МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах

кафедра Информационные системы

09.03.02 Информационные системы и технологии (уровень бакалавриата)

Лабораторная работа №2

по дисциплине: «Управление данными»

на тему: «Язык SQL. Генераторы. Триггеры.»

Вариант – 2

Выполнил

студент 3 курса группы ИС/б-33-о

Генералов Николай Николаевич

Отметка о зачете\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Проверил

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность) (подпись) (фамилия, инициалы)

г. Севастополь

2018 г.

# ЦЕЛЬ РАБОТЫ

Ознакомится с принципом работы генераторов и триггеров, продемонстрировать работу на примерах.

# ПОСТАНОВКА ЗАДАЧИ



Рисунок 1 – Схема реализуемой базы данных

1. Разработать и выполнить с помощью SQL-редактора запросы, соединяющие две таблицы с помощью JOIN и без него.
2. Разработать и выполнить с помощью SQL-редактора запросы, соединяющие более чем две таблицы с помощью JOIN и без него.
3. Продемонстрировать с помощью SQL-редактора следующие возможности SQL:

* использование псевдонимов на примере рекурсивного запроса;
* привести пример запроса с подзапросом;
* использование агрегатных функций в подзапросе;
* подзапросы, использующие вычисление;
* использование подзапросов в HAVING;

1. Ознакомиться с принципов и продемонстрировать работу коррелированных подзапросов:

* привести пример соединения таблицы со своей копией;
* привести пример коррелированного запроса, использующего две разные таблицы;
* привести пример запроса с оператором EXIST;
* привести пример запроса с оператором ALL;
* привести пример запроса с оператором ANY;

1. В соответствии с вариантом задания создать генераторы для полей «Номер преподавателя», «Номер курса», «Номер студента». Выполнить автоматическую генерацию полей «Номер преподавателя», «Номер курса», «Номер студента».
2. Изменить значение генератора, в соответствии с хранимыми данными.
3. Внести изменения в указанные таблицы, используя триггеры (не менее 5 строк). Просмотреть полученный результат.

# ХОД РАБОТЫ

1. Выполним запрос, соединяющий две таблицы с помощью JOIN.

SELECT t1.date\_of\_lesson, t2.course\_name FROM lesson t1 JOIN discipline\_course t2 ON t1.course\_num = t2.course\_num;

Данный запрос можно проинтерпретировать как: вывести дату проведения и название дисциплины, взяв данные из двух таблиц.

Выполним аналогичный запрос без использования JOIN:

SELECT (t1.date\_of\_lesson), t2.course\_name FROM lesson t1, discipline\_course t2 WHERE t1.course\_num = t2.course\_num;

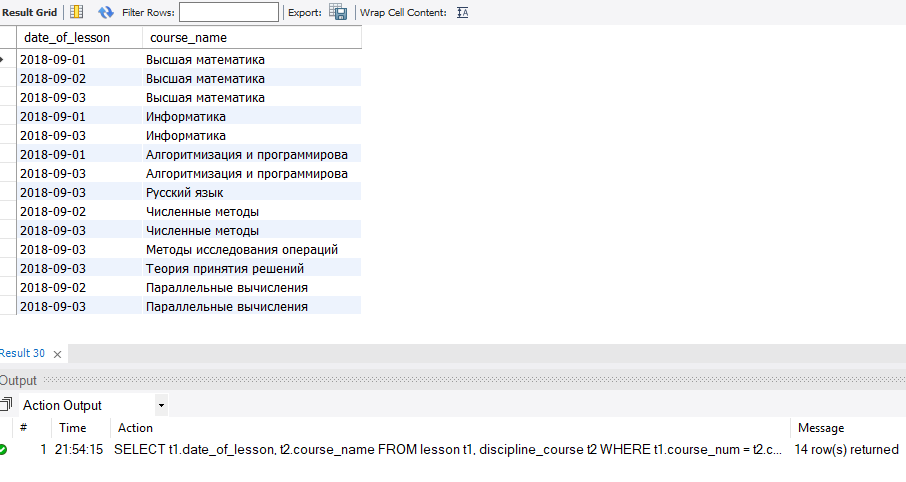


Рисунок 1 – Результат выполнения запросов п.3.1

1. Выполним запрос, соединяющий более чем две таблицы с помощью JOIN.

Запрос будет выводить результат аналогичный предыдущим, но дополненный данными из третей таблицы содержащий фамилии преподавателей, ведущих дисциплины.

SELECT t1.date\_of\_lesson, t2.course\_name, t3.fio FROM lesson t1, discipline\_course t2, teacher t3 WHERE t1.course\_num = t2.course\_num AND t2.teacher\_num = t3.id;

Выполним аналогичный запрос без использования JOIN:

SELECT t1.date\_of\_lesson, t2.course\_name, t3.fio FROM lesson t1 JOIN discipline\_course t2 ON t1.course\_num = t2.course\_num JOIN teacher t3 ON t2.teacher\_num = t3.id;

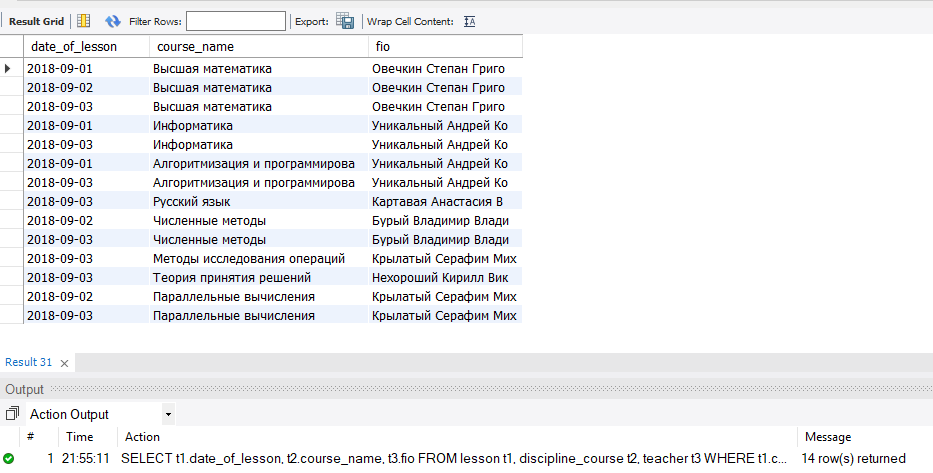


Рисунок 2 – Результат выполнения запросов п.3.2

1. Продемонстрируем использование псевдонимов на примере рекурсивного запроса:

SELECT distinct t1.course\_num, t2.pre\_course\_num FROM discipline\_course t1, discipline\_course t2 WHERE t1.pre\_course\_num = t2.pre\_course\_num AND t1.course\_num <> t2.course\_num;

Данный запрос выводит только номер дисциплины и номер дисциплины предшествующей ей.

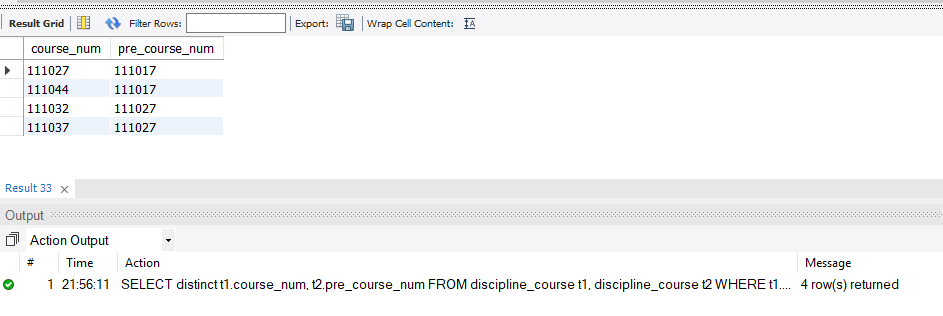


Рисунок 3 – Результат выполнения запроса п.3.3

1. Выполним запрос с использованием агрегатной функций в подзапросе:

SELECT t1.course\_num FROM lesson t1 WHERE t1.date\_of\_lesson = (SELECT MIN(t2.date\_of\_lesson) FROM lesson t2);

Данный запрос выводит номера дисциплин, которые проводились самыми первыми.

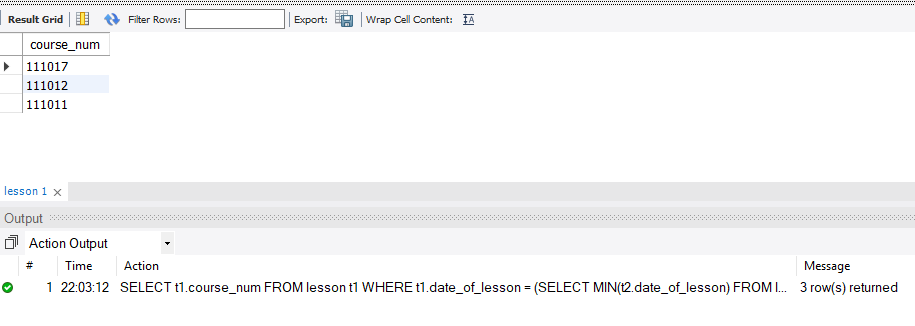


Рисунок 4 – Результат выполнения запроса п.3.4

1. Выполним запрос с использованием вычислений в подзапросе:

SELECT t1.course\_num FROM lesson t1 WHERE t1.date\_of\_lesson = (SELECT (MAX(date\_of\_lesson) - 1) FROM lesson);

Данный запрос выводит номера дисциплин, которые проводились в предпоследний день работы кафедры.

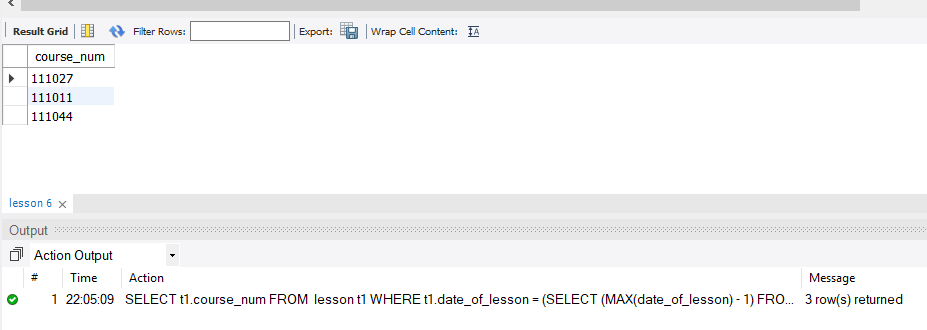


Рисунок 5 – Результат выполнения запроса п.3.5

1. Выполнение запроса с подзапросом, возвращающим множественные значения:

SELECT t1.course\_name FROM discipline\_course t1 WHERE t1.course\_num IN (SELECT t2.course\_num FROM lesson t2) AND t1.teacher\_num = 2;

Данный запрос выводит названия дисциплин, которые ведет преподаватель с номером 2, которые имеются в расписании.

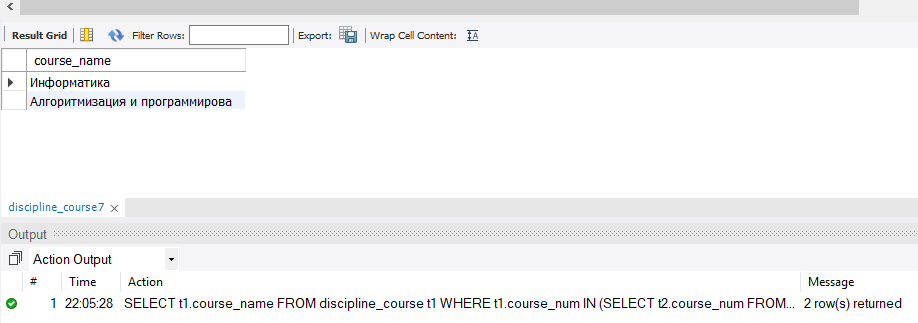


Рисунок 6 – Результат выполнения запроса п.3.6

1. Выполнение запроса с подзапросом, возвращающим единственное значение:

SELECT t1.teacher\_num FROM discipline\_course t1 WHERE t1.course\_num = (SELECT t2.course\_num FROM lesson t2 WHERE t2.date\_of\_lesson = '2018-09-02' AND t2.class\_room\_num = 419); использование подзапросов в HAVING;

Данный запрос выводит номер преподавателя, который ведет занятие 2 сентября в аудитории ­– 419.

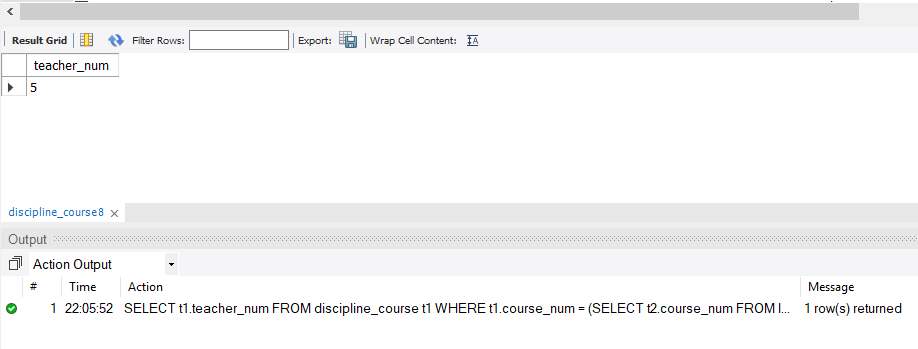


Рисунок 7 – Результат выполнения запроса п.3.7

1. Выполнение подзапроса в директиве HAVING:

SELECT COUNT(\*), t1.course\_num FROM lesson t1 GROUP BY t1.course\_num HAVING t1.course\_num IN (SELECT t2.course\_num FROM discipline\_course t2 WHERE teacher\_num = 2);

Данный запрос выводит номера дисциплин и количество занятий по каждой из них, которые ведет преподаватель с номером 2.

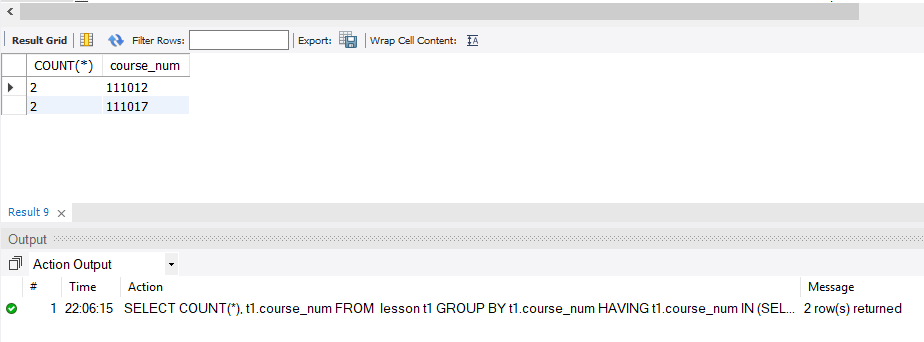


Рисунок 8 – Результат выполнения запроса п.3.8

1. Выполним запрос для соединения таблицы со своей копией:

SELECT t1.course\_name, copy.course\_name, t1.course\_num FROM discipline\_course t1, discipline\_course copy WHERE t1.teacher\_num = copy.teacher\_num AND t1.course\_num != copy.course\_num;

Данный запрос выводит пары номеров дисциплин, которые читаются в одном курсе и ведутся одним преподавателем.

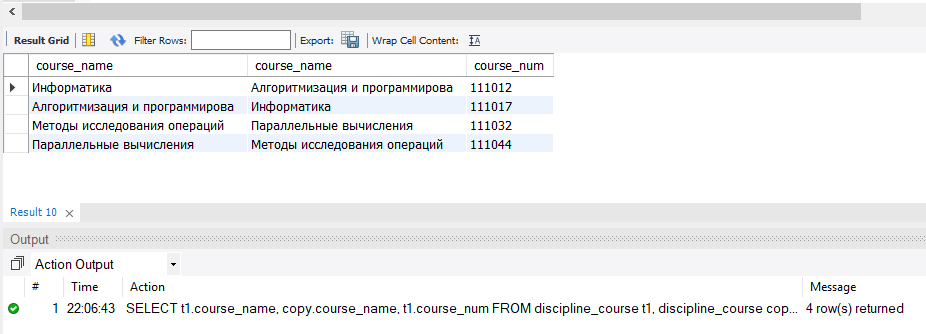
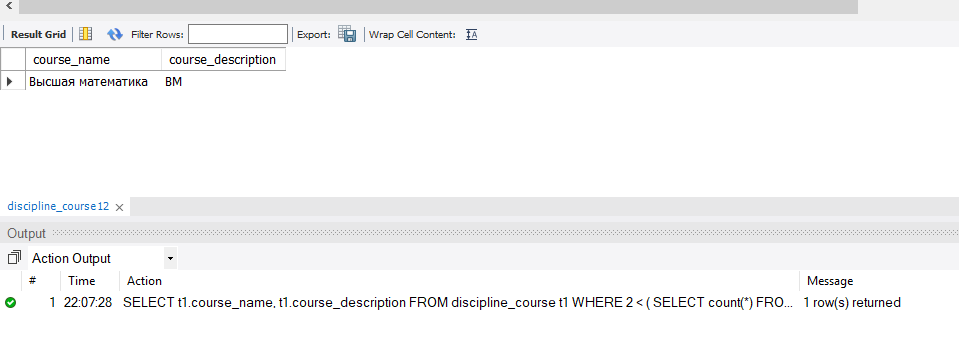


Рисунок 9 – Результат выполнения запроса п.3.9

1. Выполним коррелированный запрос, использующий две разные таблицы;

SELECT t1.course\_name, t1.course\_description FROM discipline\_course t1 WHERE 2 < (SELECT count(\*) FROM lesson t2 WHERE t1.course\_num = t2.course\_num);

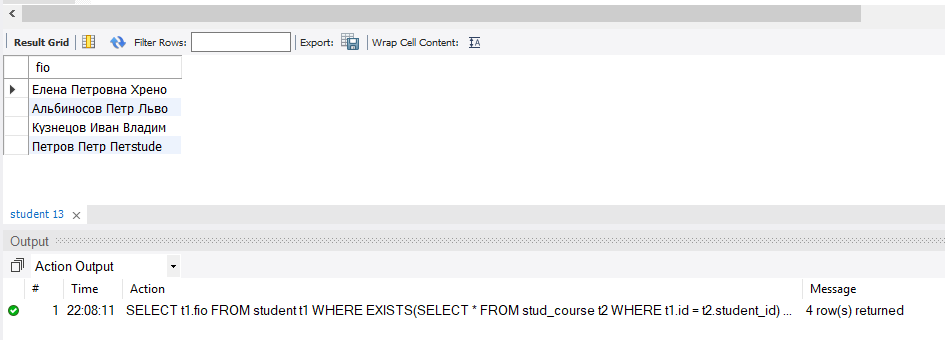
Данный запрос выводит название и описание дисциплин, количество занятий по которым больше 2.

Рисунок 10 – Результат выполнения запроса п.3.10

1. Выполним запрос с использованием оператора EXIST:

SELECT t1.fio FROM student t1 WHERE EXISTS(SELECT \* FROM stud\_course t2 WHERE t1.id = t2.student\_sid);

Данный запрос выводит фамилии студентов, которые выбрали курсы для посещения.

Рисунок 11 – Результат выполнения запроса п.3.11

1. Выполним запрос с использованием оператора ALL;

SELECT DISTINCT t1.course\_num FROM lesson t1 WHERE t1.date\_of\_lesson <= ALL (SELECT t2.date\_of\_lesson FROM lesson t2 WHERE t2.class\_room\_num = 419);

Данный запрос выводит номера курсов, которые проводились в день дисциплин проводимых в аудитории – 419 или раньше.

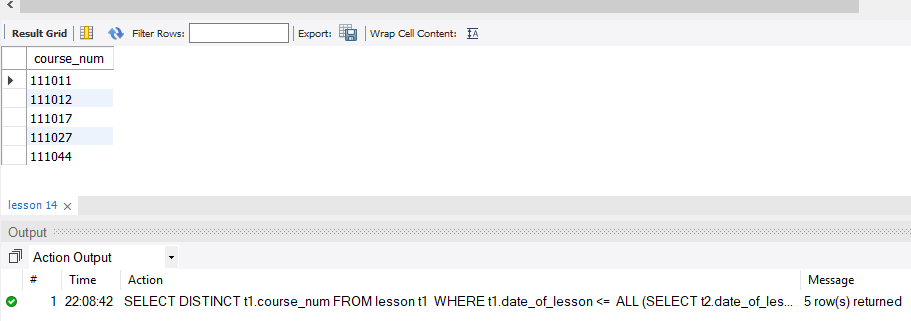


Рисунок 12 – Результат выполнения запроса п.3.12

1. Выполним запрос с использованием оператора ANY;

SELECT t1.course\_name FROM discipline\_course t1 WHERE t1.course\_num = ANY(SELECT t2.course\_num FROM lesson t2 WHERE date\_of\_lesson < '2018-09-03');

Данный запрос выводит название дисциплин, занятия по которым проводились до 3 сентября.

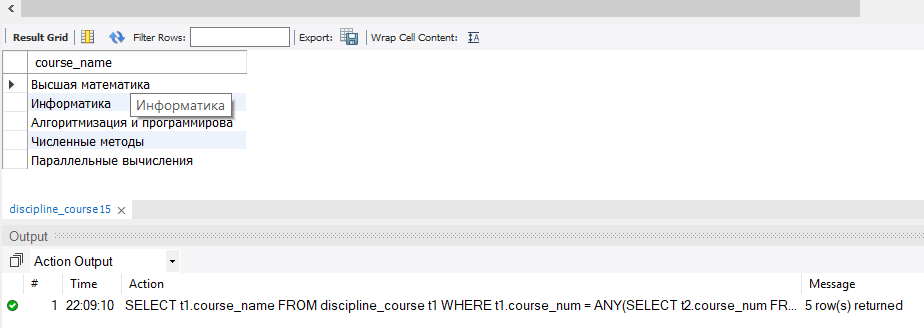


Рисунок 13 – Результат выполнения запроса п.3.13

1. Создадим генератор для поля «Номер студента» и напишем триггер для автоматической генерации полей при добавлении данных о студенте в таблицу student:

CREATE GENERATOR NUM\_GENERATOR;

SET GENERATOR NUM\_GENERATOR to 10000;

DELIMITER ::

CREATE TRIGGER generate\_student\_num AFTER INSERT ON student

FOR EACH ROW

BEGIN

DECLARE student\_id INT DEFAULT GEN\_ID(NUM\_GENERATOR,1);

UPDATE student SET id = student\_id WHERE id = new.ID;

END;

::

DELIMITER ;

В результате добавления новые записи в таблице будут иметь id начинающиеся с 10001.

1. Создадим триггер запрещающий посещать студенту более 5 курсов.

DELIMITER ::

CREATE TRIGGER lock\_insert\_when\_student\_try\_register\_more\_5\_courses BEFORE INSERT ON `stud\_course`

FOR EACH ROW

BEGIN

DECLARE count\_of\_student\_courses INT DEFAULT 0;

SELECT COUNT(\*) INTO count\_of\_student\_courses FROM stud\_course WHERE student\_id = new.student\_id;

IF count\_of\_student\_courses >= 5

THEN

SIGNAL SQLSTATE '2201R'

SET MESSAGE\_TEXT = "Нельзя зарегистрировать больше 5 курсов";

END IF;

END;

::

DELIMITER ;

Убедимся в работоспособности триггера выполним добавление записей в таблицу stud\_course:

INSERT INTO stud\_course VALUE ('4', '111011');

INSERT INTO stud\_course VALUE ('4', '111012');

INSERT INTO stud\_course VALUE ('4', '111017');

INSERT INTO stud\_course VALUE ('4', '111044');

INSERT INTO stud\_course VALUE ('4', '111027');

INSERT INTO stud\_course VALUE ('4', '111032');



Рисунок 14 – Результат срабатывания триггера lock\_insert\_when\_student\_try\_register\_more\_5\_courses

ВЫВОД

В ходе выполнения лабораторной работы на практике были исследованы операции для создания сложных, коррелированных запросов, запросов с подзапросами, а так же запросов с использованием операторов EXISTS, ANY, ALL. Так же были получены практические навыки создания и использования генераторов и триггеров, которые в связке позволяют задать автоматически генерируемые поля при добавлении данных в таблицы. Генераторы могут быть использованы для отправки сообщений пользователю и выполнения заданных действий при наступлении определенных событий.