Курс: Основы реляционных баз данных. MySQL
Урок 9. Транзакции, переменные, представления. Администрирование. Хранимые процедуры и функции, триггеры
Выполнил: Кузнецов Сергей (Факультет Geek University Python-разработки)

Домашнее задание:

“Транзакции, переменные, представления”

1. В базе данных shop и sample присутствуют одни и те же таблицы, учебной базы данных. Переместите запись id = 1 из таблицы shop.users в таблицу sample.users. Используйте транзакции.
2. Создайте представление, которое выводит название name товарной позиции из таблицы products и соответствующее название каталога name из таблицы catalogs.
3. (по желанию) Пусть имеется таблица с календарным полем created_at. В ней размещены разряженые календарные записи за август 2018 года '2018-08-01', '2016-08-04', '2018-08-16' и 2018-08-17. Составьте запрос, который выводит полный список дат за август, выставляя в соседнем поле значение 1, если дата присутствует в исходном таблице и 0, если она отсутствует.
4. (по желанию) Пусть имеется любая таблица с календарным полем created_at. Создайте запрос, который удаляет устаревшие записи из таблицы, оставляя только 5 самых свежих записей.

“Администрирование MySQL” (эта тема изучается по вашему желанию)

5. Создайте двух пользователей которые имеют доступ к базе данных shop. Первому пользователю shop_read должны быть доступны только запросы на чтение данных, второму пользователю shop — любые операции в пределах базы данных shop.
6. (по желанию) Пусть имеется таблица accounts содержащая три столбца id, name, password, содержащие первичный ключ, имя пользователя и его пароль. Создайте представление username таблицы accounts, предоставляющий доступ к столбца id и name. Создайте пользователя user_read, который бы не имел доступа к таблице accounts, однако, мог бы извлекать записи из представления username.

“Хранимые процедуры и функции, триггеры"

7. Создайте хранимую функцию hello(), которая будет возвращать приветствие, в зависимости от текущего времени суток. С 6:00 до 12:00 функция должна возвращать фразу "Доброе утро", с 12:00 до 18:00 функция должна возвращать фразу "Добрый день", с 18:00 до 00:00 — "Добрый вечер", с 00:00 до 6:00 — "Доброй ночи".

8. В таблице products есть два текстовых поля: name с названием товара и description с его описанием. Допустимо присутствие обоих полей или одно из них. Ситуация, когда оба поля принимают неопределенное значение NULL неприемлема. Используя триггеры, добейтесь того, чтобы одно из этих полей или оба поля были заполнены. При попытке присвоить полям NULL-значение необходимо отменить операцию.

9. (по желанию) Напишите хранимую функцию для вычисления произвольного числа Фибоначчи. Числами Фибоначчи называется последовательность в которой число равно сумме двух предыдущих чисел. Вызов функции FIBONACCI(10) должен возвращать число 55.

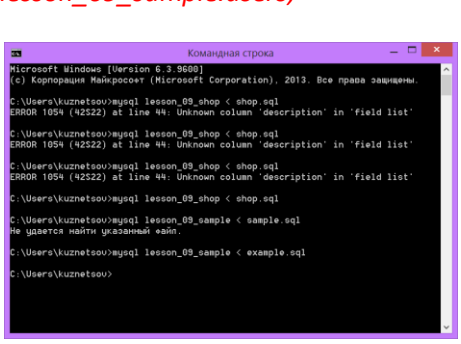
0	1	2	3	4	5	6	7	8	9	10
0	1	1	2	3	5	8	13	21	34	55

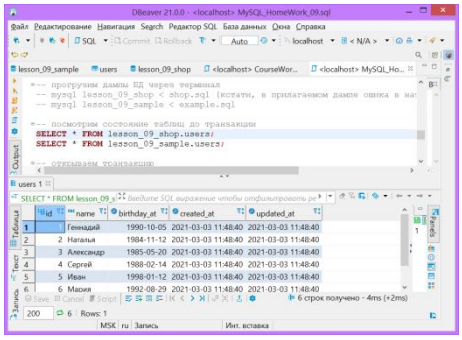
---

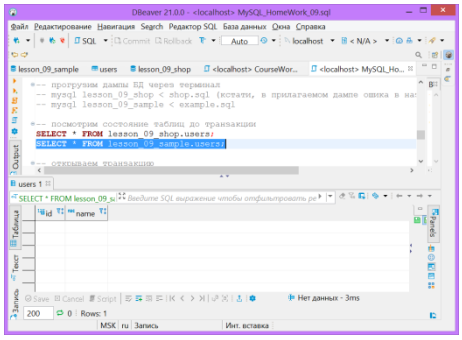
1. *В базе данных shop и sample присутствуют одни и те же таблицы, учебной базы данных. Переместите запись id = 1 из таблицы shop.users в таблицу sample.users. Используйте транзакции.*

Создадим БД «*CREATE DATABASE lesson_09_shop;*», «*CREATE DATABASE lesson_09_sample;*»
Прогрузим дампы БД через терминал «*mysql lesson_09_shop < shop.sql*» (кстати, в прилагаемом дампе ошибка в названии столбца "description"), «*mysql lesson_09_sample < example.sql*»

посмотрим состояние таблиц до транзакции «*SELECT * FROM lesson_09_shop.users;*», «*SELECT * FROM lesson_09_sample.users;*»

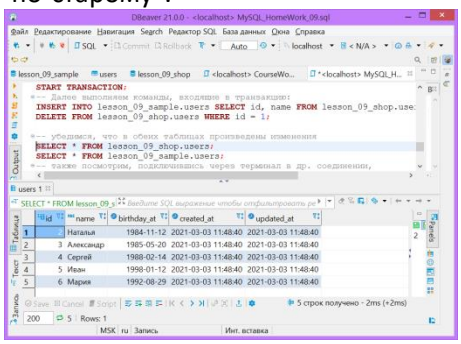


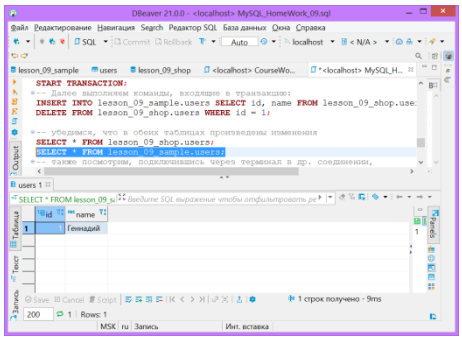


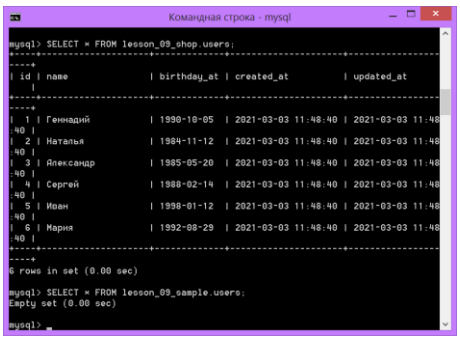


Открываем транзакцию «*START TRANSACTION;*». Далее выполняем команды, входящие в транзакцию:
INSERT INTO lesson_09_sample.users SELECT id, name FROM lesson_09_shop.users WHERE id = 1;
DELETE FROM lesson_09_shop.users WHERE id = 1;
Убедимся, что в обеих таблицах произведены изменения.

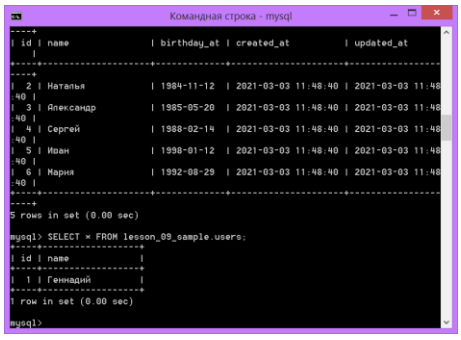
Также посмотрим, подключившись через терминал, что для др. пользователей таблицы выглядят "по-старому".







Закроем транзакцию «*COMMIT;*». Убедимся, что изменения вступили в силу (во второй консоли).

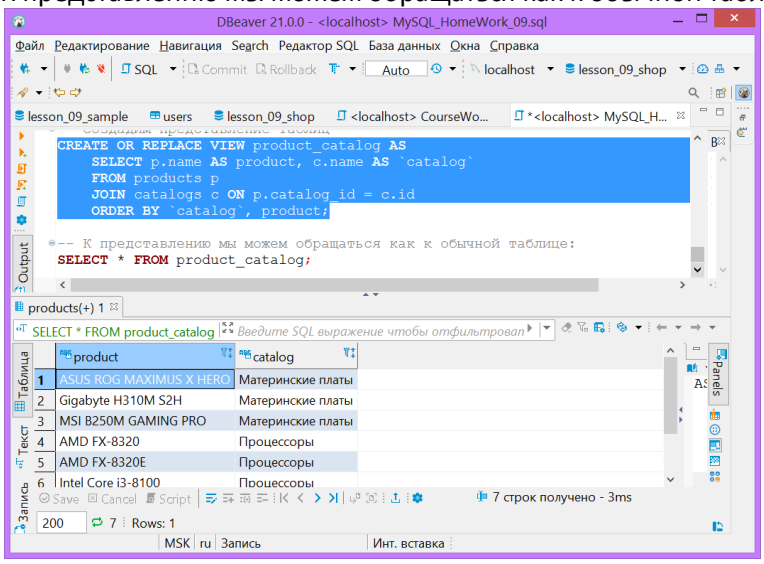


2. *Создайте представление, которое выводит название name товарной позиции из таблицы products и соответствующее название каталога name из таблицы catalogs.*

Создадим представление таблиц

```
CREATE OR REPLACE VIEW product_catalog AS
        SELECT p.name AS product, c.name AS `catalog`
        FROM products p
        JOIN catalogs c ON p.catalog_id = c.id
        ORDER BY `catalog`, product;
```

К представлению мы можем обращаться как к обычной таблице: «*SELECT * FROM product_catalog;*»



3. *Пусть имеется таблица с календарным полем created_at. В ней размещены разряженные календарные записи за август 2018 года '2018-08-01', '2018-08-04', '2018-08-16' и 2018-08-17. Составьте запрос, который выводит полный список дат за август, выставляя в соседнем поле значение 1, если дата присутствует в исходном таблице и 0, если она отсутствует.*

Создадим таблицу «*CREATE TABLE task_03 (created_at DATETIME);*»
Заполним ее произвольными значениями из интересуемого интервала (14 записей)

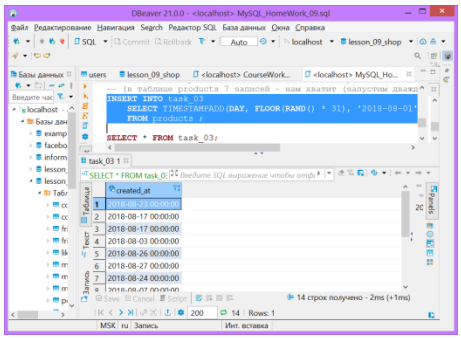
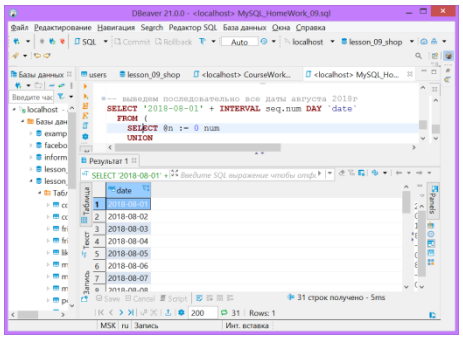
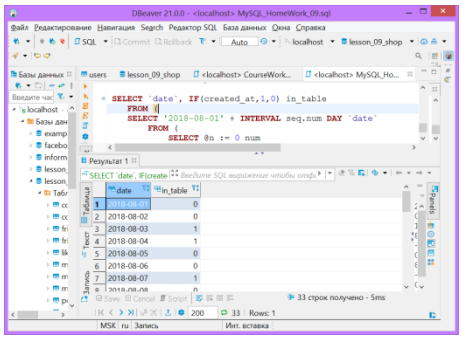
```
INSERT INTO task_03
        SELECT TIMESTAMPADD(DAY, FLOOR(RAND() * 31), '2018-08-01')
        FROM products ;
```

Воспользуемся "большой" таблицей в одной из схем «lesson_06.users (100 записей)» для вывода последовательно всех дат августа 2018г:

```
SELECT '2018-08-01' + INTERVAL seq.num DAY `date`
 FROM (
  SELECT @n := 0 num
  UNION
  SELECT @n := @n + 1 num
  FROM lesson_06.users
 ) seq
 WHERE seq.num BETWEEN 0 AND DATEDIFF(/*end_date*/'2018-08-31', /*begin_date*/'2018-08-01');
```

Объединим с запросом из нашей таблицы:

```
SELECT `date`, IF(created_at,1,0) in_table
        FROM (
        SELECT '2018-08-01' + INTERVAL seq.num DAY `date`
                FROM (
                  SELECT @n := 0 num
                  UNION
                  SELECT @n := @n + 1 num
                  FROM lesson_06.users
                        ) seq
                WHERE seq.num BETWEEN 0 AND DATEDIFF(/*end_date*/'2018-08-31', /*begin_date*/'2018-08-01')) all_date
        LEFT JOIN task_03 ON created_at = all_date.date;
```

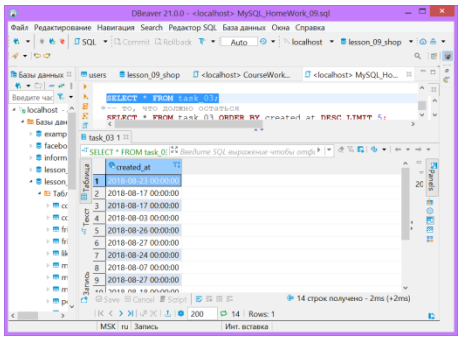
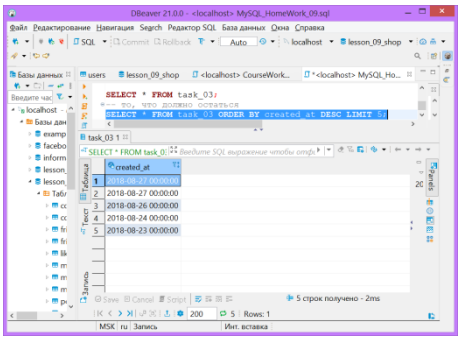
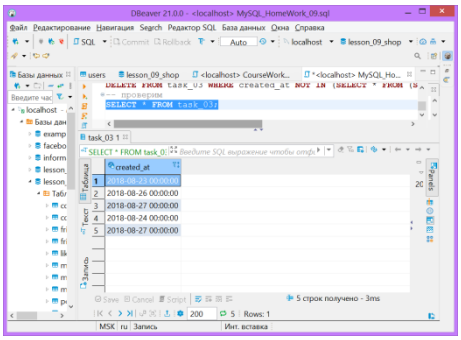
Несколько напрягает две вещи (НЕидеально). В запросе пришлось дважды указать дату начала периода «01.08.2018» (можно избавиться введя доп. переменную @date_begin , но запрос станет еще более громоздким). И, дополнительное обертывание первого запроса, т.к. иначе второй запрос не «понимает» колонку `date`, а ставить второй раз выражение «*ON created_at = '2018-08-01' + INTERVAL seq.num DAY*» - некрасиво.

4. *Пусть имеется любая таблица с календарным полем created_at. Создайте запрос, который удаляет устаревшие записи из таблицы, оставляя только 5 самых свежих записей.*

Попробуем решить задачу «в лоб». Таблица содержит 14 записей «*SELECT * FROM task_03;*»

Запрос «*SELECT * FROM task_03 ORDER BY created_at DESC LIMIT 5;*» выдает то, что должно остаться после удаления.

Удалим все, кроме выборки из предыдущего запроса «*DELETE FROM task_03 WHERE created_at NOT IN (SELECT * FROM (SELECT * FROM task_03 ORDER BY created_at DESC LIMIT 5) ttt);*»
Проверим «*SELECT * FROM task_03;*» - задача выполнена.

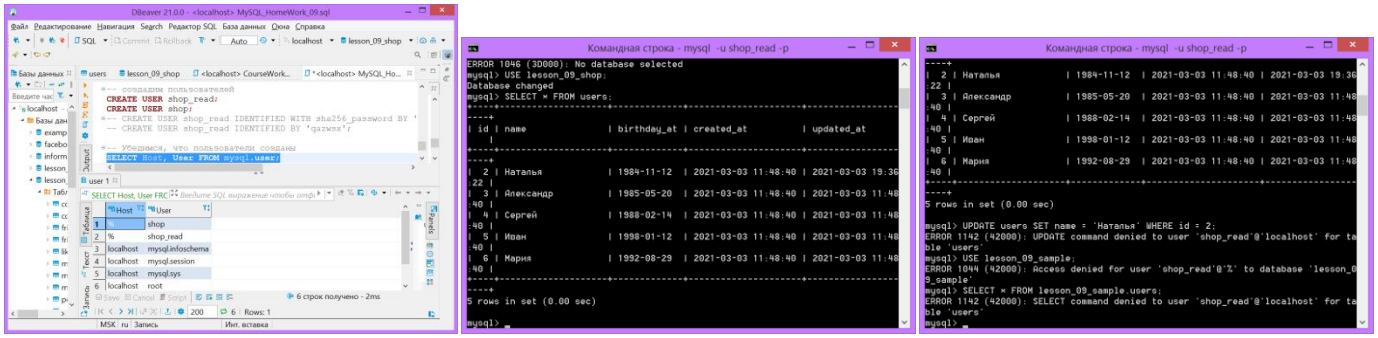




5. *Создайте двух пользователей которые имеют доступ к базе данных shop. Первому пользователю shop_read должны быть доступны только запросы на чтение данных, второму пользователю shop — любые операции в пределах базы данных shop.*

Создадим пользователей «*CREATE USER shop_read;*» и «*CREATE USER shop;*». Не будем пока заморачиваться с паролями доступа «*CREATE USER shop_read IDENTIFIED WITH sha256_password BY 'qazwsx';*». Убедимся, что пользователи созданы «*SELECT Host, User FROM mysql.user;*»

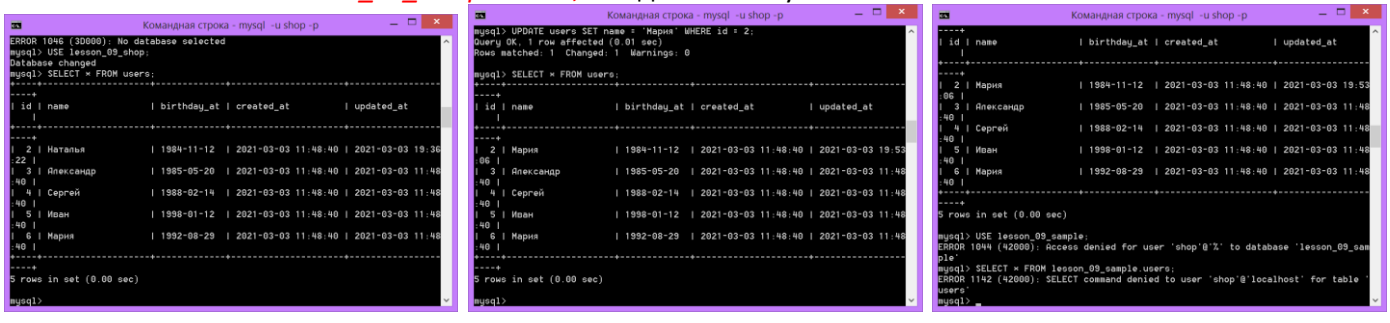
Предоставим пользователю shop_read права на чтение из любой таблицы БД "lesson_09_shop"
GRANT SELECT ON lesson_09_shop. TO shop_read;*
Проверим в терминале – все верно:

USE lesson_09_shop; - работает
*SELECT * FROM users;* - работает
UPDATE users SET name = 'Наталья' WHERE id = 2; - выдает ошибку
USE lesson_09_sample; - выдает ошибку
*SELECT * FROM lesson_09_sample.users;* - выдает ошибку



Предоставим пользователю shop любые операции в пределах БД "lesson_09_shop":
GRANT SELECT ON lesson_09_shop. TO shop_read;*
Проверим в терминале – все верно:

USE lesson_09_shop; - работает
*SELECT * FROM users;* - работает
UPDATE users SET name = 'Наталья' WHERE id = 2; - работает
USE lesson_09_sample; - выдает ошибку
*SELECT * FROM lesson_09_sample.users;* - выдает ошибку



6. ***Пусть имеется таблица accounts содержащая три столбца id, name, password, содержащие первичный ключ, имя пользователя и его пароль. Создайте представление username таблицы accounts, предоставляющий доступ к столбца id и name. Создайте пользователя user_read, который бы не имел доступа к таблице accounts, однако, мог бы извлекать записи из представления username.***

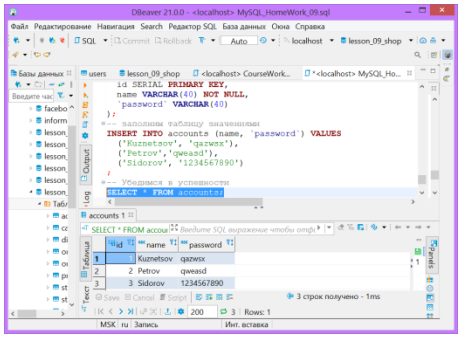
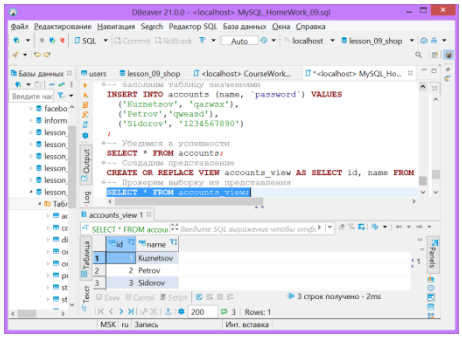
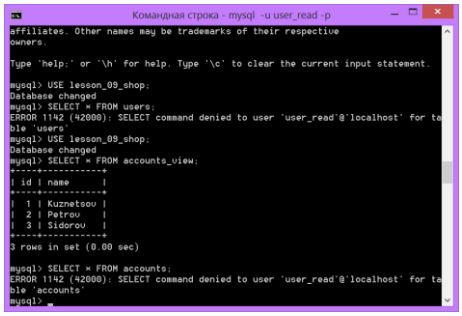
Создаем таблицу «accounts», заполним ее значениями и убедимся в успешности действий «*SELECT * FROM accounts;*».

Создадим представление «*CREATE OR REPLACE VIEW accounts_view AS SELECT id, name FROM accounts;*». Проверим выборку из представления «*SELECT * FROM accounts_view;*»

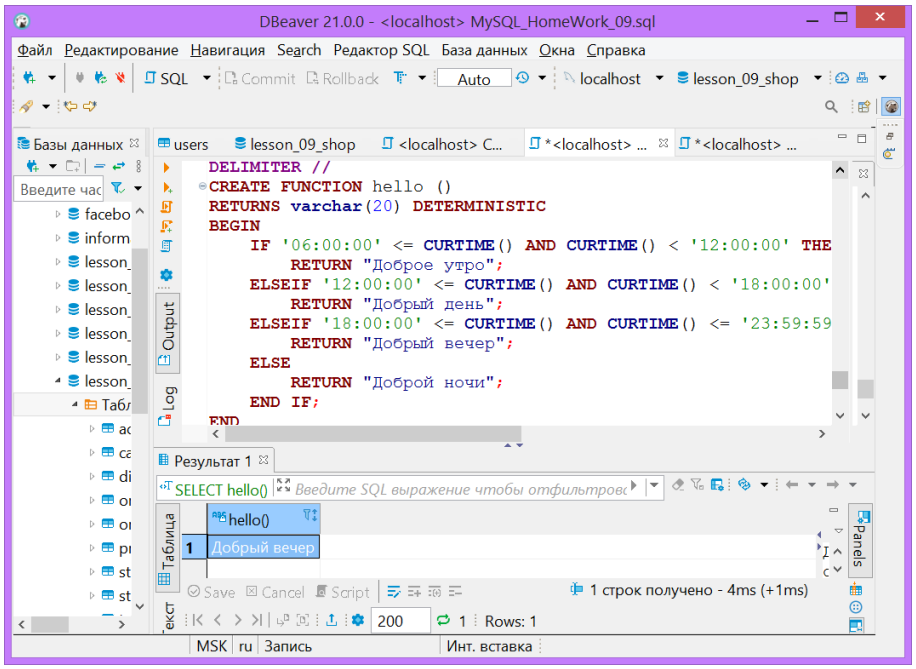
Создадим пользователя «*CREATE USER user_read;*». Предоставим пользователю права на чтение из представления «*GRANT SELECT ON lesson_09_shop.accounts_view TO user_read;*»

Проверим в терминале – все верно:

USE lesson_09_shop; - работает
*SELECT * FROM accounts_view;* - работает
*SELECT * FROM accounts;* - выдает ошибку

7. ***Создайте хранимую функцию hello(), которая будет возвращать приветствие, в зависимости от текущего времени суток. С 6:00 до 12:00 функция должна возвращать фразу "Доброе утро", с 12:00 до 18:00 функция должна возвращать фразу "Добрый день", с 18:00 до 00:00 — "Добрый вечер", с 00:00 до 6:00 — "Доброй ночи".***

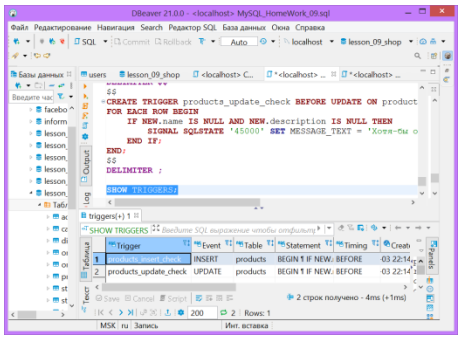
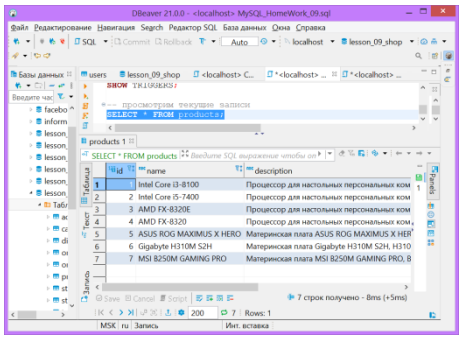
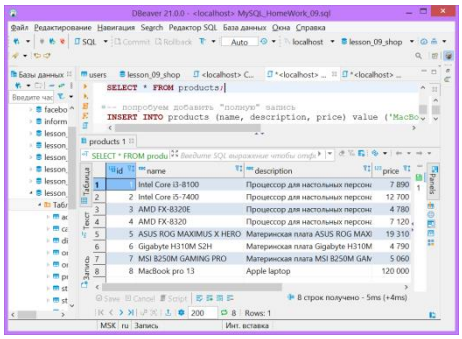


8. *В таблице products есть два текстовых поля: name с названием товара и description с его описанием. Допустимо присутствие обоих полей или одно из них. Ситуация, когда оба поля принимают неопределенное значение NULL неприемлема. Используя триггеры, добейтесь того, чтобы одно из этих полей или оба поля были заполнены. При попытке присвоить полям NULL-значение необходимо отменить операцию.*

Создаем триггеры «перед добавлением» и «перед изменением» таблицы «products», убедимся что они появились в БД «*SHOW TRIGGERS;*»

Текущее содержимое таблицы «*SELECT * FROM products;*»

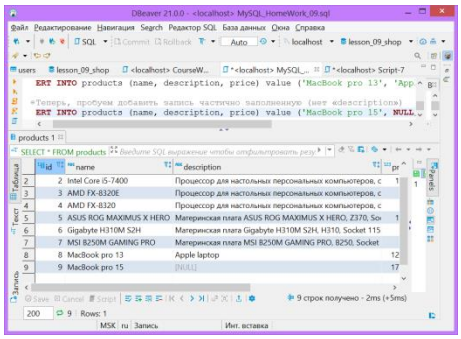
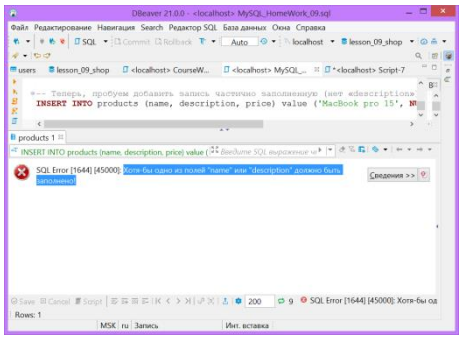
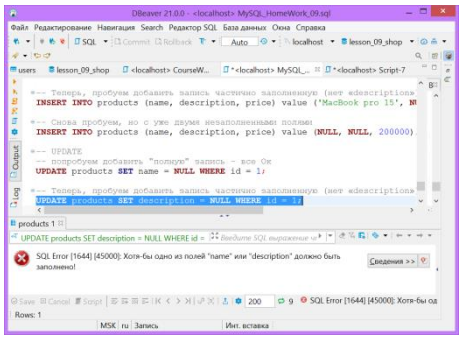
Пробуем вставить новую «полную» запись
«*INSERT INTO products (name, description, price) value ('MacBook pro 13', 'Apple laptop', 120000);*» - все Ок.

Теперь, пробуем добавить запись частично заполненную (нет «description»)
«*INSERT INTO products (name, description, price) value ('MacBook pro 15', NULL, 170000);*» - все Ок.

Снова пробуем, но с уже двумя незаполненными полями «*INSERT INTO products (name, description, price) value (NULL, NULL, 200000);*». Система выдала пользовательскую ошибку «*Хотя-бы одно из полей "name" или "description" должно быть заполнено!*»

Ситуация с UPDATE идентична…

9. *Напишите хранимую функцию для вычисления произвольного числа Фибоначчи. Числами Фибоначчи называется последовательность в которой число равно сумме двух предыдущих чисел. Вызов функции FIBONACCI(10) должен возвращать число 55.*

Создадим функцию:

```
DELIMITER //
CREATE FUNCTION FIBONACCI (num INT)
RETURNS bigint DETERMINISTIC
BEGIN
	DECLARE i, prev_1, prev_2, summ BIGINT DEFAULT 0;
	SET prev_1 = 1;
	IF num > 0 THEN
		cycle: WHILE i < num DO
			SET summ = prev_1 + prev_2;
			SET prev_1 = prev_2;
			SET prev_2 = summ;
			SET i = i + 1;
		END WHILE cycle;
		RETURN summ;
	ELSE
		RETURN 0;
	END IF;
END
//
DELIMITER ;

SELECT FIBONACCI(10);
```