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. 43501/3 Никитенко А.П.

(подпись)

Руководитель А.В. Мяснов

(подпись)

“ ” 2016 г.

Санкт-Петербург

2016

**Цель 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

**Выполнение работы**

Был создан триггер для автоматического заполнения поля emp\_id в таблице employee:

CREATE GENERATOR gen\_emp\_id;

SET GENERATOR gen\_emp\_id TO 5000;

CREATE TRIGGER trig\_emp\_id FOR employee ACTIVE

BEFORE INSERT

AS

BEGIN

if(new.emp\_id is null) then

new.emp\_id=gen\_id(gen\_emp\_id,1);

END;

Проверим работу триггера:

execute procedure insert\_employee(null,'ewsafd','qwefaf','89117812222','raeg','09:21:21','21:13:45');

execute procedure insert\_employee(null,'9867iu','qwefaf','89117812222','raeg','09:21:21','21:13:45');

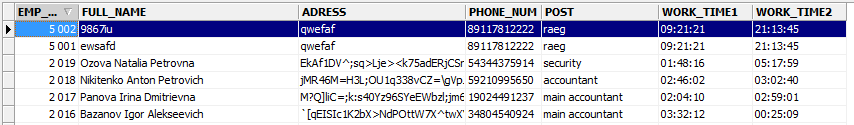


Рис.1 Результат работы триггера

Был создан триггер для контроля целостности данных в таблицах periodadmission и temporaryadmission при изменении записей в таблице employee:

DROP TRIGGER delmod\_emp;

DROP exception except\_mod\_emp;

CREATE EXCEPTION except\_mod\_emp 'This employee uses in other tables';

CREATE TRIGGER delmod\_emp FOR employee

BEFORE delete or update

AS

BEGIN

if(old.emp\_id

in (select temporaryadmission.emp\_id from temporaryadmission

UNION

select periodadmission.emp\_id from periodadmission))

then exception except\_mod\_emp;

END;

Проверим работу триггера:

delete from employee where emp\_id=1;



Рис.2 Результат работы триггера

Выполнение индивидуальных заданий:

1. Не фиксировать фактическое посещение, если сотрудник не имеет права доступ в момент фиксации.

CREATE EXCEPTION except\_no\_access 'This employee has no access to this room now';

drop trigger for\_fact\_vis;

CREATE OR ALTER TRIGGER for\_fact\_vis for factualvisit

ACTIVE BEFORE INSERT

AS

BEGIN

if (new.emp\_id not in

(select emp\_id

from temporaryadmission

where new.emp\_id=temporaryadmission.emp\_id and

new.room\_num=temporaryadmission.room\_num and

new.visit\_date=temporaryadmission.visit\_date and

new.start\_of\_vis between temporaryadmission.start\_of\_adm and temporaryadmission.end\_of\_adm

UNION

select emp\_id

from periodadmission

where new.emp\_id=periodadmission.emp\_id and

new.room\_num=periodadmission.room\_num and

new.start\_of\_vis between periodadmission.start\_of\_adm and periodadmission.end\_of\_adm)

) then exception except\_no\_access;

END

Проверим работу триггера при добавлении новой записи, причем у работника есть допуск в комнату в данный момент:

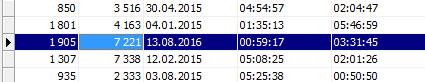


Рис.3 Существующий допуск

execute procedure insert\_factvisit(1905,7221,'13.08.2016','02:47:15','15:22:23');



Рис.4 Запись о фактическом посещении была добавлена

Теперь проверим работу триггера при отсутствии допуска:

execute procedure insert\_factvisit(789,1314,'27.04.2015','08:00:00','21:22:23');



Рис.5 Результат работы триггера

1. Ввести в комнаты тип (обычный, проходная организации). Для гостевых пропусков после выхода из комнаты с типом "проходная" аннулировать пропуск.

Для того, чтобы выполнить данное задание, в таблицу room было добавлено поле type, а в таблицу visitoradmission добавлено поле state(действителен пропуск или нет):

alter table room add room\_type varchar(50) not null;

alter table visitoradmission add state varchar(9) not null;

Комнаты с номерами <3500 сделаны 'usual'-обычными

Комнаты с номерами от 3501 до 7999 были сделаны 'communicating'-проходными

Комнаты с номерами от 8000 были сделаны 'secret'-секретными

update room set room\_type='usual' where room\_num<=3500;

update room set room\_type='communicating' where room\_num between 35001 and 8000;

update room set room\_type='secret' where room\_num>=8000;

Все существующие гостевые пропуски были сделаны 'valid'-действительными

update visitoradmission set state='valid';

Была создана процедура для аннулирования записей в таблице visitoradmission:

drop procedure set\_notvalid;

create procedure set\_notvalid (gname varchar(255), vdate date, rnum int, start\_time date, end\_time DATE)

as

begin

update visitoradmission set state='not valid' where visitoradmission.full\_name=:gname and

visitoradmission.visit\_date=:vdate and

visitoradmission.room\_num=:rnum and

visitoradmission.start\_of\_vis between :start\_time and :end\_time;

end

Также была добавлена таблица для учета фактических гостевых посещений:

create table guest\_fact\_vis (

full\_name varchar(255) not null,

room\_num int not null references room(room\_num),

visit\_date date not null,

start\_of\_vis time not null,

end\_of\_vis time not null);

Для данной таблицы было сгенерировано 100000 записей.

Код написанного триггера:

drop trigger for\_guest\_exit;

CREATE OR ALTER TRIGGER for\_guest\_exit for guest\_fact\_vis

ACTIVE BEFORE INSERT

AS

BEGIN

if (new.full\_name in

(select full\_name from visitoradmission where

new.room\_num=visitoradmission.room\_num and

new.visit\_date=visitoradmission.visit\_date and

new.start\_of\_vis between visitoradmission.start\_of\_vis and visitoradmission.end\_of\_vis

) and

new.room\_num in

(select room\_num from room where

room\_type='communicating')

) then execute procedure set\_notvalid(new.full\_name,new.visit\_date,new.room\_num,new.start\_of\_vis,new.end\_of\_vis);

END

Проверим работу триггера:

При добавлении в фактические гостевые посещения записи, соответствующей выходу из комнаты не типа "проходная"(номер комнаты < 3500)



Рис. 6 Запись в таблице допусков, которая будет проверяться(не должна изменить состояние)

insert into guest\_fact\_vis values('Belkin Dmitriy Glebovich',2733,'2015-11-28', '02:45:00','03:58:19');

Состояние пропуска не изменилось:



Рис.7 Результат работы триггера(состояние пропуска не изменилось)

Теперь проверим работу триггера при выходе гостя из комнаты с типом "проходная"(3500 < номер комнаты < 8000):



Рис.8 Запись в таблице допусков, которая будет проверяться(должна изменить состояние)

insert into guest\_fact\_vis values('Popov Igor Igorevich',5309,'2016-10-10','04:45:00','05:05:19');

Состояние пропуска изменилось:



Рис. 9 Результат работы триггера(состояние пропуска изменилось)

**Вывод**

В данной работы мы познакомились с триггерами. Триггер — это [хранимая процедура](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0) 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](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Триггеры применяются для обеспечения целостности данных и реализации сложной [бизнес-логики](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0" \o "Бизнес-логика). Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в [транзакции](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F), 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).