

КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ГЕОЛОГИИ И НЕФТЕГАЗОВЫХ ТЕХНОЛОГИЙ
Кафедра геофизики и геоинформационных технологий

СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ MY SQL

Методические указания для практических работ по дисциплине

«Базы данных и системы управления базами данных»

Для магистрантов II курса

Казань – 2018

*Печатается по решению учебно-методической комиссии
института геологии и нефтегазовых технологий
Протокол № 3 от 29 января 2018 года*

Составитель:
доцент Сайфутдинова Г.М.

Система управления базами данных My SQL: Методические указания для практических работ по дисциплине «Базы данных и системы управления базами данных» / сост. Г.М. Сайфутдинова– Казань: КФУ, 2018. – 36 с.

Методические указания состоят из четырех лабораторных работ. Указания включают этапы проектирования баз данных, их заполнение, связывание таблиц. Рассмотрены типы функций и хранимые процедуры, а также работа с метаданными. В каждой лабораторной работе предусмотрены задания по вариантам. Каждое задание сопровождается методическими рекомендациями. Задания выполняются в условиях учебной лаборатории, оснащенной компьютерными технологиями и программными средствами. К каждой работе приводятся контрольные вопросы.

Методические указания предназначены для магистрантов, обучающихся по направлению 05.04.01 Геология.

© Сайфутдинова Г.М., 2018

© Казанский университет, 2018

Содержание

Введение.....	4
Лабораторная работа № 1.....	5
Лабораторная работа № 2.....	14
Лабораторная работа № 3.....	20
Лабораторная работа № 4.....	27
Список литературы.....	32
Приложение.....	33

Введение

Методические указания для практических работ по дисциплине «Базы данных и системы управления базами данных» предназначены для магистрантов II курса Института геологии и нефтегазовых технологий Казанского (Приволжского) федерального университета.

Основная цель практических занятий – применение и закрепление теоретических знаний, полученных в процессе обучения, необходимых при проектировании, разработке и использовании баз данных, а также для решения прикладных задач.

Методические указания состоят из четырех лабораторных работ по разработке, наполнению, редактированию баз данных и использованию основных операторов языка MySQL. Лабораторные работы состоят из заданий, соответствующих разделам программы дисциплины 05.04.01 – Геология. Каждая лабораторная работа сопровождается подробными методическими указаниями по их выполнению и пошаговому осуществлению команд. В приложении методических указаний приведены варианты заданий, которые выдаются на усмотрение преподавателя. По завершении каждой лабораторной работы необходимо ответить на контрольные вопросы и оформить отчет, согласно требованиям, предъявляемым к оформлению такого вида работ.

На выполнение четырех лабораторных работ выделяется 14 часов, в том числе на защиту и написание отчета по лабораторным работам.

Лабораторная работа №1.

РАБОТА С БАЗАМИ ДАННЫХ И ТАБЛИЦАМИ MYSQL

1. Цель работы

- Закрепление теоретических знаний по базам данных и таблицам.
- Приобретение практических навыков по созданию, изменению и удалению баз данных и таблиц.
- Создание основных таблиц базы данных предметной области.

2. Методика создания баз данных и работы с ней

2.1. Создание базы данных

Синтаксис команды:

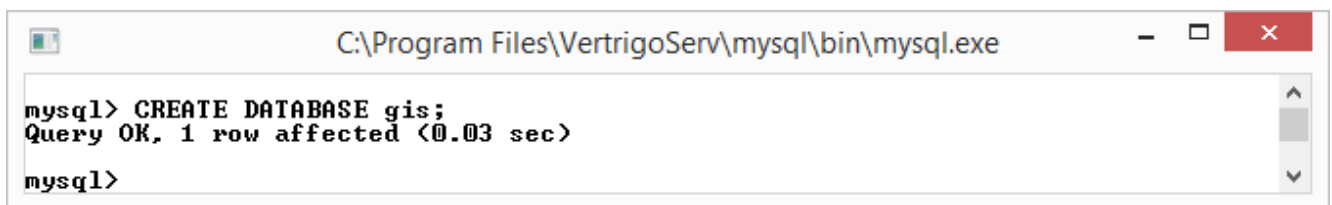
```
CREATE DATABASE [IF NOT EXISTS] db_name;
```

Создается база данных с указанным именем *db_name*. Если попытаться создать уже существующую базу данных, возникнет ошибка. Для предотвращения ошибки оператор `CREATE DATABASE` необходимо снабдить конструкцией `IF NOT EXISTS`, при наличии которой база данных создается, если она еще не существует, иначе – никаких действий не производится.

База данных в *MySQL* реализована в виде каталога, содержащего файлы, которые соответствуют таблицам в базе данных. `CREATE DATABASE` создает только каталог в каталоге данных *MySQL* и файл `db.opt`.

Имя может содержать основные латинские буквы (a-z, A-Z), цифры (0-9), символ доллара '\$', символ подчеркивания '_'. Имя может начинаться с цифры, но не должно состоять только из цифр, а также заканчиваться пробелами. Максимальная длина имени составляет 64 знака.

Пример 1. Создание базы данных с именем *GIS*.



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> CREATE DATABASE gis;
Query OK, 1 row affected (0.03 sec)
mysql>
```

При создании базы данных можно указать кодировку таблицам и столбцам по умолчанию. Для этого после имени базы данных следует указать ключевое слово `DEFAULT CHARACTER SET charset_name` – имя кодировки, например, `cp1251`, которая обозначает русскую *Windows*-кодировку.

Пример 2. Создание базы данных с именем *GIS* с кодировкой.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> CREATE DATABASE gis DEFAULT CHARACTER SET cp1251;
Query OK, 1 row affected (0.00 sec)
mysql>
```

2.2. Перечисление баз данных на сервере

Синтаксис команды:

```
SHOW DATABASES;
```

Оператор выводит список существующих баз данных.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| gis         |
| mysql       |
| performance_schema |
| phpmyadmin  |
| sys        |
+-----+
6 rows in set (0.00 sec)
mysql> _
```

2.3. Удаление базы данных

Синтаксис команды:

```
DROP DATABASE [IF EXISTS] db_name;
```

Удаляет все таблицы в базе данных `db_name` и удаляет саму базу данных. Ключевое слово `IF EXISTS` используется, для предотвращения возникновения ошибки, если база данных не существует.

Пример 3. Удаление базы данных с именем *GIS*.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> DROP DATABASE gis;
Query OK, 0 rows affected (0.00 sec)
mysql> _
```

2.4. Выбор базы данных

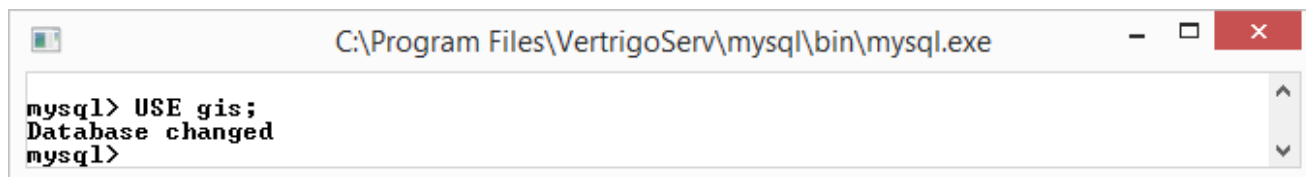
Перед созданием таблицы следует выбрать базу данных, с которой будет производиться работа. Выбирать базу данных необходимо в каждом сеансе работы с *MySQL*.

Синтаксис команды:

```
USE db_name;
```

Оператор сообщает *MySQL* имя базы данных *db_name* в качестве текущей базы данных по умолчанию для последующих операторов. База данных остается по умолчанию до конца.

Пример 4. Выборка базы данных с именем *GIS*.



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> USE gis;
Database changed
mysql>
```

2.5. Создание таблицы

Синтаксис команды:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition, ...);
create_definition:
    col_name column_definition
```

Создается таблица с именем *tbl_name*. Имя может содержать основные латинские буквы (a-z, A-Z), цифры (0-9), символ доллара '\$', символ подчеркивания '_'. Имя может начинаться с цифры, но не должно состоять только из цифр, а также заканчиваться пробелами. Максимальная длина имени составляет 64 знака.

Ключевое слово `IF NOT EXISTS` предотвращает появление ошибки, если таблица существует. Тем не менее, нет никакой гарантии, что существующая таблица имеет структуру, идентичную указанной в инструкции `CREATE TABLE`.

Ключевое слово `TEMPORARY` используется для создания временной таблицы. Временная таблица видна только в текущей сессии, и при закрытии сессии удаляется автоматически.

В круглых скобках через запятую указываются имена столбцов (*col_name*) и их тип (*data_type*).

```
column_definition:
    data_type [NOT NULL | NULL] [DEFAULT default_value]
    [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
    [COMMENT 'string']
    [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
    [STORAGE {DISK|MEMORY|DEFAULT}]
    [reference_definition]
```

Пример 5. Создание таблицы *STUDENTS* в базе данных *GIS*.

```

C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> CREATE TABLE students
-> <name VARCHAR(20) NOT NULL,
-> surname VARCHAR(30) NOT NULL,
-> sex ENUM('man', 'woman') NOT NULL,
-> birthday DATE NULL,
-> city VARCHAR(20) DEFAULT 'Казань',
-> address TEXT NULL,
-> group_num INT(3));
Query OK, 0 rows affected (0.07 sec)
mysql> _

```

Создается таблица STUDENTS с семью полями – name (имя), surname (фамилия), sex (пол), birthday (дата рождения), city (город), address (адрес проживания) и group_num (номер группы). Каждое имя поля снабжено описателем типа, характеризующее тип хранимых данных. В описание поля также можно вставлять дополнительные спецификаторы.

Для просмотра структуры таблиц служит оператор DESCRIBE.

```

C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> DESCRIBE students;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name  | varchar(20) | NO | | NULL | |
| surname | varchar(30) | NO | | NULL | |
| sex   | enum('man', 'woman') | NO | | NULL | |
| birthday | date | YES | | NULL | |
| city  | varchar(20) | YES | | Казань | |
| address | text | YES | | NULL | |
| group_num | int(3) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
mysql> _

```

Типы данных

Числовые типы-----

BOOL

0 или 1

TINYINT[(M)] [UNSIGNED] [ZEROFILL]

Очень малое целое число. Диапазон со знаком от -128 до 127. Диапазон без знака от 0 до 255.

SMALLINT[(M)] [UNSIGNED] [ZEROFILL]

Малое целое число. Диапазон со знаком от -32768 до 32767. Диапазон без знака от 0 до 65535.

MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]

Целое число среднего размера. Диапазон со знаком от -8388608 до 8388607. Диапазон без знака от 0 до 16777215.

INT[(*M*)] [*UNSIGNED*] [*ZEROFILL*]

Целое число нормального размера. Диапазон со знаком от -2147483648 до 2147483647. Диапазон без знака от 0 до 4294967295.

BIGINT[(*M*)] [*UNSIGNED*] [*ZEROFILL*]

Большое целое число. Диапазон со знаком от -9223372036854775808 до 9223372036854775807. Диапазон без знака от 0 до 18446744073709551615.

FLOAT[(*M,D*)] [*UNSIGNED*] [*ZEROFILL*]

Малое число с плавающей точкой обычной точности. Диапазон со знаком от -3,402823466E+38 до -1,175494351E-38. Диапазон без знака от 1,175494351E-38 до 3,402823466E+38. Если указан атрибут *UNSIGNED*, отрицательные значения недопустимы. Атрибут *M* указывает количество выводимых пользователю знаков, а атрибут *D* – количество разрядов, следующих за десятичной точкой. Обозначение *FLOAT* без указания аргументов или запись вида *FLOAT*(*X*), где $X \leq 24$ справедливы для числа с плавающей точкой обычной точности.

DOUBLE[(*M,D*)] [*UNSIGNED*] [*ZEROFILL*]

Число с плавающей точкой удвоенной точности нормального размера. Диапазон со знаком от -1,7976931348623157E+308 до -2,2250738585072014E-308. Диапазон без знака от 2,2250738585072014E-308 до 1,7976931348623157E+308. Если указан атрибут *UNSIGNED*, отрицательные значения недопустимы. Атрибут *M* указывает количество выводимых пользователю знаков, а атрибут *D* – количество разрядов, следующих за десятичной точкой. Обозначение *DOUBLE* без указания аргументов или запись вида *DOUBLE* (*X*), где $25 \leq X \leq 53$ справедливы для числа с плавающей точкой двойной точности.

Календарные типы -----

DATE

Используется для величин с информацией только о дате. *MySQL* извлекает и выводит величины *DATE* в формате 'YYYY-MM-DD'. Поддерживается диапазон величин от '1000-01-01' до '9999-12-31'.

TIME

Используется для величин с информацией только о времени. *MySQL* извлекает и выводит величины *TIME* в формате 'HH:MM:SS'. Поддерживается диапазон величин от '-838:59:59' до '838:59:59'.

DATETIME

Комбинация даты и времени. Поддерживается интервал от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'. *MySQL* выводит значения *DATETIME* в

формате 'YYYY-MM-DD HH:MM:SS', но можно устанавливать значения в столбце *DATETIME*, используя как строки, так и числа.

YEAR [(2|4)]

Год в двухзначном или четырехзначном форматах (по умолчанию формат четырехзначный). Допустимы следующие значения: с 1901 по 2155, 0000 для четырехзначного формата года и 1970 - 2069 при использовании двухзначного формата (70 - 69). *MySQL* выводит значения *YEAR* в формате *YYYY*, но можно задавать значения в столбце *YEAR*, используя как строки, так и числа.

Строковые типы-----

CHAR [(M)] [*BINARY*]

Строка фиксированной длины, которая справа дополняется пробелами до указанной длины, при хранении. Диапазон длины от 1 до 255 символов. Завершающие пробелы удаляются, когда значение извлекается. Значения *CHAR* сортируются и сравниваются без учета регистра в зависимости от кодировки по умолчанию, если не установлен флаг *BINARY*.

VARCHAR (M) [*BINARY*]

Строка переменной длины. Примечание: конечные пробелы удаляются при сохранении. Диапазон длины от 1 до 255 символов. При хранении величин используется только то количество символов, которое необходимо, плюс один байт для записи длины. Значения *VARCHAR* сортируются и сравниваются без учета регистра, если не установлен флаг *BINARY*.

TINYTEXT [*BINARY*]

Строка текста с максимальной длиной 255 символов. Сортировка и сравнение данных выполняются без учета регистра для величин, если не установлен флаг *BINARY*.

TEXT [*BINARY*]

Строка текста с максимальной длиной 65535 символов. Сортировка и сравнение данных выполняются без учета регистра для величин, если не установлен флаг *BINARY*.

MEDIUMTEXT [*BINARY*]

Строка текста с максимальной длиной 16777215 символов. Сортировка и сравнение данных выполняются без учета регистра для величин, если не установлен флаг *BINARY*.

LONGTEXT [*BINARY*]

Строка текста с максимальной длиной 4294967295 символов. Сортировка и сравнение данных выполняются без учета регистра для величин, если не установлен флаг *BINARY*.

```
ENUM(value1, value2, value3, ...)
```

Перечисление – может принимать значение из списка допустимых значений, явно перечисленных в спецификации столбца в момент создания таблицы. Этим значением также может быть пустая строка ("") или *NULL*. Перечисление может иметь максимум 65535 элементов.

```
SET(value1, value2, value3, ...)
```

Строковый тип, который может принимать ноль или более значений, каждое из которых должно быть выбрано из списка допустимых значений, определенных при создании таблицы. Элементы множества *SET* разделяются запятыми. Как следствие, сами элементы множества не могут содержать запятых.

TINYBLOB, *BLOB*, *MEDIUMBLOB*, *LOB* – представляет собой двоичный объект, который может содержать переменное количество данных. По размеру аналогичны типу *TEXT*.

M – указывает максимальный размер вывода в символах. Максимально допустимый размер вывода составляет 255 символов.

D – употребляется для типов данных с плавающей точкой и указывает количество разрядов, следующих за десятичной точкой. Максимально возможная величина составляет 30 разрядов, но не может быть больше, чем *M* - 2.

UNSIGNED – беззнаковое число, *ZEROFILL* – свободные позиции слева заполняются нулями.

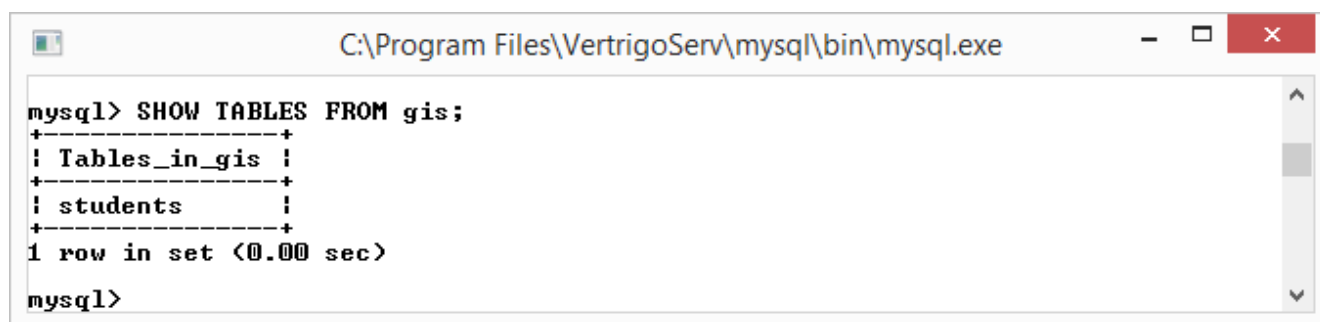
2.6. Просмотр таблиц

Синтаксис команды:

```
SHOW TABLES [FROM db_name]
```

Выводит список таблиц указанной базы данных *db_name*. Если имя базы данных не указано, выводится список текущей базы данных.

Пример 6. Получение списка таблиц базы данных *GIS*.



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> SHOW TABLES FROM gis;
+-----+
| Tables_in_gis |
+-----+
| students      |
+-----+
1 row in set (0.00 sec)
mysql>
```

2.7. Удаление таблицы

Синтаксис команды:

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

Оператор `DROP TABLE` удаляет одну или несколько таблиц. Все табличные данные и определения удаляются, так что будьте внимательны при работе с этой командой! Можно использовать ключевые слова `IF EXISTS`, чтобы предупредить ошибку, если указанные таблицы не существуют. Опции `RESTRICT` и `CASCADE` позволяют упростить перенос программы. В данный момент они не задействованы.

Опция `TEMPORARY` работает следующим образом: уничтожает только временные таблицы, не закрывает открытую транзакцию, права доступа не проверяются.

Использование слова `TEMPORARY` – это хороший способ удостовериться, что вы случайно не уничтожите настоящую таблицу.

2.8. Изменение структуры таблицы

Синтаксис команды:

```
ALTER TABLE tbl_name alter_spec [, alter_spec ...]
```

Сразу после оператора следует имя таблицы, которая подвергается изменению, и производимое изменение `alter_spec`, которых может быть несколько.

Пример 7. Добавление нового столбца *foto* в таблицу *STUDENTS*.



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> ALTER TABLE students ADD COLUMN foto BLOB;
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql>
```

Пример 8. Удаление столбца *foto* из таблицы *STUDENTS*.



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> ALTER TABLE students DROP COLUMN foto;
Query OK, 0 rows affected (0.20 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> _
```

Также существуют возможности добавлять столбцы в определенное место по отношению к другим столбцам, изменять название столбцов, их тип.

3. Задание на лабораторную работу

Создать базу данных предметной области согласно методическим рекомендациям и варианта приведенного в приложении.

Оформить отчет, содержащий подробное описание выполненной работы, скрины диалоговых окон (команд) и вывод по лабораторной работе.

4. Контрольные вопросы

1. Дайте определение понятиям База данных и Система управления базами данных.
2. Назовите команды создания и удаления базы данных. В чем разница создания базы данных и таблицы?
3. Какие типы данных используются при работе с *MySQL*?

Лабораторная работа №2

ВСТАВКА, ИЗМЕНЕНИЕ И УДАЛЕНИЕ ДАННЫХ

1. Цель работы

- Закрепление теоретических знаний по основным операторам: вставка, изменение и удаление данных.
- Приобретение практических навыков использования основных операторов манипулирования данными.
- Написание запросов по работе с данными предметной области.

2. Методика внесения, изменение и удаления данных в таблице

2.1. Добавление данных

После успешного создания базы данных и таблиц перед разработчиком встает задача заполнения таблиц данными. Традиционно для осуществления этой операции применяют три подхода:

- однострочный оператор *INSERT* – добавляет в таблицу данных новую запись;
- многострочный оператор *INSERT* – добавляет в таблицу данных несколько новых записей;
- пакетная загрузка данных – добавляет в таблицу данных из внешнего файла.

2.1.1. Однострочный оператор *INSERT*

Синтаксис команды:

```
INSERT [INTO] tbl_name [(col_name, ...)] VALUES (expr, ...)
```

Данный оператор вставляет строку в таблицу *tbl_name*, список столбцов, в которые вносятся значения, указываются через запятую после имени таблицы, а значения столбцов указываются в скобках через запятую после ключевого слова *VALUES*. Если не указаны столбцы, то должны быть определены значения для всех столбцов в списке *VALUES()*. Любой столбец, для которого явно не указано значение, будет установлен в свое значение по умолчанию.

Выражение *expr* может относиться к любому столбцу, который ранее был внесен в список значений. Например, можно указать следующее:

```
INSERT INTO tbl_name (col1, col2) VALUES(15, col1*2);
```

Но нельзя указать:

```
INSERT INTO tbl_name (col1, col2) VALUES(col2*2, 15);
```

Пример 1. Добавление одной строки в таблицу *STUDENTS*.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> INSERT students (name, surname, sex)
-> VALUES ('Пётр', 'Иванов', 'man');
Query OK, 1 row affected (0.00 sec)
mysql>
```

Выполним оператор выборки данных из таблицы *STUDENTS* для просмотра введенной строки:

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> SELECT * FROM students;
+----+-----+-----+-----+-----+-----+-----+
| name | surname | sex | birthday | city | address | group_num |
+----+-----+-----+-----+-----+-----+-----+
| Пётр | Иванов | man | NULL | Казань | NULL | NULL |
+----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
mysql>
```

Пример 2. Добавление одной строки со значением *DEFAULT* в таблицу *STUDENTS*.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> INSERT students (name, surname, sex, city, group_num)
-> VALUES ('Ольга', 'Казакова', 'woman', DEFAULT, 102);
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM students;
+----+-----+-----+-----+-----+-----+-----+
| name | surname | sex | birthday | city | address | group_num |
+----+-----+-----+-----+-----+-----+-----+
| Пётр | Иванов | man | NULL | Казань | NULL | NULL |
| Ольга | Казакова | woman | NULL | Казань | NULL | 102 |
+----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
mysql>
```

2.1.2. Многострочный оператор *INSERT*

Многострочный оператор *INSERT* совпадает с однострочным оператором. В нем после ключевого слова *VALUES* добавляется несколько expr.

Пример 3. Добавление несколько строк в таблицу *STUDENTS*.

```

C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> INSERT students (name, surname, sex, group_num)
-> VALUES ('Анна', 'Андреева', 'woman', 103),
-> ('Игорь', 'Васильев', 'man', 103),
-> ('Владимир', 'Романов', 'man', 104);
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM students;
+----+-----+-----+-----+-----+-----+-----+
| name      | surname | sex   | birthday | city   | address | group_num |
+----+-----+-----+-----+-----+-----+-----+
| Пётр     | Иванов  | man   | NULL     | Казань | NULL    | NULL      |
| Ольга    | Казакова | woman | NULL     | Казань | NULL    | 102      |
| Анна     | Андреева | woman | NULL     | Казань | NULL    | 103      |
| Игорь    | Васильев | man   | NULL     | Казань | NULL    | 103      |
| Владимир | Романов  | man   | NULL     | Казань | NULL    | 104      |
+----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>

```

2.1.3. Пакетная загрузка данных

Упрощенный синтаксис команды:

```
LOAD DATA INFILE 'file_name';
```

Поля в загружаемом файле должны быть разделены символом табуляции. Даже при наличии в файле только одного столбца с данными, в конце строки обязательно должен быть символ табуляции. Значение *NULL* обозначается как \N.

Пример 4. Загрузка из файла.

```

C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> LOAD DATA INFILE 'f:\test.txt'
-> INTO TABLE students;
Query OK, 5 rows affected, 4 warnings (0.00 sec)
Records: 5 Deleted: 0 Skipped: 0 Warnings: 4

mysql> SELECT * FROM students;
+----+-----+-----+-----+-----+-----+-----+
| name      | surname | sex   | birthday | city   | address | group_num |
+----+-----+-----+-----+-----+-----+-----+
| Пётр     | Иванов  | man   | NULL     | Казань | NULL    | NULL      |
| Ольга    | Казакова | woman | NULL     | Казань | NULL    | 102      |
| Анна     | Андреева | woman | NULL     | Казань | NULL    | 103      |
| Игорь    | Васильев | man   | NULL     | Казань | NULL    | 103      |
| Владимир | Романов  | man   | NULL     | Казань | NULL    | 104      |
| Кирилл   | Борисов  | man   | NULL     | Москва | NULL    | 201      |
| Мария    | Никитина | woman | NULL     | Уфа    | NULL    | 201      |
| Татьяна  | Климова  | woman | NULL     | Саратов | NULL    | 202      |
| Татьяна  | Журавлева | woman | NULL     | Омск   | NULL    | 202      |
| Игорь    | Захаров  | man   | NULL     | Москва | NULL    | 202      |
+----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> _

```

При формировании пути к файлу в операционной системе *Windows* обратные слэши необходимо экранировать, этого можно избежать, если использовать прямые слэши.

Пример 5. Загрузка из файла.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> LOAD DATA INFILE 'f:/test.txt'
-> INTO TABLE students;
Query OK, 5 rows affected, 4 warnings (0.00 sec)
Records: 5 Deleted: 0 Skipped: 0 Warnings: 4

mysql> SELECT * FROM students;
+-----+-----+-----+-----+-----+-----+-----+
| name   | surname | sex   | birthday | city   | address | group_num |
+-----+-----+-----+-----+-----+-----+-----+
| Пётр  | Иванов  | man   | NULL     | Казань | NULL    | NULL     |
| Ольга | Казакова | woman | NULL     | Казань | NULL    | 102     |
| Анна  | Андреева | woman | NULL     | Казань | NULL    | 103     |
| Игорь | Васильев | man   | NULL     | Казань | NULL    | 103     |
| Владимир | Романов | man   | NULL     | Казань | NULL    | 104     |
| Кирилл | Борисов | man   | NULL     | Москва | NULL    | 201     |
| Мария | Никитина | woman | NULL     | Уфа    | NULL    | 201     |
| Татьяна | Климова | woman | NULL     | Саратов | NULL    | 202     |
| Татьяна | Журавлева | woman | NULL     | Омск   | NULL    | 202     |
| Игорь | Захаров | man   | NULL     | Москва | NULL    | 202     |
+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

2.2. Изменение данных

Синтаксис команды:

```
UPDATE tbl_name
  SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}]
  ...
  [WHERE where_condition]
```

Оператор UPDATE обновляет столбцы в соответствии с их новыми значениями в строках выбранной таблицы. В выражении SET указывается, какие именно столбцы следует модифицировать и какие величины должны быть в них установлены. В выражении WHERE, если оно присутствует, задается, какие строки подлежат обновлению. В остальных случаях обновляются все строки. Каждое значение может быть задано как выражение, либо значением по умолчанию с помощью ключевого слова DEFAULT, чтобы установить столбец явным образом на значение по умолчанию. Предложение WHERE, если задано, определяет условия, которые определяют, какие строки будут обновляться. Без предложения WHERE, обновляются все строки.

Пример 6. Обновление одной строки в таблице STUDENTS.

```

C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> UPDATE students SET birthday='1996-03-17', group_num=101
-> WHERE name='Пётр';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM students;
+----+-----+-----+-----+-----+-----+-----+
| name      | surname | sex   | birthday | city   | address | group_num |
+----+-----+-----+-----+-----+-----+-----+
| Пётр     | Иванов  | man   | 1996-03-17 | Казань | NULL    | 101       |
| Ольга    | Казакова | woman | NULL      | Казань | NULL    | 102       |
| Анна     | Андреева | woman | NULL      | Казань | NULL    | 103       |
| Игорь    | Васильев | man   | NULL      | Казань | NULL    | 103       |
| Владимир | Романов  | man   | NULL      | Казань | NULL    | 104       |
| Кирилл   | Борисов  | man   | NULL      | Москва | NULL    | 201       |
| Мария    | Никитина | woman | NULL      | Уфа    | NULL    | 201       |
| Татьяна  | Климова  | woman | NULL      | Саратов | NULL    | 202       |
| Татьяна  | Журавлева | woman | NULL      | Омск   | NULL    | 202       |
| Игорь    | Захаров  | man   | NULL      | Москва | NULL    | 202       |
+----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> _

```

2.3. Удаление данных

Синтаксис команды:

```
DELETE FROM tbl_name
      [WHERE where_condition]
```

Оператор DELETE удаляет из таблицы *tbl_name* строки, удовлетворяющие заданным в *where_definition* условиям, и возвращает число удаленных записей.

Если оператор DELETE запускается без определения WHERE, то удаляются все строки.

Пример 7. Удаление строки в таблице *STUDENTS*.

```

C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> DELETE FROM students
-> WHERE surname='Иванов';
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM students;
+----+-----+-----+-----+-----+-----+-----+
| name      | surname | sex   | birthday | city   | address | group_num |
+----+-----+-----+-----+-----+-----+-----+
| Ольга    | Казакова | woman | NULL      | Казань | NULL    | 102       |
| Анна     | Андреева | woman | NULL      | Казань | NULL    | 103       |
| Игорь    | Васильев | man   | NULL      | Казань | NULL    | 103       |
| Владимир | Романов  | man   | NULL      | Казань | NULL    | 104       |
| Кирилл   | Борисов  | man   | NULL      | Москва | NULL    | 201       |
| Мария    | Никитина | woman | NULL      | Уфа    | NULL    | 201       |
| Татьяна  | Климова  | woman | NULL      | Саратов | NULL    | 202       |
| Татьяна  | Журавлева | woman | NULL      | Омск   | NULL    | 202       |
| Игорь    | Захаров  | man   | NULL      | Москва | NULL    | 202       |
+----+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)

mysql>

```

Для полного очищения таблицы используется оператор TRUNCATE TABLE *tbl_name*;

3. Задание на лабораторную работу

Научиться заполнять, изменять и удалять данные в таблицах согласно методическим рекомендациям. Заполнить таблицы базы данных, созданные в лабораторной работе №1.

Оформить отчет, содержащий подробное описание выполненной работы, скрины диалоговых окон (команд) и вывод по лабораторной работе.

4. Контрольные вопросы

4. Чем отличается реляционная модель данных от объектно-ориентированной модели?
5. Какое количество таблиц может содержать реляционная модель данных?
6. Какую команду выполняет однострочный оператор *INSERT*?
7. Для чего используется ключевое слово *SET*?
8. При помощи, какой команды осуществляется изменение данных?
9. При помощи, какой команды осуществляется удаление данных?

Лабораторная работа №3

ВЫБОРКА ДАННЫХ

1. Цель работы

- Закрепление теоретических знаний по выборке данных.
- Приобретение практических навыков по работе с запросами.
- Построение основных запросов поиска информации в БД предметной области.

2. Использование основных операторов выборки, сортировки, ограничения выборки, группировки записей и сохранения результатов выборки во внешний файл

2.1. Оператор *SELECT*

Выборка из таблиц определенных данных с заданными условиями выполняется оператором *SELECT*.

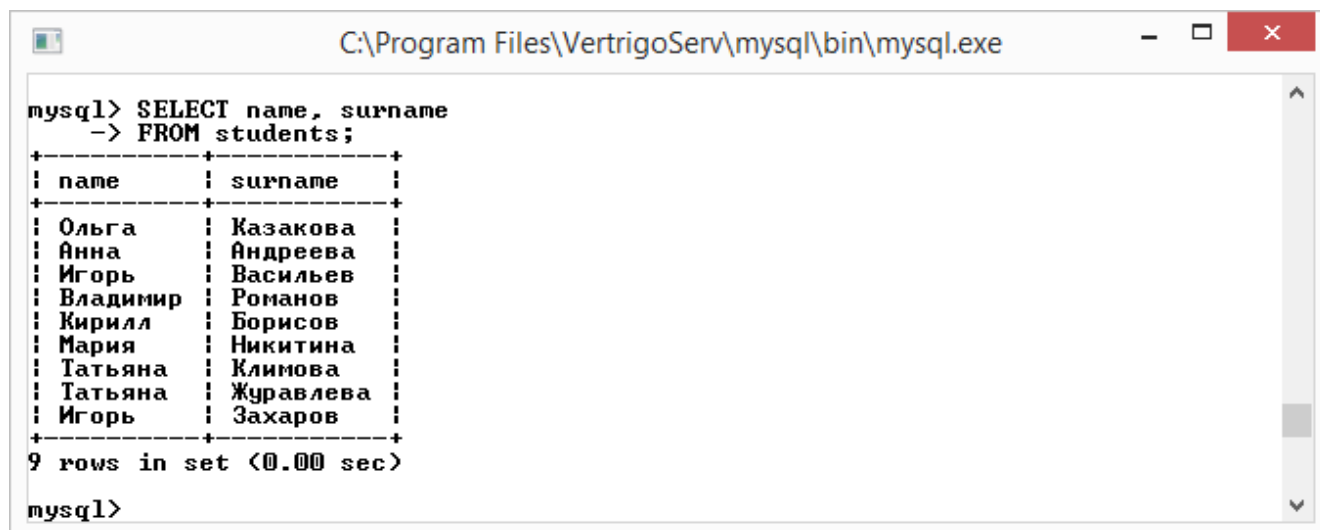
Упрощённый синтаксис команды:

```
SELECT select_expr [, select_expr ...]
      FROM table_references
      WHERE where_condition;
```

SELECT, используется для извлечения строк, выбранных из одной или нескольких таблиц. Выражение *select_expr* указывает на столбец, который необходимо получить. Должно быть, по крайней мере, одно выражение *select_expr*. Выражение *table_references* указывает на таблицу или таблицы, из которых для извлекаются строки. Ключевое слово *WHERE* указывает условие *where_condition* или условия, которым должны удовлетворять выбранные строки.

Оператор выбирает все строки, если нет *WHERE*.

Пример 1. Выборка записей из таблицы.

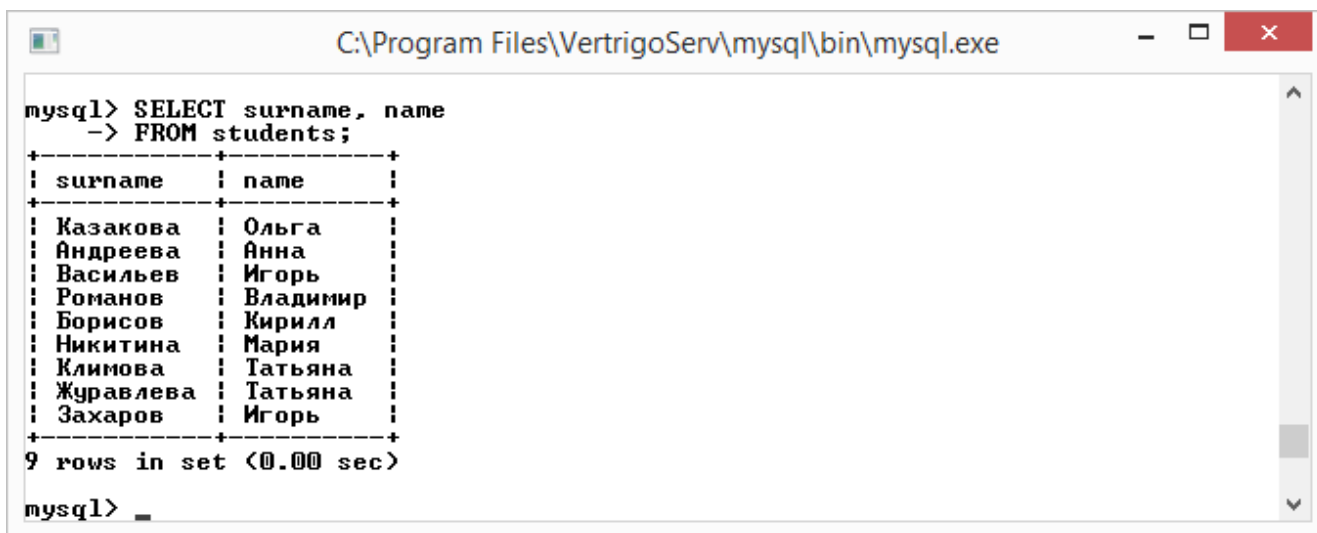


```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> SELECT name, surname
-> FROM students;
+-----+-----+
| name  | surname |
+-----+-----+
| Ольга | Казакова |
| Анна  | Андреева |
| Игорь | Васильев |
| Владимир | Романов |
| Кирилл | Борисов |
| Мария | Никитина |
| Татьяна | Климова |
| Татьяна | Журавлева |
| Игорь | Захаров |
+-----+-----+
9 rows in set (0.00 sec)
mysql>
```

В данном примере выводятся поля записей *name* и *surname*.

Данный оператор позволяет изменить порядок следования (вывода) полей, т.е. последовательность полей необязательно должна быть такой же, как она прописана в таблице.

Пример 2. Выборка записей из таблицы с измененной последовательностью полей.



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

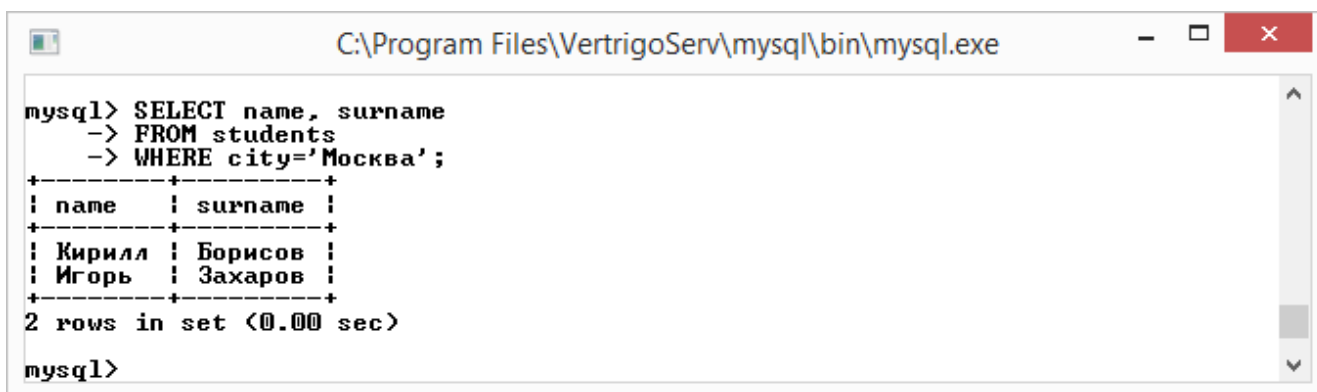
mysql> SELECT surname, name
-> FROM students;
+-----+-----+
| surname | name |
+-----+-----+
| Казакова | Ольга |
| Андреева | Анна |
| Васильев | Игорь |
| Романов | Владимир |
| Борисов | Кирилл |
| Никитина | Мария |
| Климова | Татьяна |
| Журавлева | Татьяна |
| Захаров | Игорь |
+-----+-----+
9 rows in set (0.00 sec)

mysql> _
```

2.2. Условия выборки

Операция, когда необходимо вывести записи из таблиц по одному или нескольким критериям используется чаще, чем выборка всех записей. Для ввода ограничений в операторе вводится ключевое слово *WHERE*, после которого следует логическое условие. Если запись удовлетворяет данному условию, она попадает в результат выборки, иначе такая запись отбрасывается.

Пример 3. Выборка строк из таблицы с условием.



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> SELECT name, surname
-> FROM students
-> WHERE city='Москва';
+-----+-----+
| name | surname |
+-----+-----+
| Кирилл | Борисов |
| Игорь | Захаров |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

В данном примере выводятся поля *name* и *surname* для записей, для которых в поле *city* указано 'Москва'.

Условие может быть составным и объединяться при помощи логических операторов.

Пример 4. Выборка строк из таблицы с составным условием.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> SELECT name, surname
-> FROM students
-> WHERE city='Казань' AND sex='woman';
+-----+-----+
| name | surname |
+-----+-----+
| Ольга | Казакова |
| Анна  | Андреева |
+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

Пример 5. Использование конструкции *BETWEEN*.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> SELECT name, surname, group_num
-> FROM students
-> WHERE group_num BETWEEN 100 AND 199;
+-----+-----+-----+
| name | surname | group_num |
+-----+-----+-----+
| Ольга | Казакова | 102 |
| Анна  | Андреева | 103 |
| Игорь | Васильев | 103 |
| Владимир | Романов | 104 |
+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>
```

Существует противоположенная конструкция *NOT BETWEEN*, которая возвращает записи, не попавшие в интервал.

Пример 6. Использование конструкции *NOT BETWEEN*.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> SELECT name, surname, group_num
-> FROM students
-> WHERE group_num NOT BETWEEN 100 AND 199;
+-----+-----+-----+
| name | surname | group_num |
+-----+-----+-----+
| Кирилл | Борисов | 201 |
| Мария | Никитина | 201 |
| Татьяна | Климова | 202 |
| Татьяна | Журавлева | 202 |
| Игорь | Захаров | 202 |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> _
```

Есть возможность выбирать записи, соответствующие не диапазону, а некоторому списку в условии. Данный способ реализуется использованием ключевого слова *IN*.

Пример 7. Использование конструкции IN.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> SELECT name, surname, group_num
-> FROM students
-> WHERE group_num IN(102, 202);
+-----+-----+-----+
| name   | surname | group_num |
+-----+-----+-----+
| Ольга  | Казакова | 102       |
| Татьяна | Климова  | 202       |
| Татьяна | Журавлева | 202       |
| Игорь   | Захаров  | 202       |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Существует противоположенная конструкция NOT IN, которая возвращает записи, не попавшие в список.

Пример 8. Использование конструкции NOT IN.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> SELECT name, surname, group_num
-> FROM students
-> WHERE group_num NOT IN(102, 202);
+-----+-----+-----+
| name   | surname | group_num |
+-----+-----+-----+
| Анна   | Андреева | 103       |
| Игорь   | Васильев | 103       |
| Владимир | Романов  | 104       |
| Кирилл  | Борисов  | 201       |
| Мария   | Никитина | 201       |
+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Оператор SELECT можно использовать для извлечения строк, вычисленных без ссылки на какую-либо таблицу.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> SELECT (3+2)*4;
+-----+
| (3+2)*4 |
+-----+
| 20      |
+-----+
1 row in set (0.00 sec)

mysql>
```

Для более осмысленного названия результирующего столбца можно использовать спецификатор AS, после которого следуют псевдоним.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> SELECT (3+2)*4 AS result;
+-----+
| result |
+-----+
|      20 |
+-----+
1 row in set (0.00 sec)
mysql> _
```

2.3. Сортировка

Очень часто интерес представляет результат выборки, отсортированный определенным образом по каким-либо столбцам. Это достигается применением конструкции ORDER BY. После этой конструкции следует имя столбца, по которому следует сортировать данные.

Пример 9. Сортировка по одному столбцу.

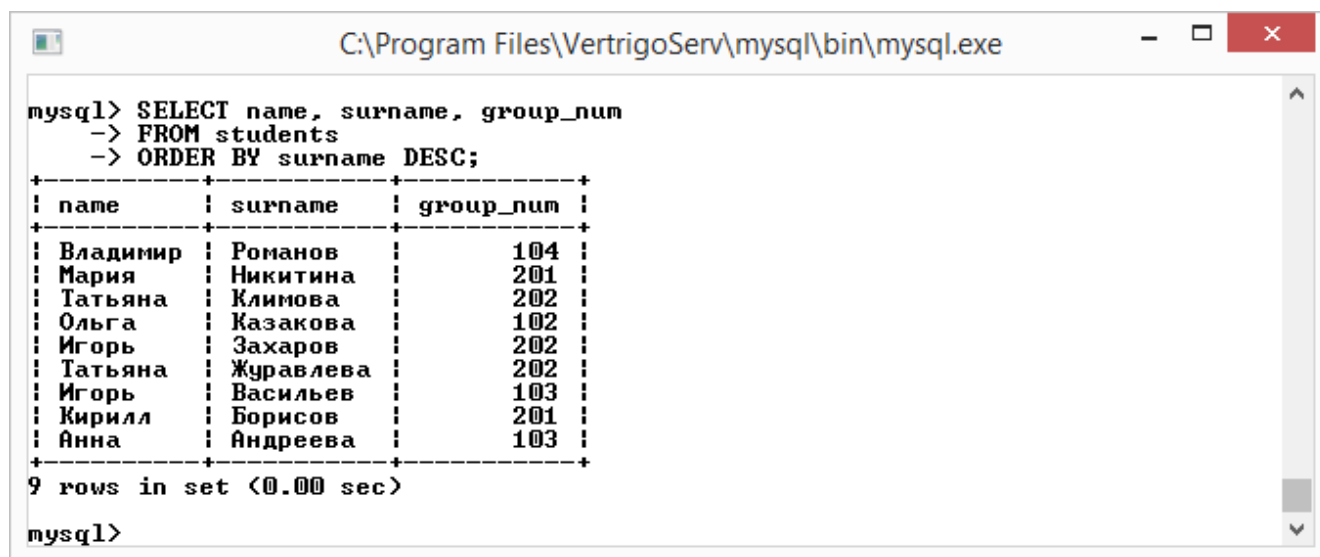
```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> SELECT name, surname, group_num
-> FROM students
-> ORDER BY surname;
+-----+-----+-----+
| name   | surname | group_num |
+-----+-----+-----+
| Анна   | Андреева | 103       |
| Кирилл | Борисов  | 201       |
| Игорь  | Васильев | 103       |
| Татьяна | Журавлева | 202       |
| Игорь  | Захаров  | 202       |
| Ольга  | Казакова | 102       |
| Татьяна | Климова  | 202       |
| Мария  | Никитина | 201       |
| Владимир | Романов | 104       |
+-----+-----+-----+
9 rows in set (0.00 sec)
mysql> _
```

Пример 10. Сортировка по нескольким столбцам.

```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> SELECT name, surname, group_num
-> FROM students
-> ORDER BY group_num, surname;
+-----+-----+-----+
| name   | surname | group_num |
+-----+-----+-----+
| Ольга  | Казакова | 102       |
| Анна   | Андреева | 103       |
| Игорь  | Васильев | 103       |
| Владимир | Романов | 104       |
| Кирилл | Борисов  | 201       |
| Мария  | Никитина | 201       |
| Татьяна | Журавлева | 202       |
| Игорь  | Захаров  | 202       |
| Татьяна | Климова  | 202       |
+-----+-----+-----+
9 rows in set (0.00 sec)
mysql>
```


По умолчанию данные сортируются в прямом порядке. Порядок можно изменить на обратный при помощи ключевого слова DESC. Для прямой сортировки используется ключевое слово ASC.

Пример 11. Сортировка в обратном порядке.



```
mysql> SELECT name, surname, group_num
-> FROM students
-> ORDER BY surname DESC;
```

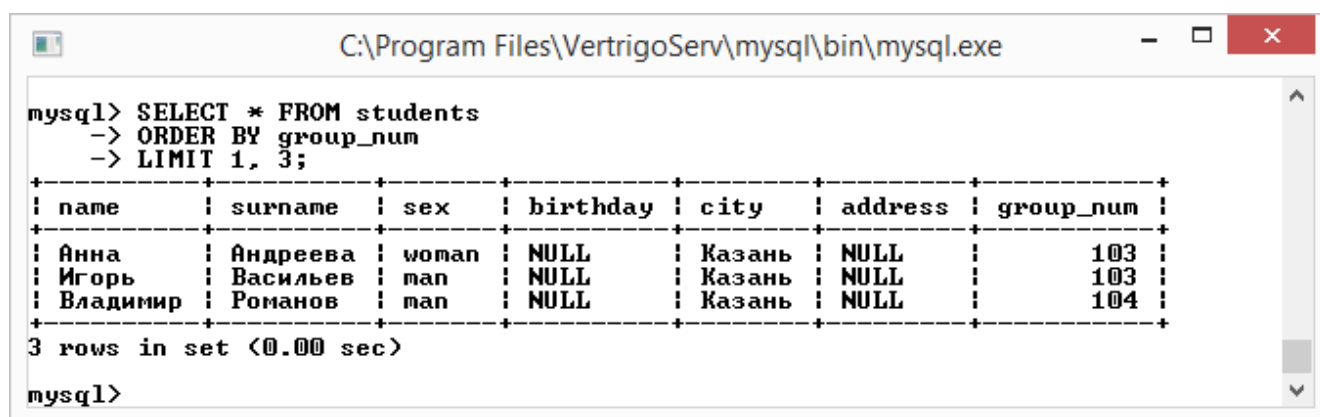
name	surname	group_num
Владимир	Романов	104
Мария	Никитина	201
Татьяна	Климова	202
Ольга	Казакова	102
Игорь	Захаров	202
Татьяна	Журавлева	202
Игорь	Васильев	103
Кирилл	Борисов	201
Анна	Андреева	103

```
9 rows in set (0.00 sec)
mysql>
```

2.4. Ограничение выборки

Для ограничения по количеству выводимых записей используется ключевое слово LIMIT start_index, count. На номер записи указывает первая цифра start_index, а вторая цифра count указывает на количество выводимых записей.

Пример 12. Ограничение выборки.



```
mysql> SELECT * FROM students
-> ORDER BY group_num
-> LIMIT 1, 3;
```

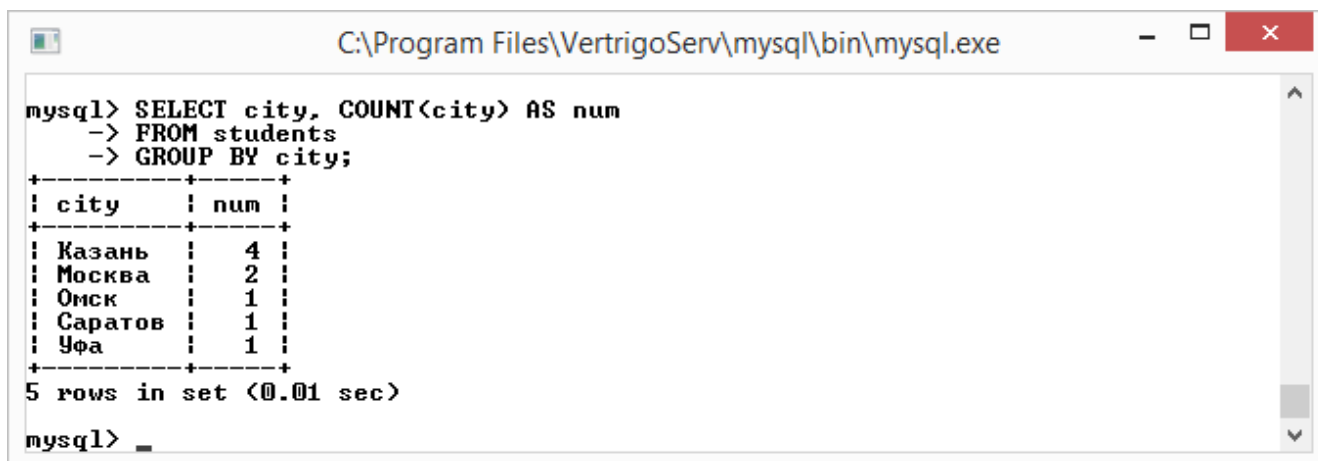
name	surname	sex	birthday	city	address	group_num
Анна	Андреева	woman	NULL	Казань	NULL	103
Игорь	Васильев	man	NULL	Казань	NULL	103
Владимир	Романов	man	NULL	Казань	NULL	104

```
3 rows in set (0.00 sec)
mysql>
```

2.5. Группировка записей

Для подсчета числа записей с одинаковыми полями используется конструкция GROUP BY совместно с функцией COUNT().

Пример 13. Совместное использование *GROUP BY* и *COUNT()*.



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

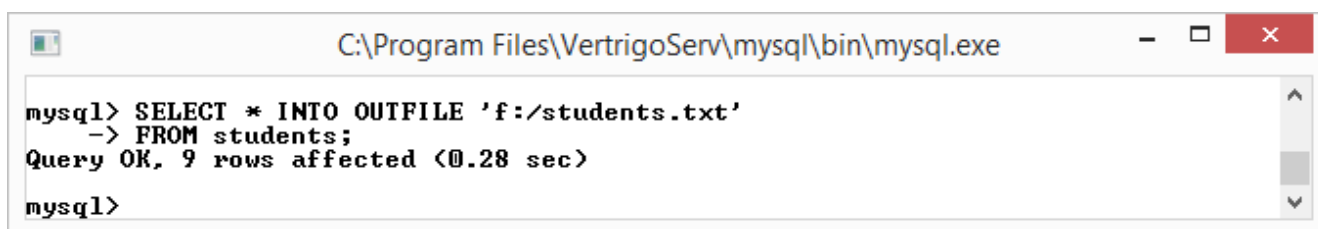
mysql> SELECT city, COUNT(city) AS num
-> FROM students
-> GROUP BY city;
+-----+-----+
| city   | num |
+-----+-----+
| Казань | 4   |
| Москва | 2   |
| Омск   | 1   |
| Саратов | 1   |
| Уфа    | 1   |
+-----+-----+
5 rows in set (0.01 sec)

mysql> _
```

В данном примере подсчитывается число студентов по каждому городу в отдельности.

2.6. Сохранение результатов выборки во внешний файл

Результат, получаемый при выполнении оператора *SELECT* можно сохранить в текстовый файл.



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe

mysql> SELECT * INTO OUTFILE 'f:/students.txt'
-> FROM students;
Query OK, 9 rows affected (0.28 sec)

mysql>
```

3. Задание на лабораторную работу

Изучить основные операторы выборки, сортировки, ограничения выборки, группировки записей и сохранения результатов выборки во внешний файл. Выполнить примеры методических рекомендаций для предметной области по вариантам.

Оформить отчет, содержащий подробное описание выполненной работы, скрины диалоговых окон (команд) и вывод по лабораторной работе.

4. Контрольные вопросы

1. Какой оператор используется для выборки записей из таблицы?
2. Для чего используются конструкции *BETWEEN* и *NOT BETWEEN*?
3. Опишите конструкции *IN* и *NOT IN*, в чем их разница?
4. Какие виды сортировок существуют?
5. С какой целью используется конструкция *GROUP BY* совместно с функцией *COUNT()*?
6. Как осуществить ограничение выборки по количеству выводимых записей?

Лабораторная работа №4

ТИПЫ ФУНКЦИЙ И ХРАНИМЫЕ ПРОЦЕДУРЫ

1. Цель работы

- Закрепление теоретических знаний понятий хранимые процедуры и типы функций.
- Приобретение практических навыков по работе с пользовательскими функциями и хранимыми процедурами.
- Создание отдельных функций и хранимых процедур для работы с БД предметной области.

2. Методика создания и использования хранимых процедур

2.1. Создание хранимой процедуры

Синтаксис команды:

```
CREATE PROCEDURE sp_name([param[, ...]])  
    routine_body
```

Здесь *sp_name* имя хранимой процедуры. За именем следует список необязательных параметров, заключенных в круглые скобки. Если параметров несколько, то они перечисляются через запятую. Круглые скобки обязательны, даже если список параметров пустой. Пробел между именем процедуры и открывающейся круглой скобкой недопустим.

param:
[IN, OUT, INOUT] param_name type

Ключевые слова *IN*, *OUT*, *INOUT* слова используются для задания направления передачи данных:

IN – данные передаются строго в хранимую процедуру. Если параметру с данным модификатором внутри процедуры присваивается новое значение, по выходу из нее не сохраняется и параметр принимает значение, которое он имел до вызова процедуры.

OUT – данные передаются строго из хранимой процедуры. По выходу из процедуры параметр с данным модификатором будет иметь значение, которое он примет внутри процедуры.

INOUT – значение этого параметра принимаются во внимание внутри процедуры, и если параметр изменится внутри, то параметр по выходу из процедуры примет измененное значение.

Если ни один из модификаторов не указан, считается, что параметр объявлен с модификатором *IN*.

type – после имени параметра указывается его тип.

routine_body – тело процедуры.

Основное неудобство заключено в использовании в конце запроса символа точки с запятой ';' воспринимаемое консольным клиентом как сигнал к выполнению запроса, т.е. отправке запроса на сервер. Обойти это можно переопределив символ разделителя запросов ';' на последовательность символов "//" с помощью оператора DELIMITER.

Пример 1. Переопределение символа разделителя запросов



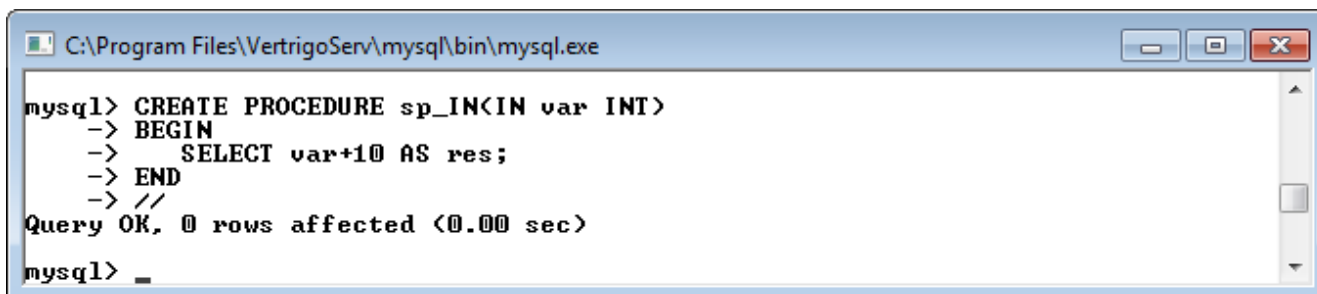
```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> DELIMITER //
mysql>
```

2.2. Тело процедуры

Тело процедуры состоит из составного оператора BEGIN ... END, внутри которого могут располагаться другие операторы, в том числе и другие операторы BEGIN ... END.

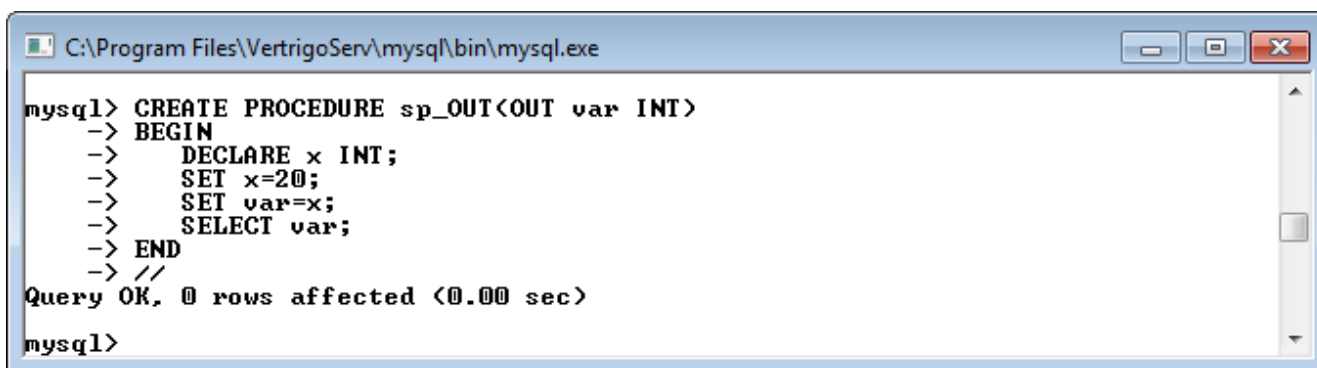
```
[label:] BEGIN
    Statements
END [label]
```

Пример 2. Параметр IN



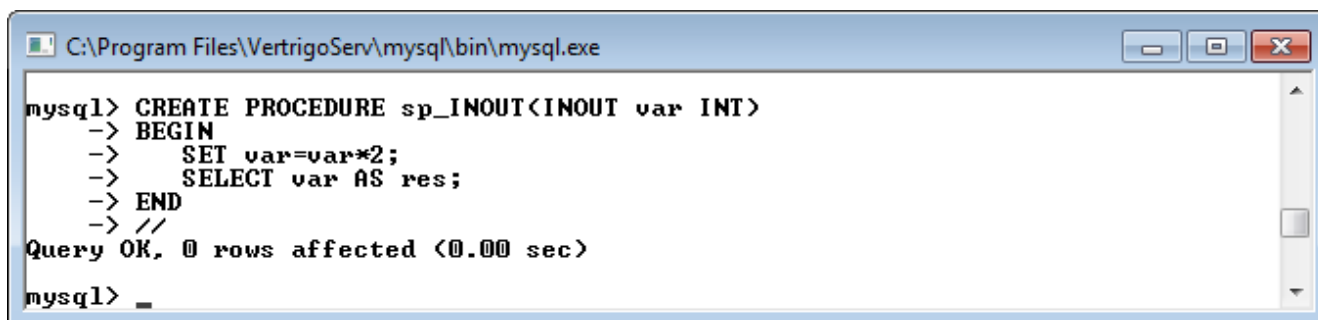
```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> CREATE PROCEDURE sp_IN(IN var INT)
-> BEGIN
->     SELECT var+10 AS res;
-> END
-> //
Query OK, 0 rows affected (0.00 sec)
mysql> _
```

Пример 3. Параметр OUT



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> CREATE PROCEDURE sp_OUT(OUT var INT)
-> BEGIN
->     DECLARE x INT;
->     SET x=20;
->     SET var=x;
->     SELECT var;
-> END
-> //
Query OK, 0 rows affected (0.00 sec)
mysql>
```

Пример 4. Параметр *INOUT*



```
C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> CREATE PROCEDURE sp_INOUT(INOUT var INT)
-> BEGIN
->     SET var=var*2;
->     SELECT var AS res;
-> END
-> //
Query OK, 0 rows affected (0.00 sec)
mysql> _
```

2.3. Переменные *SQL*

Существуют два вида переменных: пользовательские и хранимые.

Пользовательские переменные хранят значение до завершения соединения с сервером и доступны внутри хранимого кода и имеют совершенно определенный тип данных. Объявление переменных начинается с символа '@'.

Переменные хранимого кода напротив – имеют ограниченную область видимости (в пределах процедуры). Они должны объявляться в хранимом коде с ключевым словом `DECLARE` без символа '@'.

Синтаксис объявления пользовательской переменной:

```
SELECT @var_name;
```

или объявление пользовательской переменной с инициализацией (с присвоением значения):

```
SELECT @var_name:=value;
```

или с помощью оператора:

```
SET @var_name=value;
```

Синтаксис объявления хранимой переменной:

```
DECLARE var_name type DEFAULT def_value;
```

var_name – имя переменной, *value* – значение, *type* – тип переменной, *def_value* – значение по умолчанию.

Пример 5. Объявление и инициализация переменных

```

C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> SELECT @a;
+-----+
| @a    |
+-----+
| NULL  |
+-----+
1 row in set (0.00 sec)

mysql> SET @a=2; SELECT @a;
Query OK, 0 rows affected (0.00 sec)

+-----+
| @a    |
+-----+
| 2     |
+-----+
1 row in set (0.00 sec)

mysql> SELECT @b:=5;
+-----+
| @b:=5 |
+-----+
| 5     |
+-----+
1 row in set (0.00 sec)

mysql>

```

2.4. Вызов хранимой процедуры

Синтаксис команды:

```
CALL sp_name([param[, ...]]);
```

Пробел между именем процедуры и открывающейся круглой скобкой недопустим.

```

C:\Program Files\VertrigoServ\mysql\bin\mysql.exe
mysql> DELIMITER ;
mysql> SET @v=2;
Query OK, 0 rows affected (0.00 sec)

mysql> CALL sp_IN(@v);
+-----+
| res  |
+-----+
| 12   |
+-----+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> CALL sp_OUT(@v);
+-----+
| res  |
+-----+
| 15   |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql> CALL sp_INOUT(@v);
+-----+
| res  |
+-----+
| 30   |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql>

```

2.5. Удаление хранимой процедуры

Синтаксис команды:

```
DROP PROCEDURE [IF EXISTS] sp_name;
```

sp_name – имя удаляемой процедуры. Ключевое слово IF EXISTS предотвращает ошибку возникновения, если процедуры не существует.

3. Задание на лабораторную работу

Научиться создавать, вызывать и удалять хранимые процедуры, используя примеры методических рекомендаций и наработки предыдущих лабораторных работ по вариантам.

Оформить отчет, содержащий подробное описание выполненной работы, скрины диалоговых окон (команд) и вывод по лабораторной работе.

4. Контрольные вопросы

1. Дайте определение понятию Хранимые процедуры.
2. Какие типы функций поддерживает *MySQL*?
3. Какие существуют классы пользовательских функций?
4. Какие существуют ограничения на пользовательские функции?
5. Переменные, каких типов данных нельзя передавать в пользовательскую функцию?
6. Какие операторы используются для создания и вызова хранимых процедур?
7. При помощи, какой команды можно удалить пользовательскую функцию?

Список литературы

1. Основы проектирования баз данных: Учебное пособие / Голицына О.Л., Партыка Т.Л., Попов И.И., – 2-е изд. – М.: Форум, НИЦ ИНФРА-М, 2016. – 416 с.: 60x90 1/16. – (Профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-91134-655-3 <http://znanium.com/bookread2.php?book=552969>.
2. Базы данных: в 2 кн. Книга 2. Распределенные и удаленные базы данных: учебник / В.П. Агальцов. – М.: ИД «ФОРУМ»: ИНФРА-М, 2018. – 271 с. – (Высшее образование: Бакалавриат). <http://znanium.com/bookread2.php?book=929256>.
3. Базы данных: Учебник / Шустова Л.И., Тараканов О.В. – М.: НИЦ ИНФРА-М, 2016. – 304 с.: 60x90 1/16. – (Высшее образование: Бакалавриат) (Переплёт) ISBN 978-5-16-010485-0, 500 экз. <http://znanium.com/bookread2.php?book=491069>.
4. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем: учеб. пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. – М.: ИД «ФОРУМ»: ИНФРА-М, 2018. – 368 с. – (Высшее образование: Бакалавриат). <http://znanium.com/bookread2.php?book=926871>.
5. SQL – язык реляционных баз данных: Учебное пособие / Кара-Ушанов В.Ю., - 2-е изд., стер. – М.:Флинта, Изд-во Урал. ун-та, 2017. – 156 с. ISBN 978-5-9765-3120-8 <http://znanium.com/bookread2.php?book=947669>.
6. Microsoft SQL Server 2012: Пособие / Бондарь А.Г. – СПб: БХВ-Петербург, 2013. – 608 с. ISBN 978-5-9775-0501-7 <http://znanium.com/bookread2.php?book=941206>.
7. Яргер Р.Дж. MySQL и mSQL: Базы данных для небольших предприятий и Интернета / Р.Дж. Яргер, Дж. Риз, Т. Кинг. – М.: СПб: Символ-Плюс, 2014. – 560 с.

Варианты заданий для выполнения лабораторных работ

Вариант	Предметная область
1.	Химчистка
2.	Фирма по разработке ПО
3.	Охранное агентство
4.	Типография
5.	Маршруты городского транспорта
6.	Агентство недвижимости
7.	Университет
8.	Больница
9.	Дистрибьюторская фирма
10.	Аптечная сеть
11.	Автосервис
12.	Мелкое производство
13.	Гостиница
14.	Туристическая фирма
15.	Складское хозяйство
16.	Супермаркет
17.	Сервисный центр
18.	Редакция газеты
19.	Служба такси
20.	Депо электротранспорта
21.	Ресторан
22.	Жилищно-эксплуатационный участок (ЖЭУ)
23.	Аренда офисных помещений
24.	Продажа билетов
25.	Фотосалон

Примеры:

Химчистка

Таблица "Информация о заказе"

Номер заказа	ФИО заказчика	Заказ	Сумма заказа	Ответственный
--------------	---------------	-------	--------------	---------------

Таблица "Клиенты"

Дата приема заказа	Адрес клиента	Состояние заказа	Отдел	Номер заказа
--------------------	---------------	------------------	-------	--------------

Таблица "Рабочий персонал"

Отдел	Кол-во работников	Менеджер отдела	Вид деятельности	Примечание
-------	-------------------	-----------------	------------------	------------

Фирма по разработке ПО

Таблица "Информация о заказе"

Номер заказа	ФИО заказчика	Заказ	Сумма заказа	Исполнитель
--------------	---------------	-------	--------------	-------------

Таблица "Исполнители"

Отдел	ФИО исполнителя	Вид деятельности	Раб телефон	Стаж
-------	-----------------	------------------	-------------	------

Таблица «Клиенты»

Дата получения заказа	Адрес Клиента	Состав заказа	Отдел	№ заказа
-----------------------	---------------	---------------	-------	----------

Охранное агентство

Таблица "Группа быстрого реагирования"

Код	Название группы	Количество человек	Руководитель
-----	-----------------	--------------------	--------------

Таблица "Охраняемые объекты"

Код	Имя	Код ГБР	Заказчик
-----	-----	---------	----------

Таблица "Сотрудники"

Код	ФИО	Код группы	День рождения
-----	-----	------------	---------------

Типография

Таблица "Сотрудники"

Идентификатор	Имя	Фамилия	Возраст
---------------	-----	---------	---------

Таблица "Издания"

Идентификатор	Название	Тираж	Количество страниц
---------------	----------	-------	--------------------

Таблица "Отделы"

Идентификатор	Наименование отдела	Количество сотрудников	Направление
---------------	---------------------	------------------------	-------------

Маршруты городского транспорта

Таблица "Депо"

ИД депо	Название депо	Адрес депо	Телефон депо	Количество транспорта
---------	---------------	------------	--------------	-----------------------

Таблица "Водители"

Табельный номер водителя	ФИО водителя	Возраст водителя	Телефон водителя	Адрес водителя
--------------------------	--------------	------------------	------------------	----------------

Таблица "Электротранспорт"

ИД депо	Номер	Номер маршрута	Вид транспорта	Номер водителя
---------	-------	----------------	----------------	----------------

Агентство недвижимости

Таблица "Недвижимость"

Идентификатор	Адрес недвижимости	Площадь офиса	Тип недвижимости	Кол-во комнат
---------------	--------------------	---------------	------------------	---------------

Таблица "Риэлторы"

Идентификатор	Имя	Фамилия	Дата рождения	Заметки
---------------	-----	---------	---------------	---------

Таблица "Покупатели"

Идентификатор	Имя	Фамилия	Деньги	Заметки
---------------	-----	---------	--------	---------

Университет

Таблица "Преподаватели"

Имя преподавателя	Фамилия преподавателя	Название предмета	Номер телефона
-------------------	-----------------------	-------------------	----------------

Таблица "Дисциплины"

Название предмета	Группа	Номер аудитории	Имя преподавателя
-------------------	--------	-----------------	-------------------

Таблица "Студенты"

Имя	Фамилия	Группа	Имя преподавателя
-----	---------	--------	-------------------

Больница

Таблица "Медперсонал"

Идентификатор	Фамилия сотрудника	Должность	Отделение
---------------	--------------------	-----------	-----------

Таблица "Отделения"

Номер	Наименование отделения	Количество сотрудников	Оборудование
-------	------------------------	------------------------	--------------

Таблица "Пациенты"

Номер пациента	Фамилия пациента	Возраст	Лечащий врач	Заболевание
----------------	------------------	---------	--------------	-------------

Дистрибьюторская фирма

Таблица "Производитель"

Идентификатор	Наименование	Адрес	Телефон
---------------	--------------	-------	---------

Таблица "Тип товара"

Идентификатор	Наименование	Класс	Описание
---------------	--------------	-------	----------

Таблица "Товар"

Идентификатор	Тип товара	Наименование	Производитель	Количество
---------------	------------	--------------	---------------	------------

Аптечная сеть

Таблица "Аптеки"

Идентификатор	Номер/название аптеки	ФИО заведующего	Телефон	Адрес
---------------	-----------------------	-----------------	---------	-------

Таблица "Сотрудники"

Идентификатор	ФИО сотрудника	Возраст	Образование	Должность
---------------	----------------	---------	-------------	-----------

Таблица "Медикаменты"

Идентификатор товара	Наименование товара	Производитель	Поставщик	Цена
----------------------	---------------------	---------------	-----------	------

Автосервис

Таблица "Клиентская база"

ID клиента	ФИО клиента	Номер паспорта	Адрес	Номер телефона
------------	-------------	----------------	-------	----------------

Таблица "Рабочий персонал"

ID работника	ФИО работника	Должность	Стаж	Жалобы по работе
--------------	---------------	-----------	------	------------------

Таблица "Вид ремонта"

ID ремонта	ID клиента	Модель машины	Вид ремонта	Дата сдачи в ремонт	Дата исполнения заказа
------------	------------	---------------	-------------	---------------------	------------------------

Мелкое производство

Таблица "Товар"

Номер товара	Наименование	Тираж	Производитель
--------------	--------------	-------	---------------

Таблица "Оборудование"

Номер оборудования	Отдел	Стоимость о оборудования	Производитель оборудования
--------------------	-------	--------------------------	----------------------------

Таблица "Сотрудники"

ID сотрудника	Должность	Отдел	Стаж работы
---------------	-----------	-------	-------------

Гостиница

Таблица "Клиент"

Номер регистрации	ФИО	Номер документа	Номер телефона
-------------------	-----	-----------------	----------------

Таблица "Регистратура"

Номер регистрации	№ номера	Число заезда	Число выезда	Стоимость
-------------------	----------	--------------	--------------	-----------

Таблица "Номера"

№ номера	Категория	Кол-во комнат	Стоимость	Этаж
----------	-----------	---------------	-----------	------

Туристическая фирма

Таблица "Отели"

Идентификатор	Страна	Класс отдыха	Стоимость
---------------	--------	--------------	-----------

Таблица "Клиенты"

Идентификатор	ФИО клиента	Категория	Телефон
---------------	-------------	-----------	---------

Таблица "Горящие туры"

Страна	Класс отдыха	Стоимость	Дата
--------	--------------	-----------	------