполнения базы данных, так как большинство таблиц в БД связаны. Если в одной из таблиц появляется какая-либо запись, то триггер способен создать запись в другой таблице, изменить запись, или изменить данные. Триггеры позволяют предотвратить добавление данных, которые не вписываются в логику созданной БД. Одной из главных особенностей триггеров является их асинхронность. Они обеспечивают моментальную реакцию системы на какие-либо манипуляции с БД. Это позволяет контролировать корректность вводимых данных непосредственно в момент их добавления. Что исключает необходимость периодических проверок корректности данных в БД.