МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационные системы

Сирота Марина Романовна

Институт информационных технологий и управления в технических системах

курс 3 группа ИС/б-32-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №2

по дисциплине: «Управление данными»

по теме: «Язык SQL. Генераторы. Триггеры»

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

(дата)

Руководитель практикума

ст. пр. Гончаренко Д.Г.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2018

1. ЦЕЛЬ РАБОТЫ

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

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Записать запросы, соединяющие две таблицы с помощью JOIN и без него (вариант задания соответствует варианту, полученному в Лабораторной работе №1).

2. Записать запросы, соединяющие более двух таблиц с помощью JOIN и без него.

3. Продемонстрировать 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.

4. Записать запрос, соединяющий таблицу со своей копией.

5. Продемонстрировать следующие возможности SQL

- работу оператора EXISTS;

- работу оператора ALL;

- работу оператора ANY.

6. В соответствии с вариантом задания создать генератор и триггер (см. приложение А).

7. Изменить значение генератора, в соответствии с хранимыми данными.

8. Ввести данные в таблицу, используя генератор (не менее 5 строк). Просмотреть

полученный результат.

9. Внести изменения в указанные таблицы, используя триггеры (не менее 5 строк). Просмотреть полученный результат.

10. Написать отчет.

1. ВЫПОЛНЕНИЕ РАБОТЫ
2. Записать запросы, соединяющие две таблицы с помощью JOIN и без него (вариант задания соответствует варианту, полученному в Лабораторной работе №1).

SELECT `Номер паспорта` , `заказ`.`Срок проживания`, `гостиница`.`Название` from `гостиница`,`заказ` where (`заказ`.`номер гостиницы`=`гостиница`.`номер гостиницы`)

SELECT `Номер паспорта` , `Срок проживания` from `заказ` join `гостиница` on `заказ`.`номер гостиницы`=`гостиница`.`номер гостиницы`

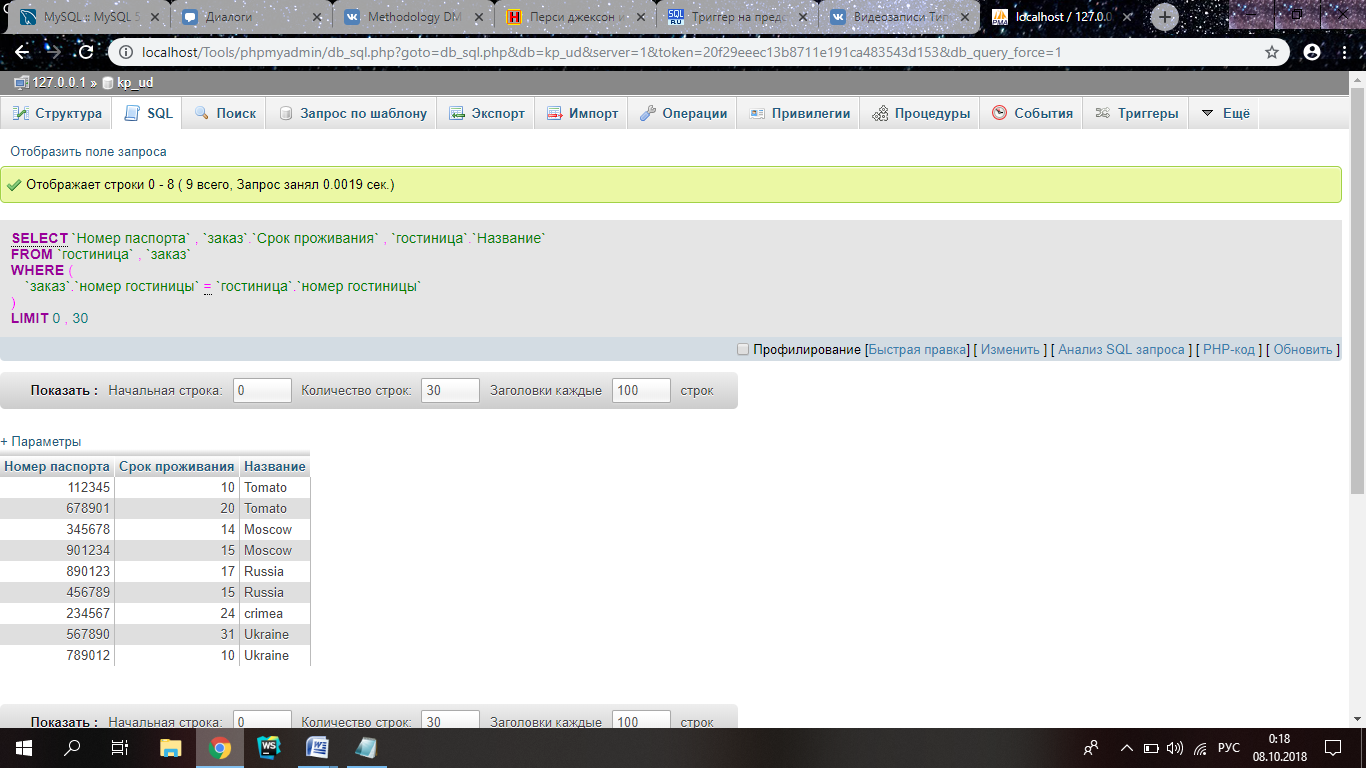
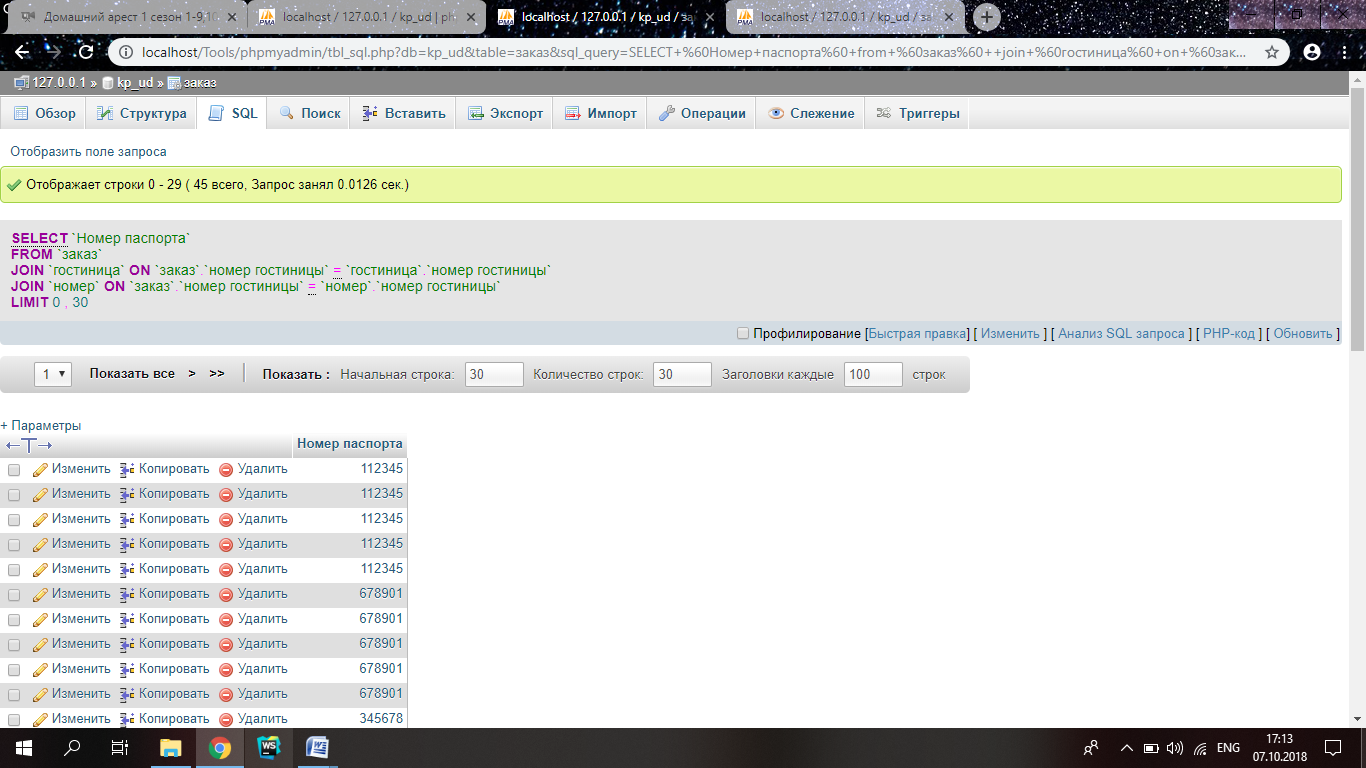


Рисунок 3.1 ‒ Запрос, соединяющий две таблицы

1. Записать запросы, соединяющие более двух таблиц с помощью JOIN и без него.

SELECT `Номер паспорта` from `заказ` join `гостиница` on `заказ`.`номер гостиницы`=`гостиница`.`номер гостиницы` join `номер` on `заказ`.`номер гостиницы`=`номер`.`номер гостиницы`

SELECT `Номер паспорта` from `гостиница`,`заказ`, `номер` where (`заказ`.`номер гостиницы`=`гостиница`.`номер гостиницы`) and (`заказ`.`номер гостиницы`=`номер`.`номер гостиницы`)



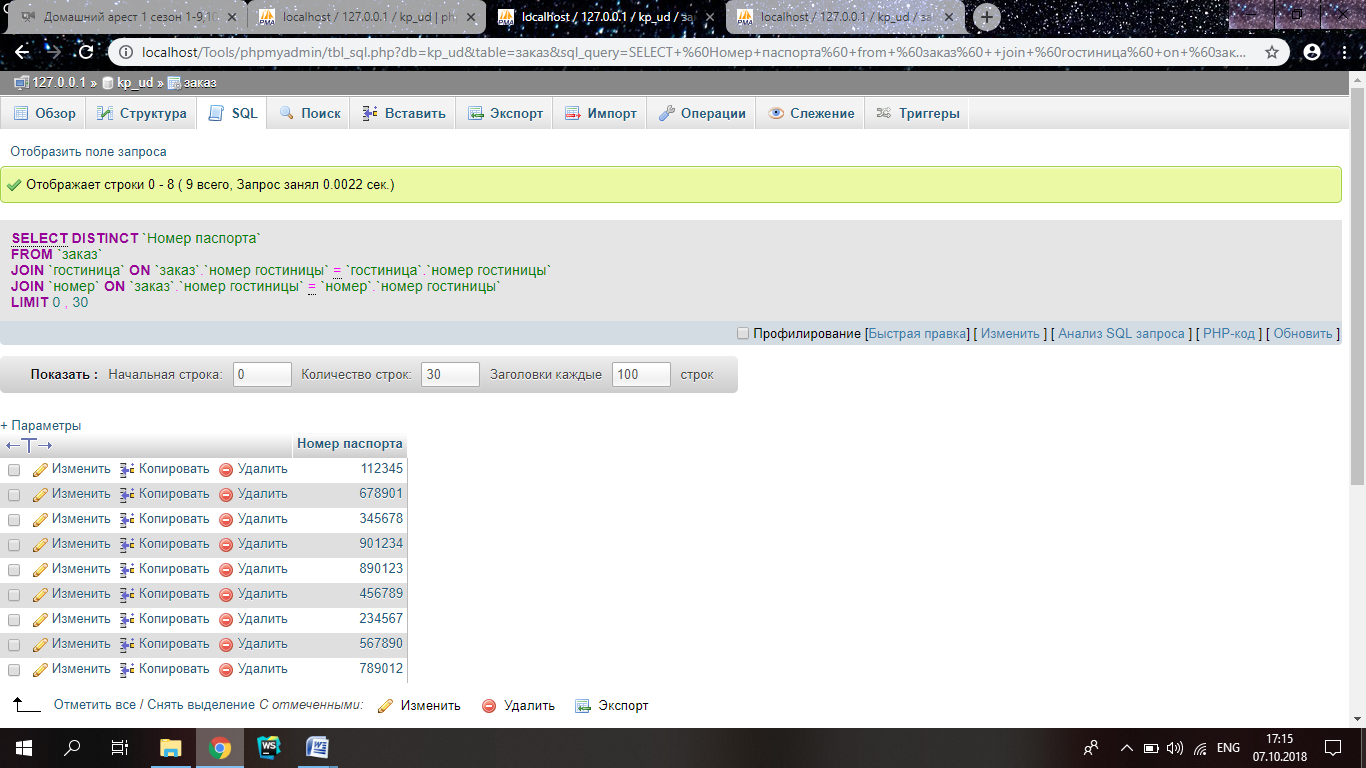


Рисунок 3.2 ‒ Запрос, соединяющий более двух таблиц

1. Продемонстрировать следующие возможности SQL:

* использование псевдонимов на примере рекурсивного запроса;

SELECT

n1.`Номер паспорта`,

n1.`Срок проживания`,

n2.`Номер паспорта`,

n2.`Срок проживания`

FROM `заказ` n1

JOIN `заказ` n2 ON n1.`Срок проживания` =n2.`Срок проживания`-1

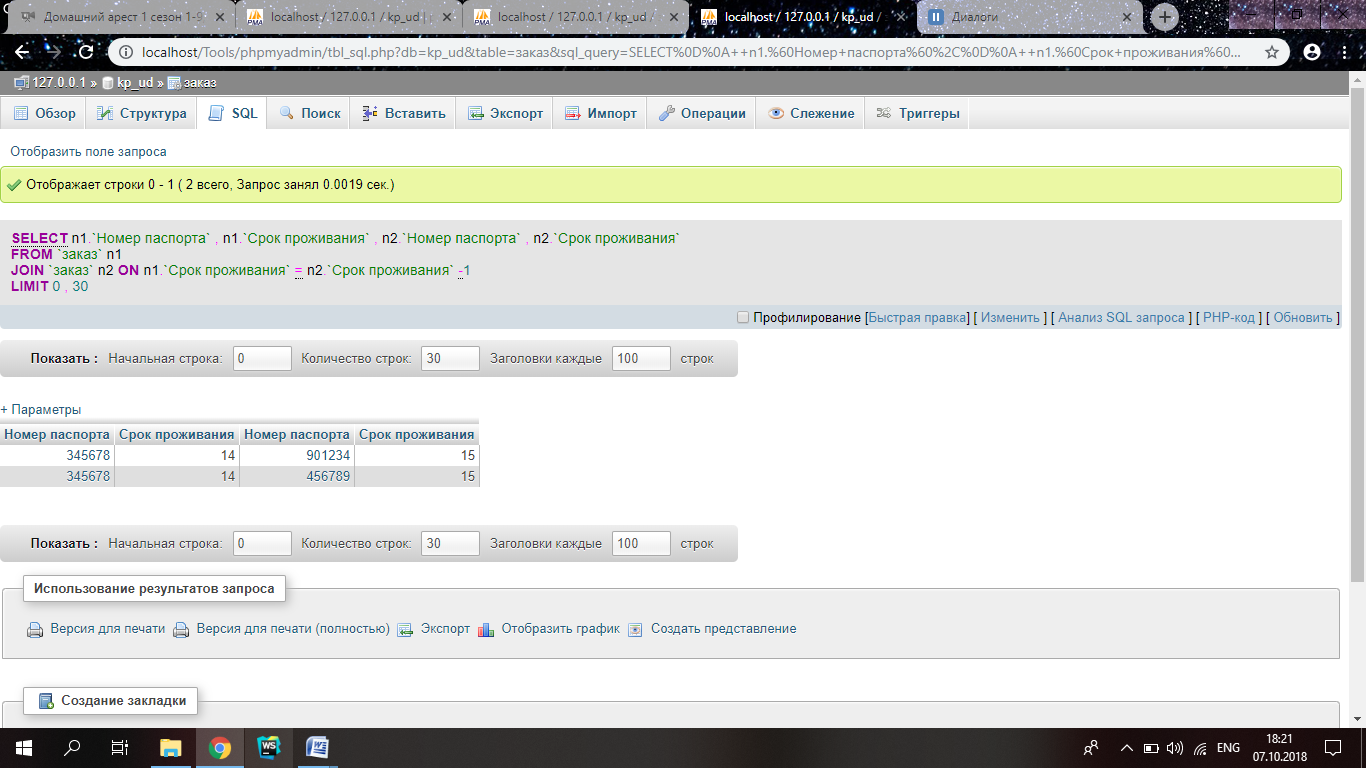


Рисунок 3.3 ‒ Использование псевдонимов на примере рекурсивного запроса

* использование агрегатных функций в подзапросе;

SELECT `Номер паспорта` FROM `заказ` where `Срок проживания`< (SELECT avg(`Срок проживания`) from `заказ`)

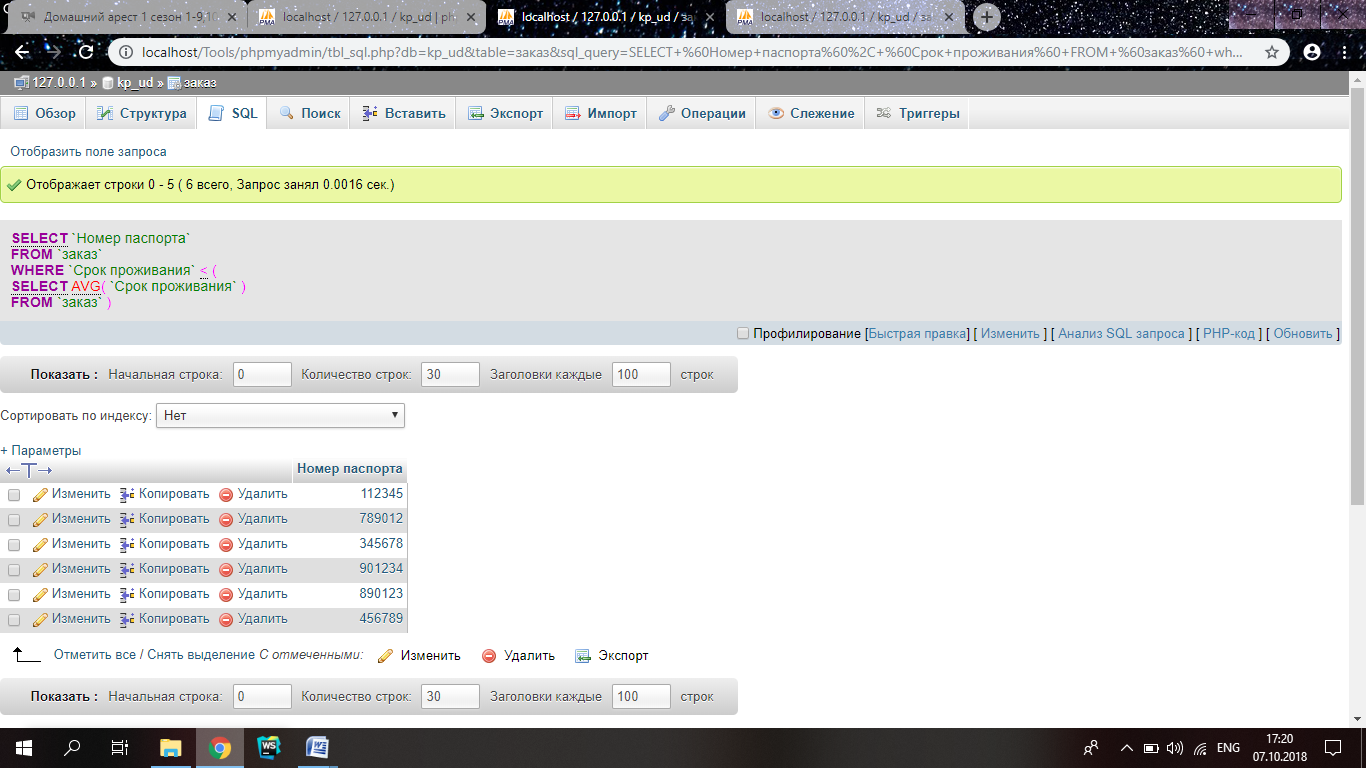


Рисунок 3.4 ‒ Использование агрегатных функций в подзапросе

* подзапросы, использующие вычисление;

SELECT \* FROM `заказ` where `Номер паспорта` in (SELECT `Номер паспорта` from `заказ` where`Дата вселения`>'2018-06-11')

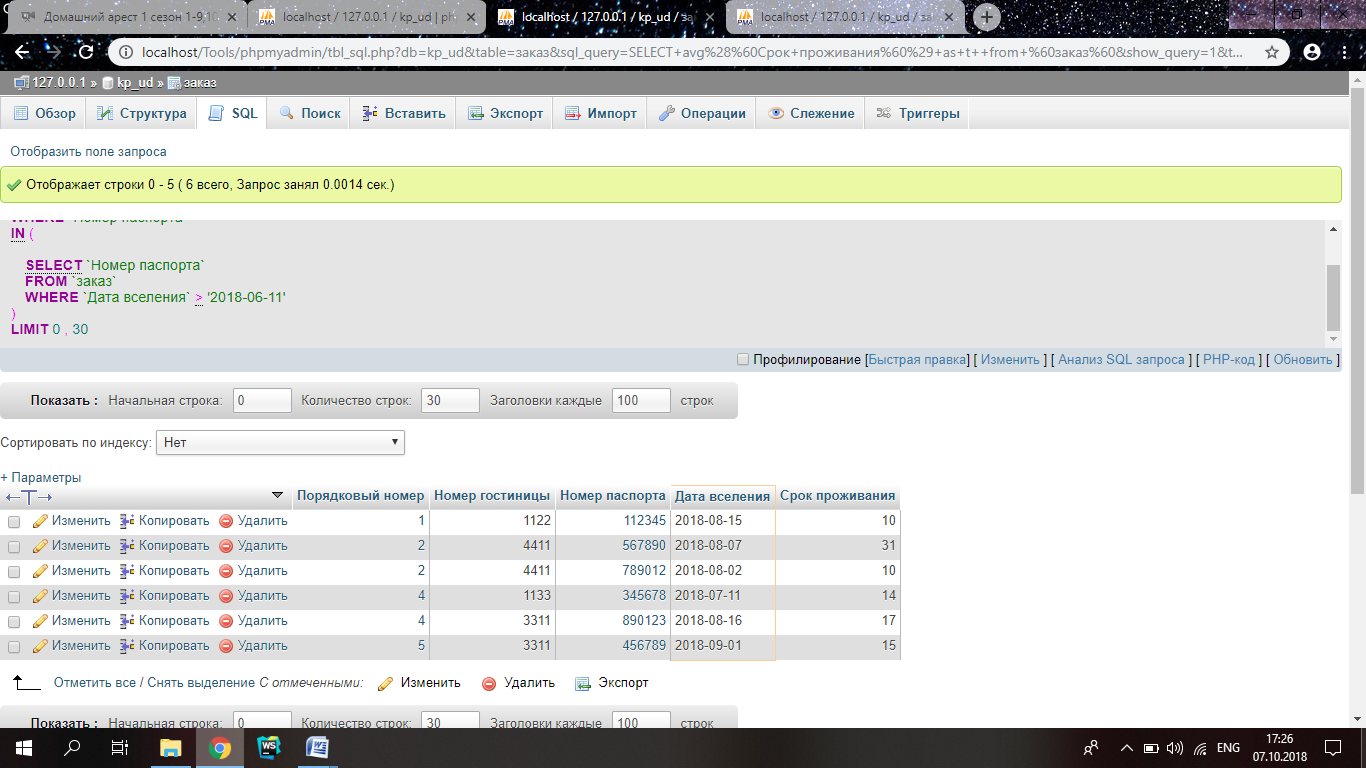


Рисунок 3.5 ‒ Подзапрос, использующий вычисление

* использование подзапросов в HAVING

SELECT \* FROM `заказ` GROUP BY `Дата вселения` HAVING (`Срок проживания`) < ( SELECT avg(`Срок проживания`) from `заказ`)

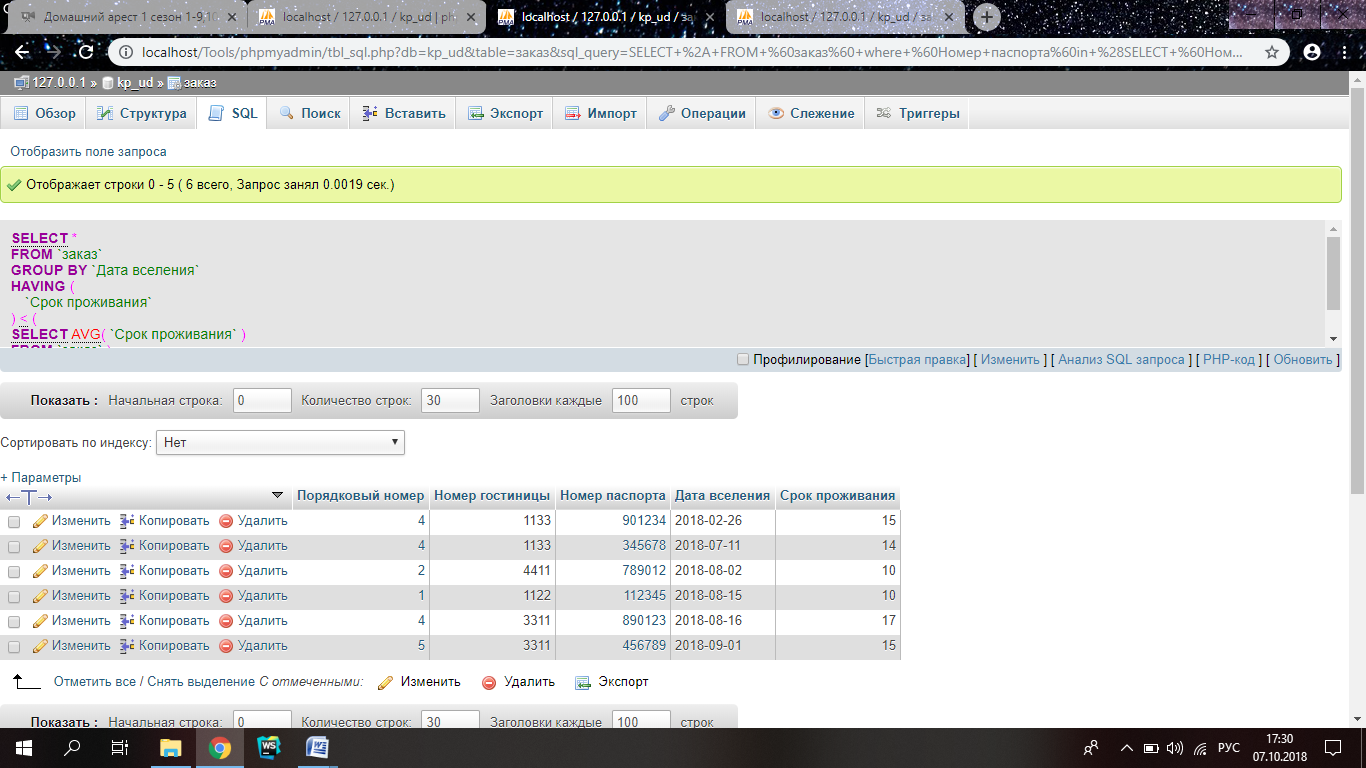


Рисунок 3.6 ‒ Подзапрос, использующийся в HAVING

1. Записать запрос, соединяющий таблицу со своей копией.

SELECT \* FROM `заказ` n1

JOIN `заказ` n2 ON n1.`Срок проживания` =n2.`Срок проживания`

1. Продемонстрировать следующие возможности SQL

* работу оператора EXISTS;

SELECT `Номер паспорта`, `Срок проживания`FROM `заказ`

WHERE not exists

(SELECT `Номер паспорта` FROM `заказ` WHERE `Срок проживания` between 15 and 25 )

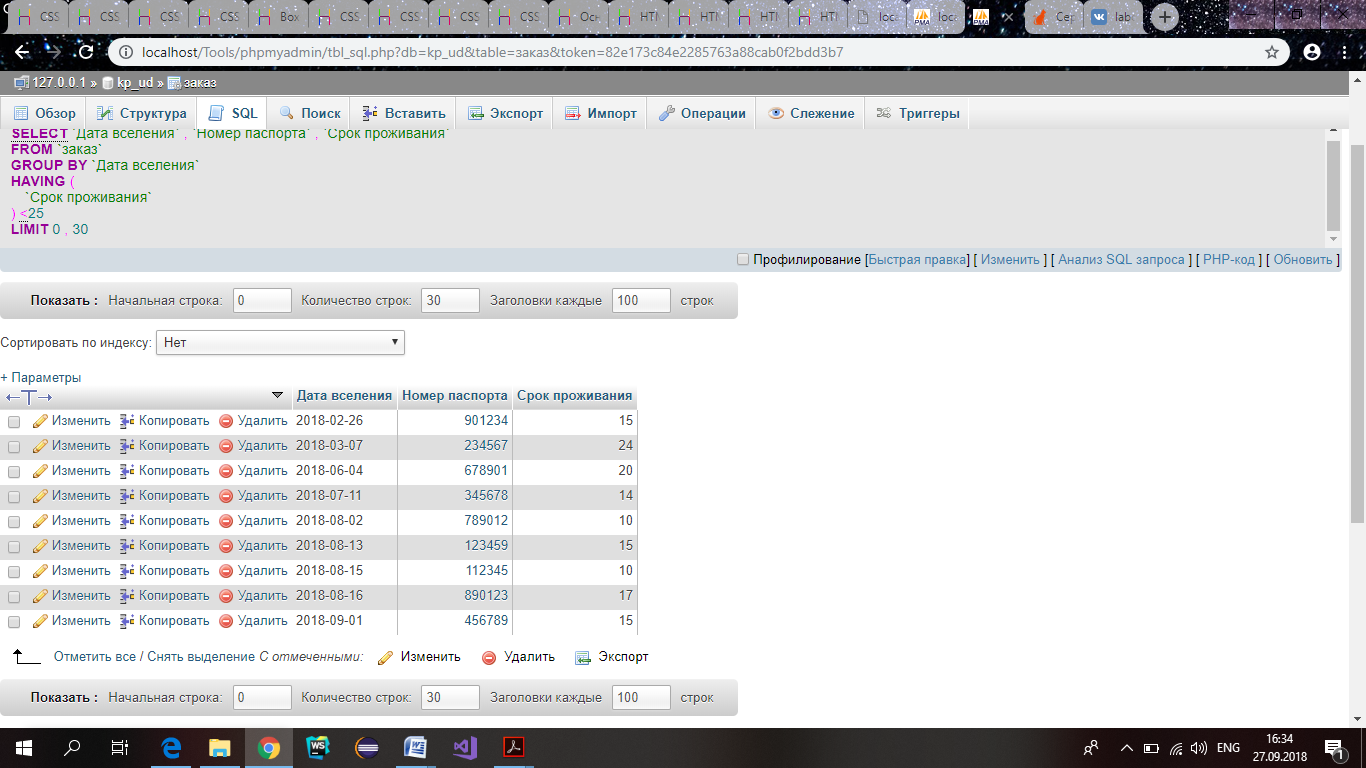


Рисунок 3.7 ‒ Подзапрос, использующий оператор EXISTS

* работу оператора ALL;

SELECT `Номер паспорта`, `Срок проживания`,`Номер гостиницы` FROM `заказ`

WHERE `Номер гостиницы` = ALL (SELECT `Номер гостиницы` FROM `гостиница` )

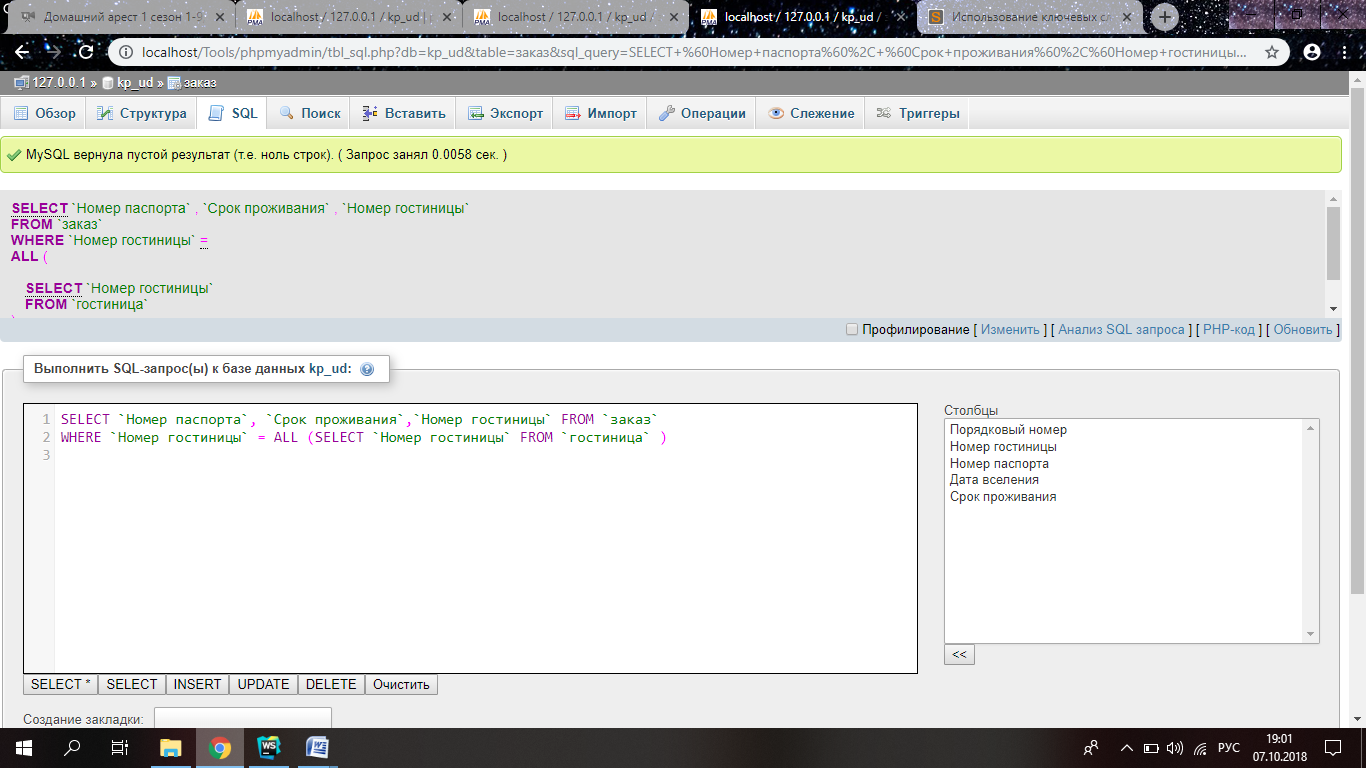


Рисунок 3.7 ‒ Подзапрос, использующий оператор ALL

* работу оператора ANY.

SELECT `Номер паспорта`, `Срок проживания`,`Номер гостиницы` FROM `заказ`

WHERE `Номер гостиницы` = ANY (SELECT `Номер гостиницы` FROM `гостиница`)

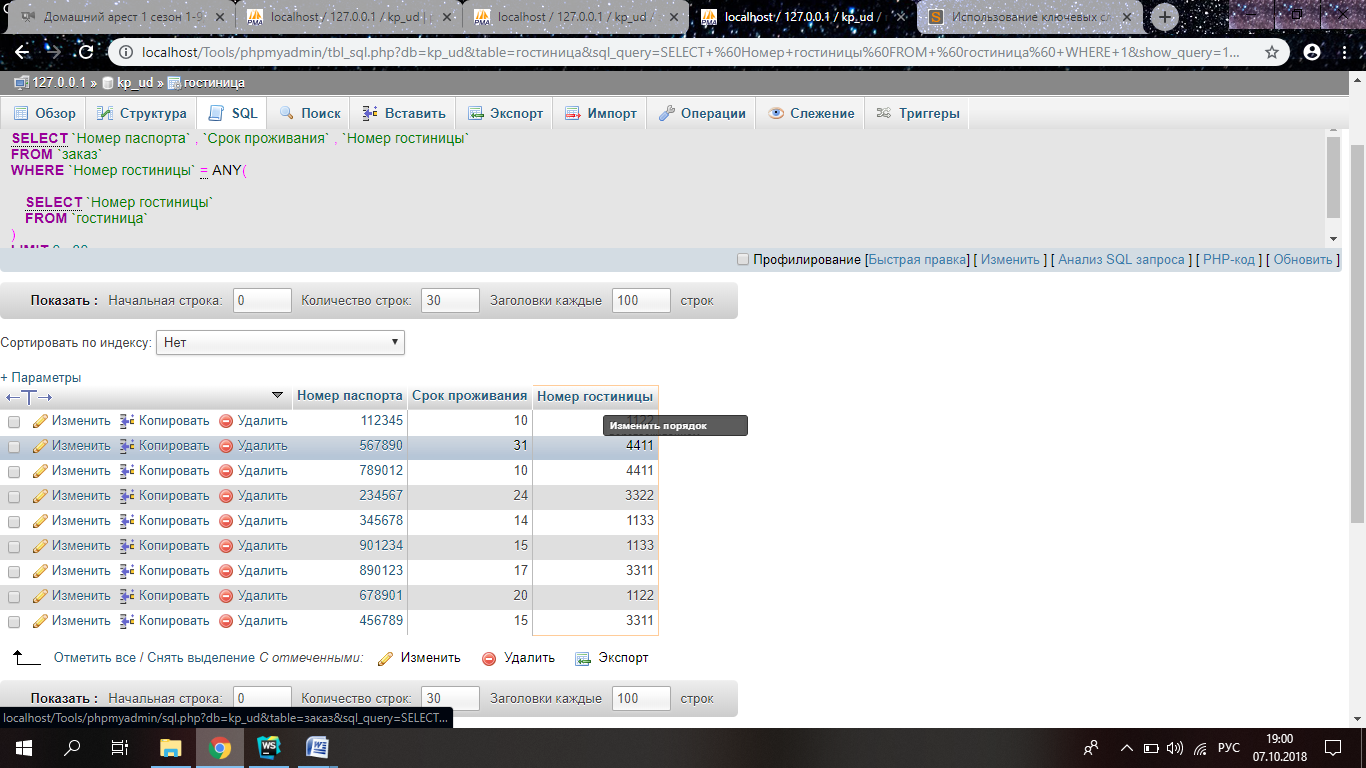


Рисунок 3.7 ‒ Подзапрос, использующий оператор ANY

1. Создать генераторы для полей «Номер гостиницы», «Номер фирмы». Автоматическая генерация полей «Номер гостиницы», «Номер фирмы». Создать обновляемое представление «Номера гостиницы», состоящее из полей «Название гостиницы», «Адрес», «Разряд», «Порядковый номер номера», «Количество комнат», «Разряд номера».

CREATE generator Номер\_фирмы\_GEN;

CREATE generator Номер\_гостиницы\_GEN;

CREATE VIEW номера\_гостиница AS

SELECT `Название`, `Адрес`, `Разряд`,`Порядковый номер`,`Количество комнат`,`Разряд номера`

FROM гостиница, номер where `гостиница`.`номер гостиницы`=`номер`.`номер гостиницы`

CREATE TRIGGER номера\_гостиница\_Delete FOR номера\_гостиница BEFORE DELETE AS

BEGIN

DELETE FROM гостиница

WHERE `номер гостиницы` = OLD.`номер гостиницы`;

DELETE FROM номер

WHERE `номер гостиницы` = OLD.`номер гостиницы`;

END;

CREATE TRIGGER номера\_гостиница\_Update FOR номера\_гостиница

BEFORE UPDATE AS

BEGIN

UPDATE гостиница

SET `Название`= NEW.`Название`

SET `Адрес`= NEW.`Адрес`

SET `Разряд`= NEW.`Разряд`

WHERE `номер гостиницы` = OLD.`номер гостиницы`;

UPDATE номер

SET `Порядковый номер`= NEW.`Порядковый номер`

SET `Количество комнат`= NEW.`Количество комнат`

SET `Разряд номера`= NEW.`Разряд номера`

WHERE `номер гостиницы` = OLD.`номер гостиницы`;

END;

CREATE TRIGGER номера\_гостиница\_Insert FOR номера\_гостиница BEFORE INSERT AS

BEGIN

INSERT INTO гостиница values (NEW.`номер гостиницы`,NEW.`Название`);

INSERT INTO гостиница values (NEW.`номер гостиницы`,NEW.`Адрес`);

INSERT INTO гостиница values (NEW.`номер гостиницы`,NEW.`Разряд`);

INSERT INTO номер values (NEW.`номер гостиницы`,NEW.`Порядковый номер`);

INSERT INTO номер values (NEW.`номер гостиницы`,NEW.`Количество комнат`);

INSERT INTO номер values (NEW.`номер гостиницы`,NEW.`Разряд номера`);

END;

CREATE VIEW history AS

SELECT \*

FROM заказ

CREATE TRIGGER заказ\_INSERT

AFTER INSERT

on заказ for each row

INSERT INTO History (`Порядковый номер`, `Номер гостиницы`,`Номер паспорта`,`Дата вселения`,`Срок проживания`)

SELECT `Дата вселения`, 'клиент'+' с номер паспорта ' + `Номер паспорта` + ' заселился в гостиницу с номером ' + `Номер гостиницы` +'на'+`Срок проживания`+'дней'

FROM INSERTED

CREATE TRIGGER заказ\_update

AFTER UPDATE

on заказ for each row

INSERT INTO History (new.`Порядковый номер`, new.`Номер гостиницы`,new.`Номер паспорта`,new.`Дата вселения`,new.`Срок проживания`)

SELECT 'Изменение информации у записи со следующими данными: '+ old.`Дата вселения`+ old.`Номер паспорта`+ old.`Номер гостиницы` +old.`Срок проживания`

FROM INSERTED

CREATE TRIGGER `заказ\_update`

AFTER UPDATE ON `заказ`

FOR EACH ROW

INSERT INTO History (`Номер гостиницыH`,`Номер паспортаH`,`Дата вселенияH`,`Срок проживанияH`)

SELECT 'Изменение информации у клиента со следующими данными: '+`Номер паспорта`

FROM INSERTED

CREATE TRIGGER номера\_гостиница\_Delete BEFORE DELETE ON номера\_гостиница FOR EACH ROW

BEGIN

DELETE FROM гостиница

WHERE `номер гостиницы` = OLD.`номер гостиницы`;

DELETE FROM номер

WHERE `номер гостиницы` = OLD.`номер гостиницы`;

END;

CREATE TRIGGER номера\_гостиница\_Update BEFORE UPDATE ON номера\_гостиница FOR EACH ROW

BEGIN

UPDATE гостиница

SET `Название`= NEW.`Название`

SET `Адрес`= NEW.`Адрес`

SET `Разряд`= NEW.`Разряд`

WHERE `номер гостиницы` = OLD.`номер гостиницы`;

UPDATE номер

SET `Порядковый номер`= NEW.`Порядковый номер`

SET `Количество комнат`= NEW.`Количество комнат`

SET `Разряд номера`= NEW.`Разряд номера`

WHERE `номер гостиницы` = OLD.`номер гостиницы`;

END;

CREATE TRIGGER номера\_гостиница\_Insert BEFORE INSERT ON номера\_гостиница FOR EACH ROW

BEGIN

INSERT INTO гостиница values (NEW.`номер гостиницы`,NEW.`Название`);

INSERT INTO гостиница values (NEW.`номер гостиницы`,NEW.`Адрес`);

INSERT INTO гостиница values (NEW.`номер гостиницы`,NEW.`Разряд`);

INSERT INTO номер values (NEW.`номер гостиницы`,NEW.`Порядковый номер`);

INSERT INTO номер values (NEW.`номер гостиницы`,NEW.`Количество комнат`);

INSERT INTO номер values (NEW.`номер гостиницы`,NEW.`Разряд номера`);

END;

ВЫВОДЫ

В ходе лабораторной работы были изучены основные принципом работы генераторов и триггеров.

Была продемонстрирована их работа на примерах.

Была продемонстрирована работа запросов с подзапросами, запросы с копиями и псевдонимами, запросы с EXISTS, ANY, ALL. Также была продемонстрирована их работа JOIN для объединения двух и более таблиц.