

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ ПРАКТИЧЕСКОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «СИСТЕМЫ ОБРАБОТКИ И ХРАНЕНИЯ ДАННЫХ»

Создать базу данных и таблицы в ней по выбранной теме, на основе разработанных моделей.

Результат работы в виде отчета должен содержать:

- снимки экрана (скриншоты) процесса разработки;
- снимок экрана завершенной базы данных;
- снимки экрана (скриншоты) списка созданных таблиц;
- снимки экрана (скриншоты) структуры созданных таблиц;
- снимки экрана (скриншоты) содержания созданных таблиц;
- итоговый скрипт БД.

Теоретические основы.

На данном этапе нам гораздо важнее разобраться с созданием запросов, нежели сразу усложнять себе жизнь созданием таблиц, загрузкой в них данных и извлечением их оттуда. В этом пункте разъясняются основные принципы ввода команд; на примере нескольких запросов вы можете поближе познакомиться с работой `mysql`.

Ниже приведена простая команда, запрашивающая у сервера информацию об его версии и текущей дате.

```
mysql> SELECT VERSION(), CURRENT_DATE;
```

```
+-----+-----+
| VERSION() | CURRENT_DATE |
+-----+-----+
| 3.22.20a-log | 1999-03-19 |
+-----+-----+
1 row in set (0.01 sec)
mysql>
```

Этот запрос иллюстрирует следующие особенности `mysql`:

- команда обычно состоит из SQL-выражения, за которым следует точка с запятой. (Из этого правила есть и исключения - команды без точки с запятой. Одним из них является упомянутая выше команда `QUIT`, остальные мы рассмотрим позднее.)

- когда пользователь вводит команду, `mysql` отправляет ее серверу для выполнения и выводит на экран сначала результаты, а затем - новую строку `mysql>`, что означает готовность к выполнению новых команд.

- `mysql` выводит результаты работы запроса в виде таблицы (строк и столбцов). В первой строке этой таблицы содержатся заголовки столбцов, а в следующих строках - собственно результаты. Обычно заголовками столбцов становятся имена, полученные из таблиц базы. Если же извлекается не столбец таблицы, а значение выражения (как это происходит в приведенном выше примере), `mysql` дает столбцу имя запрашиваемого выражения.

• mysql сообщает количество возвращаемых строк и время выполнения запроса, что позволяет в некоторой степени составить представление о производительности сервера. Эти значения обычно весьма впечатляют, так как представляют обычное (а не машинное время), кроме того, на них оказывает влияние загрузка сервера и скорость работы сети.

Для ввода ключевых слов можно использовать любой регистр символов.

Приведенные ниже запросы абсолютно идентичны:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

А это - еще один запрос. В нем демонстрируется использование mysql в качестве несложного калькулятора:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+-----+-----+
| SIN(PI()/4) | (4+1)*5 |
+-----+-----+
| 0.707107 | 25 |
+-----+-----+
```

Все команды, представленные выше, были относительно короткими и состояли из одной строки. В одну строку можно поместить и несколько команд. Но каждая из них должна заканчиваться точкой с запятой:

```
mysql> SELECT VERSION(); SELECT NOW();
+-----+
| VERSION() |
+-----+
| 3.22.20a-log |
+-----+

+-----+
| NOW() |
+-----+
| 1999-03-19 00:15:33 |
+-----+
```

Втискивать все команды в одну строку совсем не обязательно, так что создание длинных команд, занимающих несколько строк, никаких проблем не вызывает. Для mysql признаком завершения выражения является точка с запятой, а не конец строки (другими словами, mysql принимает команды без форматирования: строки с командами собираются, но не исполняются до тех пор, пока программа не обнаружит точку с запятой).

Вот пример несложного выражения, занимающего несколько строк:

```
mysql> SELECT
-> USER()
-> ,
-> CURRENT_DATE;
+-----+-----+
| USER() | CURRENT_DATE |
+-----+-----+
| joesmith@localhost | 1999-03-18 |
+-----+-----+
```

Обратите внимание на то, как изменилась метка командной строки (с `mysql>` на `->`) после ввода первой строки этого запроса. Таким образом программа `mysql` показывает, что завершено выражения она пока что не получила и ожидает его полного ввода. Эта метка очень полезна, так как предоставляет весьма ценную информацию о состоянии программы. С ее помощью всегда можно узнать, чего ждет `mysql`.

Если вы решите отменить исполнение набираемой команды, наберите `\c`:

```
mysql> SELECT
-> USER()
-> \c
mysql>
```

Обратите внимание на метку: после ввода команды `\c` она снова принимает вид `mysql>`, показывая, что программа `mysql` перешла в режим ожидания указаний.

Ниже приведены все возможные варианты вида метки командной строки и соответствующие им состояния `mysql`:

Метка	Значение
<code>mysql></code>	Ожидание новой команды.
<code>-></code>	Ожидание следующей строки многострочной команды.
<code>'></code>	Ожидание следующей строки, сбор строкового выражения, начинающегося с одиночной кавычки (<code>`'</code>).
<code>"></code>	Ожидание следующей строки, сбор строкового выражения, начинающегося с двойной кавычки (<code>`"</code>).

Обычно многострочные команды получаются случайно, когда хочешь создать обычную команду, но забываешь поставить завершающую точку с запятой. В таком случае `mysql` ожидает продолжения:

```
mysql> SELECT USER()
->
```

Если с вами произошло подобное (завершили команду, но программа выдает только метку `->`), то `mysql`, вероятнее всего, ждет точки с запятой. Не обратив внимание на метку командной строки, можно довольно долго ждать выполнения команды, не понимая в чем дело. А достаточно лишь поставить точку с запятой, завершив команду, которую `mysql` и выполнит:

```
mysql> SELECT USER()
-> ;
+-----+
| USER() |
+-----+
| joesmith@localhost |
+-----+
```

Метки `'>` и `">` используются при сборе строк. В MySQL строки можно заключать как в одинарные (``'`), так и в двойные (``"`) кавычки (можно, например, написать `'hello'` или `"goodbye"`), к тому же, `mysql` позволяет вводить строковые выражения, состоящие из нескольких строчек текста. Метка `'>` или `">` обозначает, что вы ввели строку, открывающуюся символом

кавычек ``'`` или ``"```, но еще не ввели завершающую строковое выражение закрывающую кавычку.

Это нормально, если вы собираетесь создать большое строковое выражение из нескольких строчек. Но это не слишком частый случай. Гораздо чаще оказывается, что вы просто забыли поставить закрывающую кавычку. Например:

```
mysql> SELECT * FROM my_table WHERE name = "Smith AND age <
30;
">
```

Если ввести такую команду `SELECT`, нажать `Enter` и подождать результатов, ничего не произойдет. Тут-то и нужно обратить внимание на метку командной строки, выглядящую вот так: `">`. Это значит, что `mysql` ждет ввода завершающей части строки. (Теперь заметили ошибку в команде? В строке `"Smith` нет закрывающей кавычки.)

Что делать в этом случае? Проще всего было бы отменить команду. Однако теперь просто набрать `\c` нельзя, так как `mysql` примет эти символы за часть собираемой строки! Вместо этого нужно ввести закрывающие кавычки (тем самым дав `mysql` понять, что строка закончилась) и лишь затем набрать `\c`:

```
mysql> SELECT * FROM my_table WHERE name = "Smith AND age <
30;
"> "\c
mysql>
```

Метка командной строки снова примет вид `mysql>`, показывая готовность `mysql` к выполнению команд.

Знать значение меток `'>` и `">` необходимо, так как при вводе незавершенной строки все последующие строки будут игнорироваться `mysql` - включая строку с командой `QUIT`! Это может основательно сбить с толку, особенно если не знать, что для отмены команды перед соответствующей последовательностью символов необходимо поставить закрывающую кавычку.

Практическое занятие по созданию БД на языке SQL

Утилита `mysql` (иногда называемая также «терминальным монитором» или просто «монитором») представляет собой интерактивную программу, позволяющую подсоединяться к MySQL-серверу, запускать запросы, и просматривать результаты. Программа `mysql` может работать и в пакетном режиме: для этого необходимо записать все запросы в файл, а затем передать его содержимое на исполнение `mysql`. Ниже описаны оба способа использования `mysql`.

Увидеть список команд программы `mysql` можно, запустив ее с параметром `-help`:

```
shell> mysql --help
```

На Вашем компьютере необходимо установить `mysql` и осуществить связь с сервером MySQL.

Подсоединение к серверу и отсоединение от него

Для этого прежде всего надо запустить сервер MySQL. Идем в системное меню Пуск — Программы — MySQL — MySQL Server 5.1 — MySQL Command Line Client.

При подключении к серверу с помощью `mysql` обычно нужно ввести имя пользователя MySQL и, в большинстве случаев, пароль. Если сервер запущен не на том компьютере, с которого вы вошли в систему, необходимо также указать имя хоста. Параметры соединения (а именно - соответствующее имя хоста, пользователя и пароль) вы сможете узнать у администратора. Получив соответствующие параметры, подсоединиться к серверу можно следующим образом:

```
shell> mysql -h host -u user -p
Enter password: *****
```

Символы `*****` обозначают ваш пароль; введите его, когда `mysql` выведет на экран запрос `Enter password:`.

Если все сработает, на экране должна появиться следующая информация и метка командной строки `mysql>`:

```
shell> mysql -h host -u user -p
Enter password: *****

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 459 to server version: 3.22.20a-
log

Type 'help' for help.

mysql>
```

Метка обозначает, что программа `mysql` готова к вводу команд.

В некоторых вариантах установки MySQL возможно подсоединение к запущенному на локальном хосте серверу без ввода имени пользователя (пользователь `anonymous`). Если ваша система настроена именно так, подсоединиться к серверу вы сможете, запустив `mysql` со следующими параметрами:

```
shell> mysql
```

После установки соединения можно в любой момент отключиться от сервера, набрав в командной строке `mysql>` команду `QUIT`:

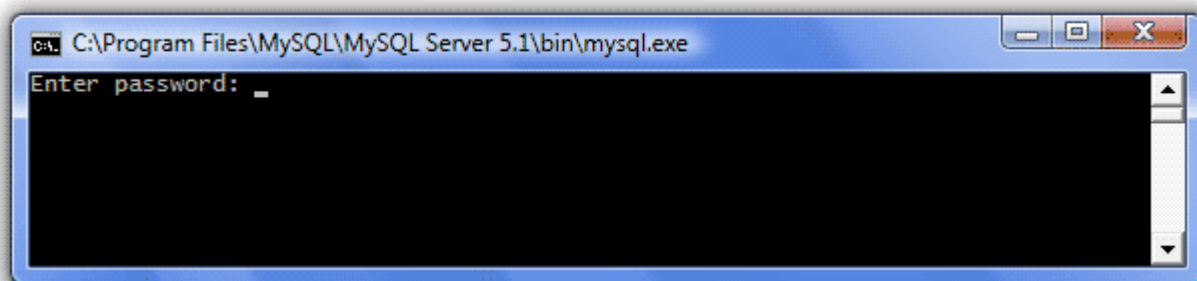
```
mysql> QUIT
Bye
```

Отсоединиться от сервера можно и при помощи сочетания клавиш `Control-D`.

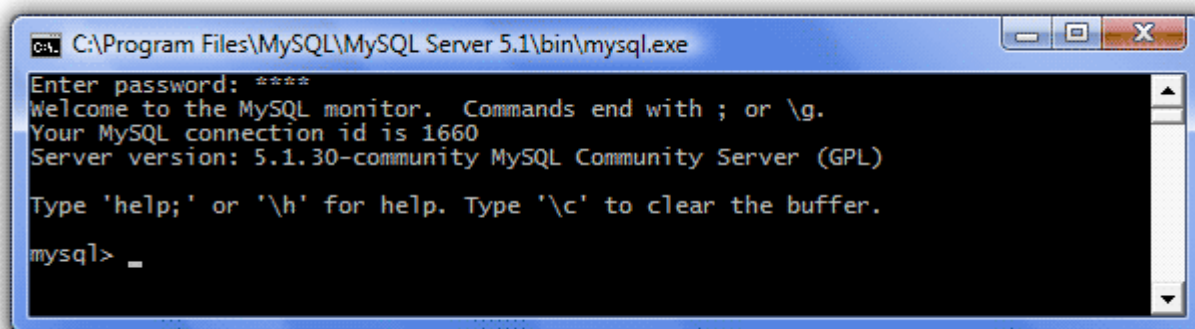
Большая часть приведенных ниже примеров построена с учетом того, что соединение с сервером уже установлено. Это видно по наличию в них командной строки `mysql>`.

Создание базы данных и таблиц

Для этого прежде всего надо запустить сервер MySQL. Идем в системное меню Пуск — Программы — MySQL — MySQL Server 5.1 — MySQL Command Line Client. Откроется окно, предлагающее ввести пароль.



Нажимаем `Enter` на клавиатуре, если вы не указывали пароль при настройке сервера или указываем пароль, если вы его задавали. Ждем приглашения `mysql>`.



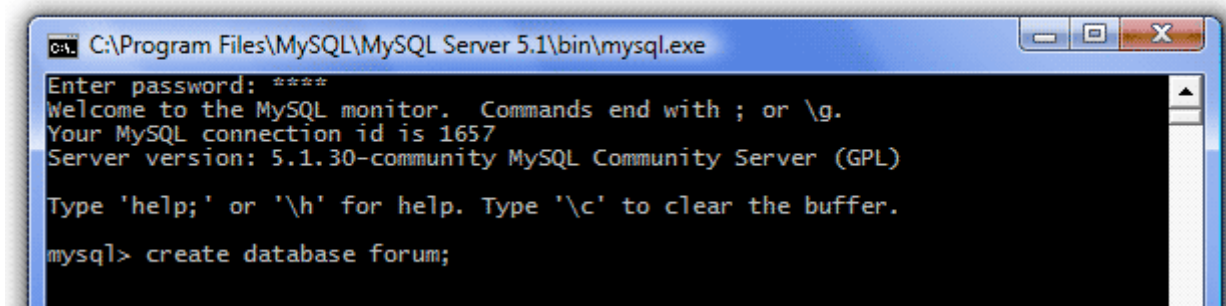
Нам надо создать базу данных, которую назовем `forum`. Для этого в SQL существует оператор `create database`. Создание базы данных имеет следующий синтаксис:

```
create database имя_базы_данных;
```

Максимальная длина имени БД составляет 64 знака и может включать буквы, цифры, символ «`_`» и символ «`$`». Имя может начинаться с цифры, но не должно полностью состоять из цифр. Любой запрос к БД заканчивается

точкой с запятой (этот символ называется разделителем — delimiter). Получив запрос, сервер выполняет его и в случае успеха выдает сообщение «Query OK ...».

Итак, создадим БД forum:

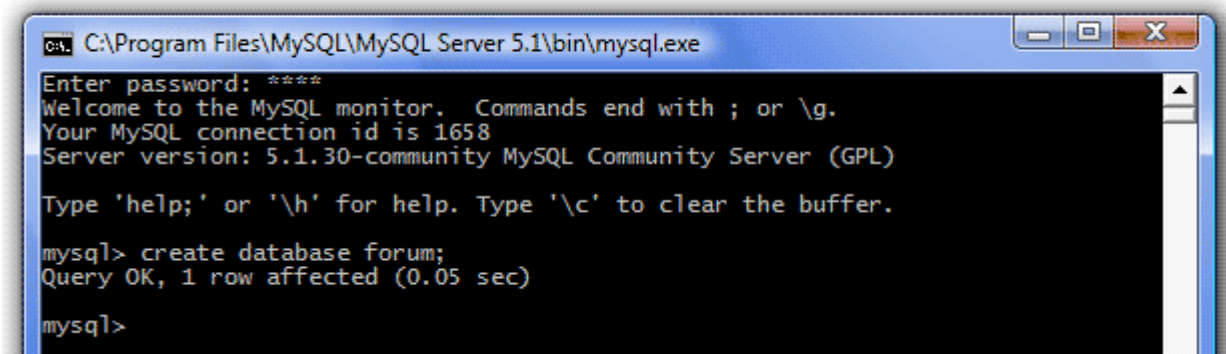


```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1657
Server version: 5.1.30-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database forum;
```

Нажимаем Enter и видим ответ «Query OK ...», означающий, что БД была создана:



```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1658
Server version: 5.1.30-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

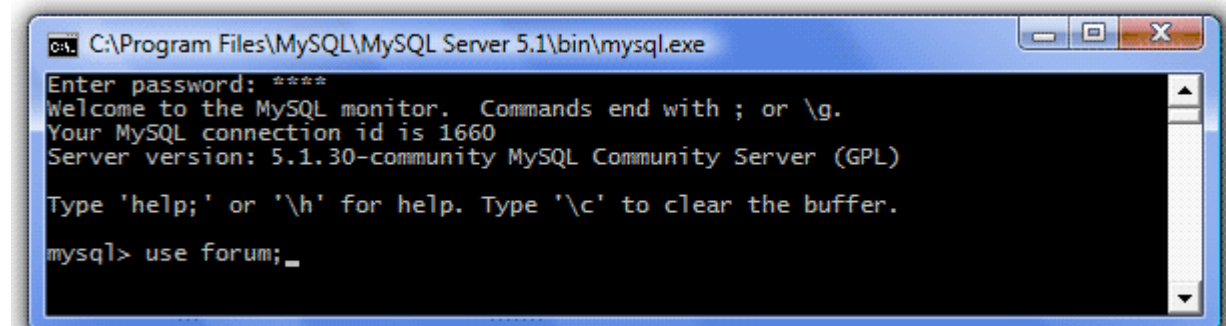
mysql> create database forum;
Query OK, 1 row affected (0.05 sec)

mysql>
```

Вот так все просто. Теперь в этой базе данных нам надо создать 3 таблицы: темы, пользователи и сообщения. Но перед тем, как это делать, нам надо указать серверу в какую именно БД мы создаем таблицы, т.е. надо выбрать БД для работы. Для этого используется оператор `use`. Синтаксис выбора БД для работы следующий:

```
use имя_базы_данных;
```

Итак, выберем для работы нашу БД forum:

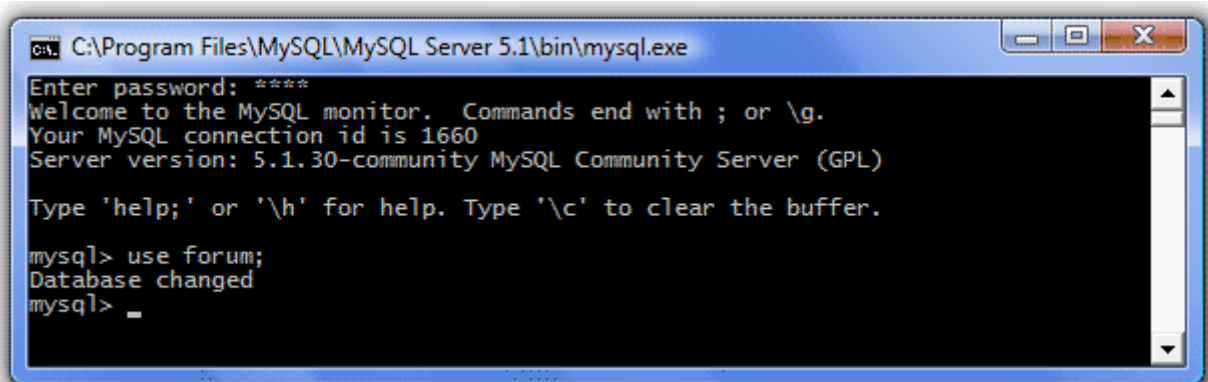


```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1660
Server version: 5.1.30-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use forum;
```

Нажимаем Enter и видим ответ «Database changed» — база данных выбрана.



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1660
Server version: 5.1.30-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

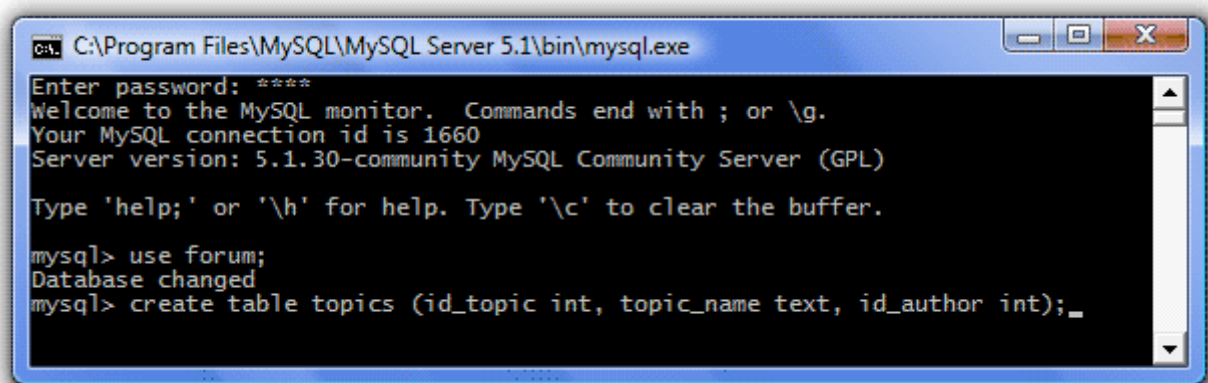
mysql> use forum;
Database changed
mysql> _
```

Выбирать БД необходимо в каждом сеансе работы с MySQL.

Для создания таблиц в SQL существует оператор `create table`. Создание базы данных имеет следующий синтаксис:

```
create table имя_таблицы (имя_первого_столбца тип,
имя_второго_столбца тип, ..., имя_последнего_столбца тип );
```

Требования к именам таблиц и столбцов такие же, как и для имен БД. К каждому столбцу привязан определенный тип данных, который ограничивает характер информации, которую можно хранить в столбце (например, предотвращает ввод букв в числовое поле). MySQL поддерживает несколько типов данных: числовые, строковые, календарные и специальный тип `NULL`, обозначающий отсутствие информации. Подробно о типах данных мы будем говорить в следующем уроке, а пока вернемся к нашим таблицам. В них у нас всего два типа данных — целочисленные значения (`int`) и строки (`text`). Итак, создадим первую таблицу — Темы:

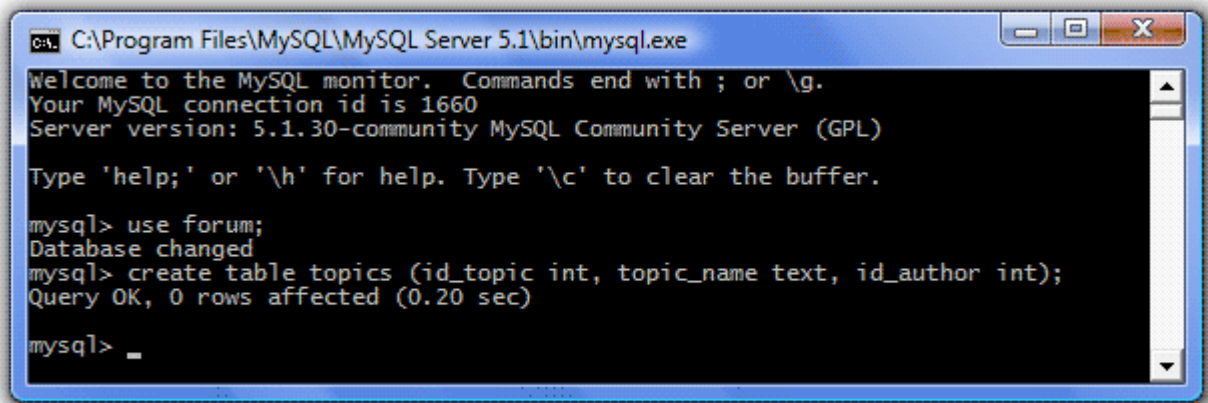


```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1660
Server version: 5.1.30-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use forum;
Database changed
mysql> create table topics (id_topic int, topic_name text, id_author int);_
```

Нажимаем Enter — таблица создана:



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1660
Server version: 5.1.30-community MySQL Community Server (GPL)

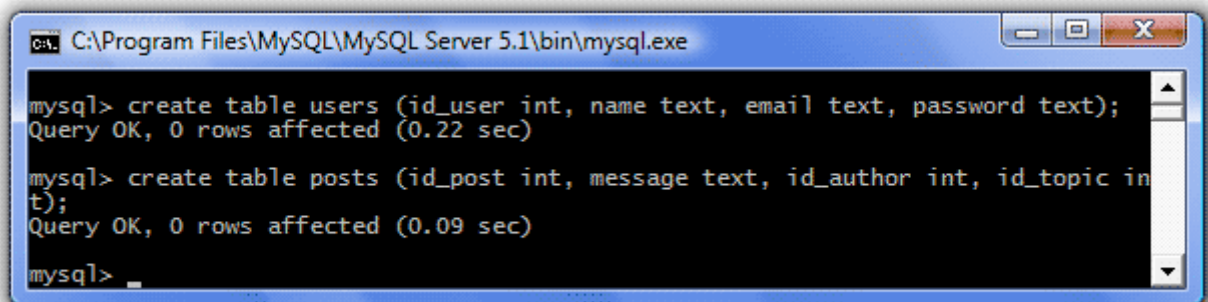
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use forum;
Database changed
mysql> create table topics (id_topic int, topic_name text, id_author int);
Query OK, 0 rows affected (0.20 sec)

mysql> _
```

Итак, мы создали таблицу `topics` (темы) с тремя столбцами:
`id_topic int` — id темы (целочисленное значение),
`topic_name text` — имя темы (строка),
`id_author int` — id автора (целочисленное значение).

Аналогичным образом создадим оставшиеся две таблицы — `users` (пользователи) и `posts` (сообщения):



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

mysql> create table users (id_user int, name text, email text, password text);
Query OK, 0 rows affected (0.22 sec)

mysql> create table posts (id_post int, message text, id_author int, id_topic int);
Query OK, 0 rows affected (0.09 sec)

mysql> _
```

Итак, мы создали БД `forum` и в ней три таблицы. Сейчас мы об этом помним, но если наша БД будет очень большой, то удержать в голове названия всех таблиц и столбцов просто невозможно. Поэтому надо иметь возможность посмотреть, какие БД у нас существуют, какие таблицы в них присутствуют, и какие столбцы эти таблицы содержат. Для этого в SQL существует несколько операторов:

- `show databases` — показать все имеющиеся БД,
- `show tables` — показать список таблиц текущей БД (предварительно ее надо выбрать с помощью оператора `use`),
- `describe имя_таблицы` — показать описание столбцов указанной таблицы.

Давайте попробуем. Смотрим все имеющиеся базы данных (у вас она пока одна — `forum`, у меня 30, и все они перечислены в столбик):

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| access    |
| astro     |
| astrologiy |
| balance   |
| beverly   |
| blog      |
| dsd       |
| forum     |
| forum1    |
| forumsitedo |
| game      |
| genskoe_video |
| graph     |
| hotel     |
| kopilka   |
| krasota   |
| mysql     |
| pay       |
| payment   |
| proba     |
| proba1    |
| sample    |
| setka     |
| shashki   |
| soveti    |
| svar      |
| svarka    |
| test      |
| user_set  |
+-----+
30 rows in set (0.31 sec)

mysql>
```

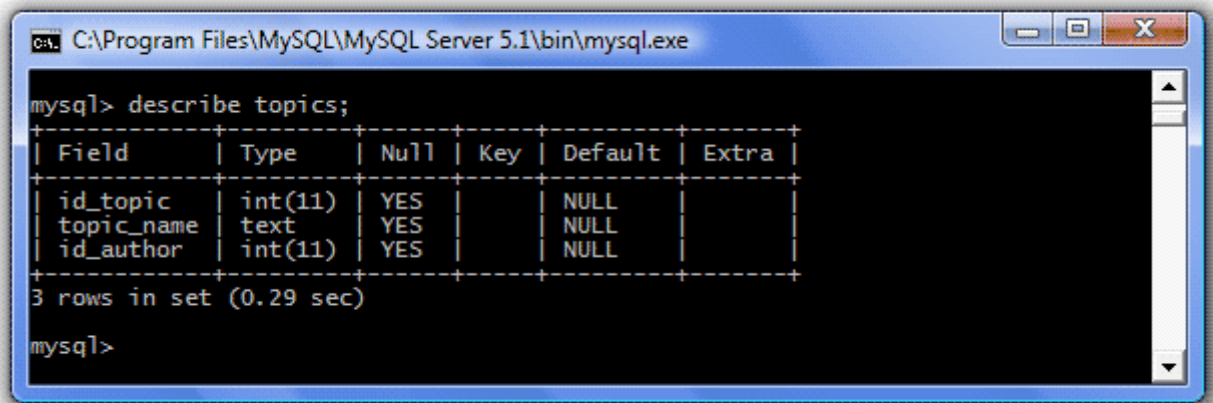
Теперь посмотрим список таблиц БД forum (для этого ее предварительно надо выбрать), не забываем после каждого запроса нажимать Enter:

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

mysql> use forum;
Database changed
mysql> show tables;
+-----+
| Tables_in_forum |
+-----+
| posts           |
| topics          |
| users           |
+-----+
3 rows in set (0.18 sec)

mysql> _
```

В ответе видим названия наших трех таблиц. Теперь посмотрим описание столбцов, например, таблицы topics:



```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe

mysql> describe topics;
+-----+-----+-----+-----+-----+-----+
| Field      | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_topic   | int(11)| YES  |     | NULL    |       |
| topic_name | text   | YES  |     | NULL    |       |
| id_author  | int(11)| YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.29 sec)

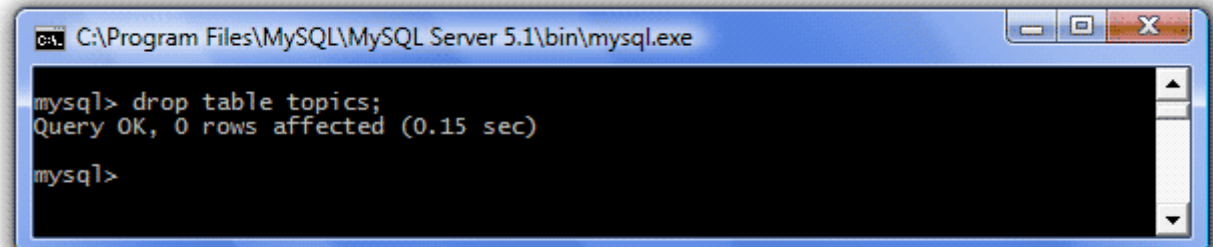
mysql>
```

Первые два столбца нам знакомы — это имя и тип данных, значения остальных нам еще предстоит узнать. Но прежде мы все-таки узнаем какие типы данных бывают, какие и когда следует использовать.

А сегодня мы рассмотрим последний оператор — `drop`, он позволяет удалять таблицы и БД. Например, давайте удалим таблицу `topics`. Так как мы два шага назад выбирали БД `forum` для работы, то сейчас ее выбирать не надо, можно просто написать:

```
drop table имя_таблицы;
```

и нажать `Enter`.

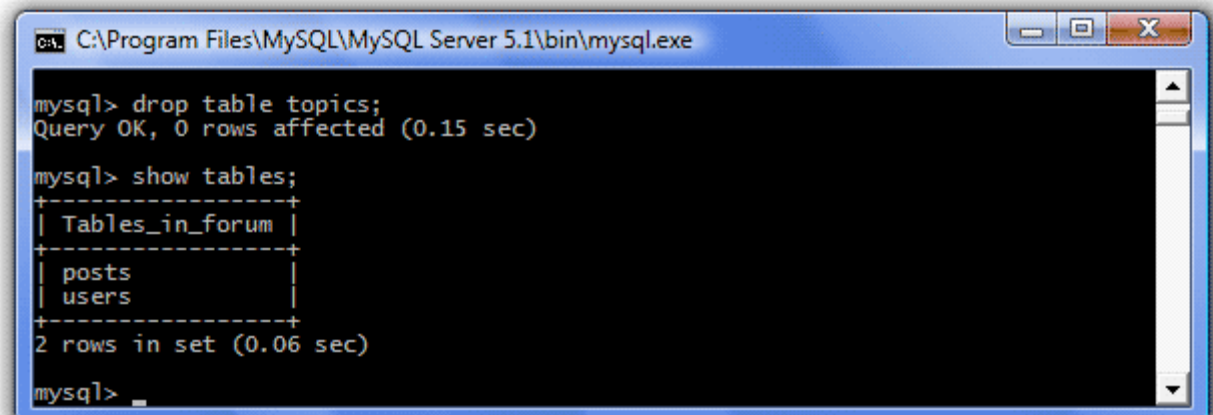


```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe

mysql> drop table topics;
Query OK, 0 rows affected (0.15 sec)

mysql>
```

Теперь снова посмотрим список таблиц нашей БД:



```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe

mysql> drop table topics;
Query OK, 0 rows affected (0.15 sec)

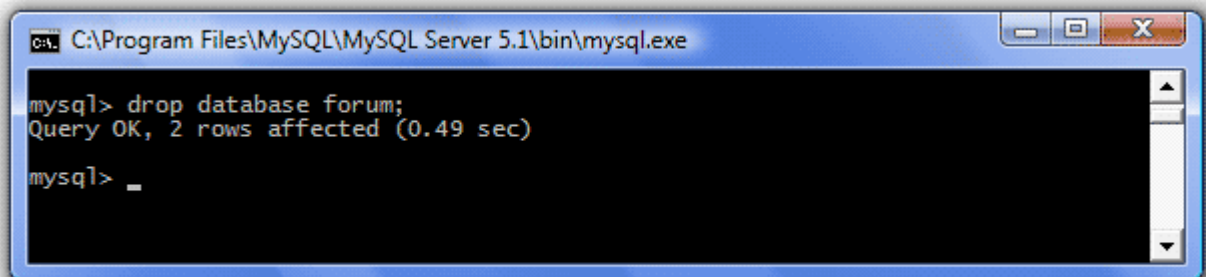
mysql> show tables;
+-----+
| Tables_in_forum |
+-----+
| posts           |
| users           |
+-----+
2 rows in set (0.06 sec)

mysql>
```

Наша таблица действительно удалена. Теперь давайте удалим и саму БД forum (удаляйте, не жалейте, ее все равно придется переделывать). Для этого напишем:

```
drop database имя_базы данных;
```

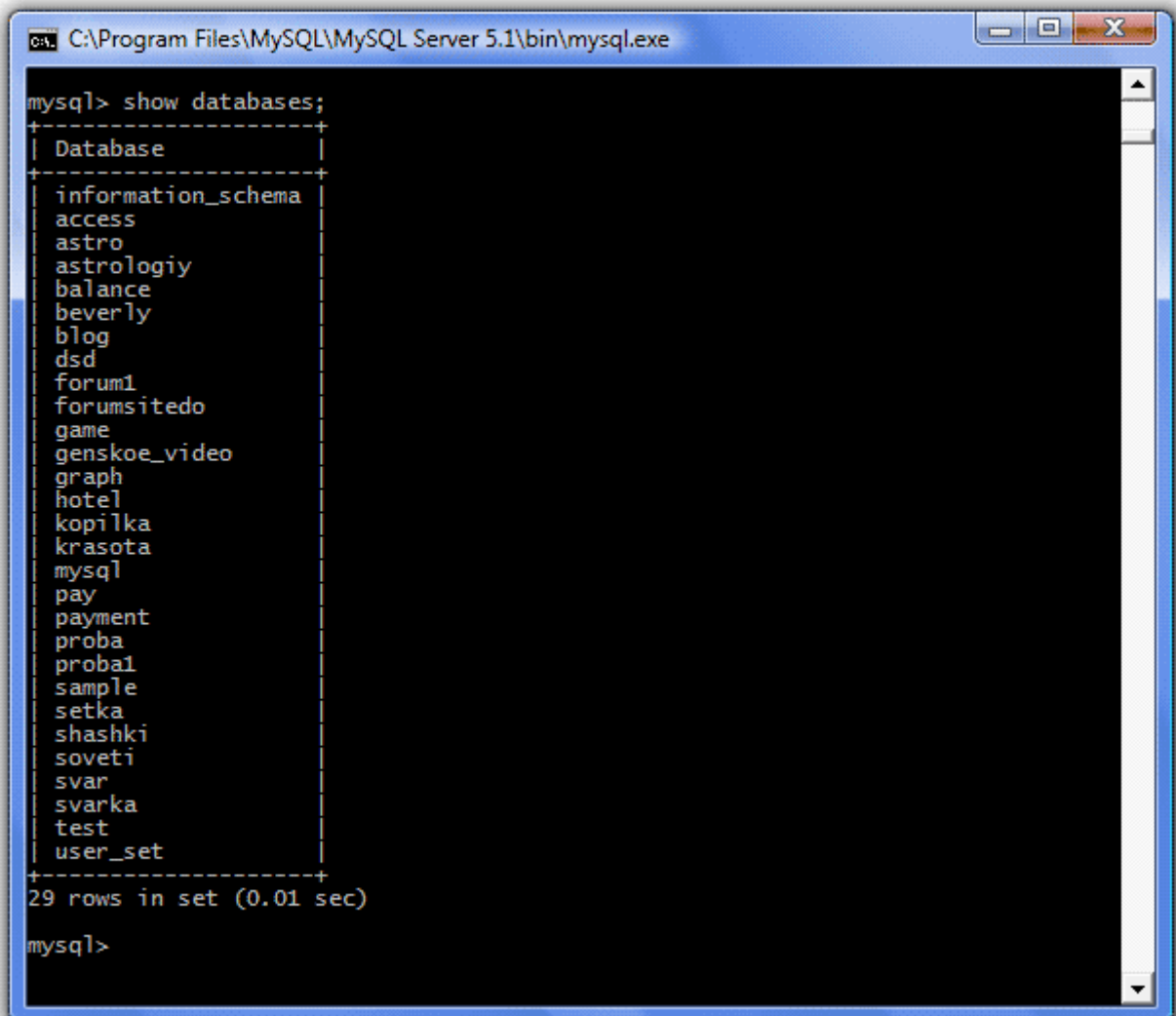
и нажмем Enter.



```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe
mysql> drop database forum;
Query OK, 2 rows affected (0.49 sec)

mysql> _
```

И убедитесь в этом, сделав запрос на все имеющиеся БД:



```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| access |
| astro |
| astrologiy |
| balance |
| beverly |
| blog |
| dsd |
| forum1 |
| forumsitedo |
| game |
| genskoe_video |
| graph |
| hotel |
| kopilka |
| krasota |
| mysql |
| pay |
| payment |
| proba |
| proba1 |
| sample |
| setka |
| shashki |
| soveti |
| svar |
| svarka |
| test |
| user_set |
+-----+
29 rows in set (0.01 sec)

mysql>
```

Создание таблиц

Цель: Закрепить знания и умения по созданию таблиц.

Теоретическая часть:

Усовершенствуем таблицы для нашего форума. Сначала разберем их. И начнем с таблицы `users` (пользователи). В ней у нас 4 столбца:

- `id_user` — целочисленные значения, значит будет тип `int`, ограничим его 10 символами — `int (10)`
- `name` — строковое значение `varchar`, ограничим его 20 символами — `varchar(20)`
- `email` — строковое значение `varchar`, ограничим его 50 символами — `varchar(50)`
- `password` — строковое значение `varchar`, ограничим его 15 символами — `varchar(15)`

Все значения полей обязательны для заполнения, значит надо добавить тип `NOT NULL`.

```
1. id_user int (10) NOT NULL
2. name varchar(20) NOT NULL
3. email varchar(50) NOT NULL
4. password varchar(15) NOT NULL
```

Первый столбец, как вы помните из концептуальной модели нашей БД, является первичным ключом (т.е. его значения уникальны, и они однозначно идентифицируют запись). Следить за уникальностью самостоятельно можно, но не рационально. Для этого в SQL есть специальный атрибут — `AUTO_INCREMENT`, который при обращении к таблице на добавление данных высчитывает максимальное значение этого столбца, полученное значение увеличивает на 1 и заносит его в столбец. Таким образом, в этом столбце автоматически генерируется уникальный номер, а следовательно тип `NOT NULL` излишен. Итак, присвоим атрибут столбцу с первичным ключом:

```
1. id_user int (10) AUTO_INCREMENT
2. name varchar(20) NOT NULL
3. email varchar(50) NOT NULL
4. password varchar(15) NOT NULL
```

Теперь надо указать, что поле `id_user` является первичным ключом. Для этого в SQL используется ключевое слово `PRIMARY KEY ()`, в скобочках указывается имя ключевого поля. Внесем изменения:

```
1. id_user int (10) AUTO_INCREMENT
2. name varchar(20) NOT NULL
3. email varchar(50) NOT NULL
4. password varchar(15) NOT NULL
5. PRIMARY KEY (id_user)
```

Итак, таблица готова, и ее окончательный вариант выглядит так:

```
1. create table users (
2. id_user int (10) AUTO_INCREMENT,
3. name varchar(20) NOT NULL,
```

```

4. email varchar(50) NOT NULL,
5. password varchar(15) NOT NULL,
6. PRIMARY KEY (id_user)
7. );

```

Теперь разберемся со второй таблицей — topics (темы). Рассуждая аналогично, имеем следующие поля:

```

1. id_topic int (10) AUTO_INCREMENT
2. topic_name varchar(100) NOT NULL
3. id_author int (10) NOT NULL
4. PRIMARY KEY (id_topic)

```

Но в модели нашей БД поле `id_author` является внешним ключом, т.е. оно может иметь только те значения, которые есть в поле `id_user` таблицы `users`. Для того, чтобы указать это в SQL есть ключевое слово `FOREIGN KEY` (), которое имеет следующий синтаксис:

```

1. FOREIGN KEY (имя_столбца_которое_является_внешним_ключом)
   REFERENCES имя_таблицы_родителя (имя_столбца_родителя);

```

Укажем, что `id_author` — внешний ключ:

```

1. id_topic int (10) AUTO_INCREMENT
2. topic_name varchar(100) NOT NULL
3. id_author int (10) NOT NULL
4. PRIMARY KEY (id_topic)
5. FOREIGN KEY (id_author) REFERENCES users (id_user)

```

Таблица готова, и ее окончательный вариант выглядит так:

```

1. create table topics (
2. id_topic int (10) AUTO_INCREMENT,
3. topic_name varchar(100) NOT NULL,
4. id_author int (10) NOT NULL,
5. PRIMARY KEY (id_topic),
6. FOREIGN KEY (id_author) REFERENCES users (id_user)
7. );

```

Осталась последняя таблица — posts (сообщения). Здесь все аналогично, только два внешних ключа:

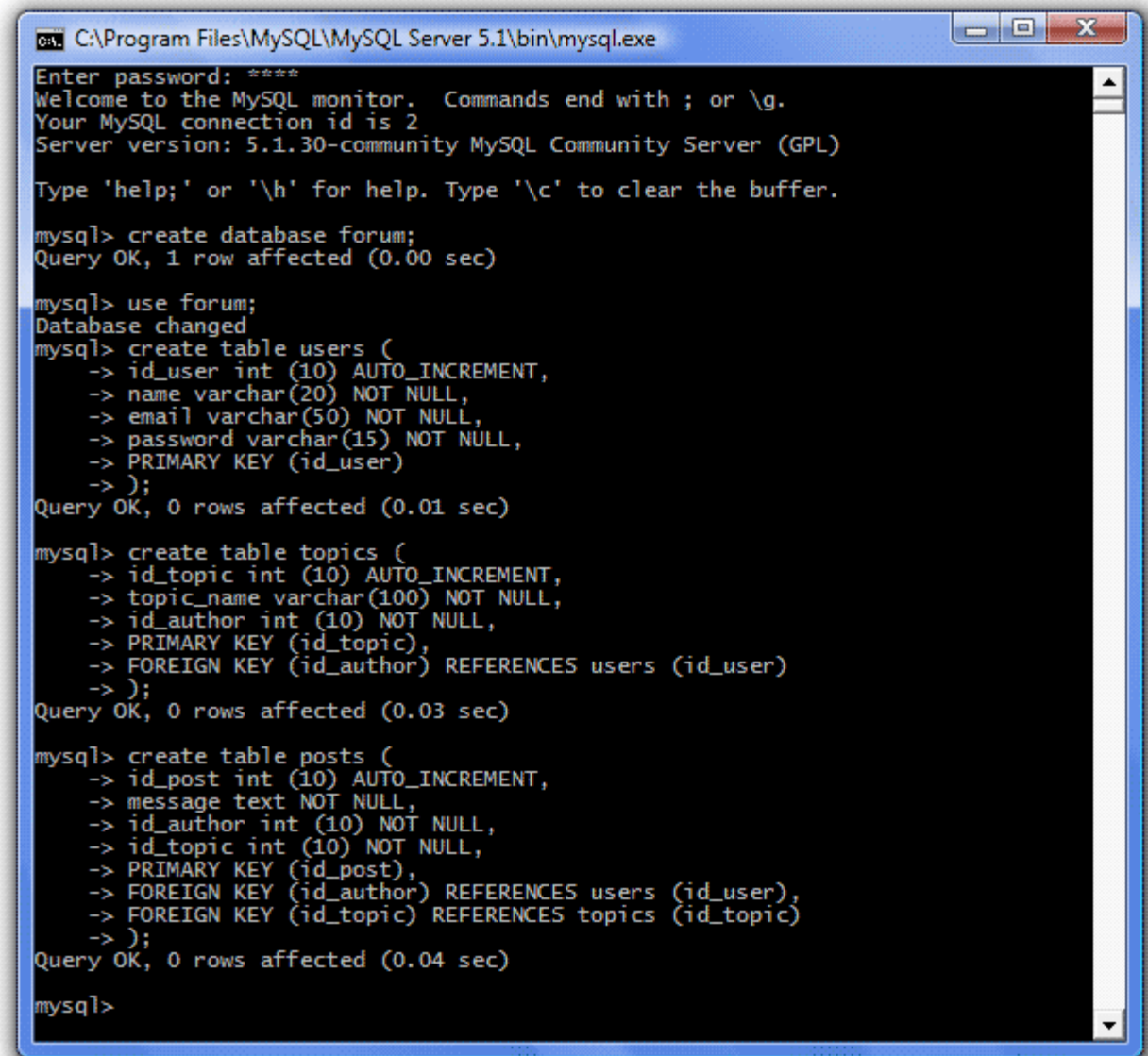
```

1. create table posts (
2. id_post int (10) AUTO_INCREMENT,
3. message text NOT NULL,
4. id_author int (10) NOT NULL,
5. id_topic int (10) NOT NULL,
6. PRIMARY KEY (id_post),
7. FOREIGN KEY (id_author) REFERENCES users (id_user),
8. FOREIGN KEY (id_topic) REFERENCES topics (id_topic)
9. );

```

Обратите внимание, внешних ключей у таблицы может быть несколько, а первичный ключ в MySQL может быть только один.

Запускаем сервер MySQL (В терминале прописываем команду MySQL —user root —password), вводим пароль, выбираем для использования базу данных `Forum` (`use forum;`) и создаем три наших таблицы:



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.30-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database forum;
Query OK, 1 row affected (0.00 sec)

mysql> use forum;
Database changed
mysql> create table users (
-> id_user int (10) AUTO_INCREMENT,
-> name varchar(20) NOT NULL,
-> email varchar(50) NOT NULL,
-> password varchar(15) NOT NULL,
-> PRIMARY KEY (id_user)
-> );
Query OK, 0 rows affected (0.01 sec)

mysql> create table topics (
-> id_topic int (10) AUTO_INCREMENT,
-> topic_name varchar(100) NOT NULL,
-> id_author int (10) NOT NULL,
-> PRIMARY KEY (id_topic),
-> FOREIGN KEY (id_author) REFERENCES users (id_user)
-> );
Query OK, 0 rows affected (0.03 sec)

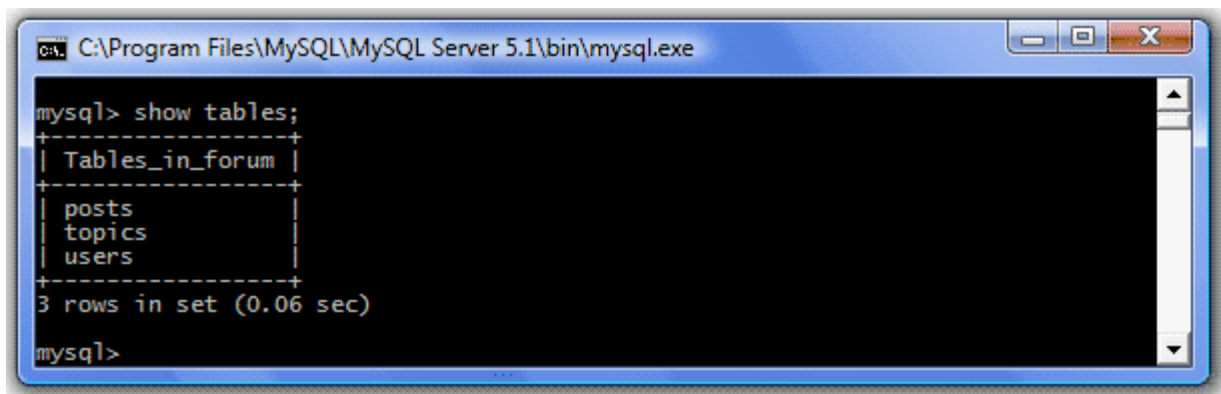
mysql> create table posts (
-> id_post int (10) AUTO_INCREMENT,
-> message text NOT NULL,
-> id_author int (10) NOT NULL,
-> id_topic int (10) NOT NULL,
-> PRIMARY KEY (id_post),
-> FOREIGN KEY (id_author) REFERENCES users (id_user),
-> FOREIGN KEY (id_topic) REFERENCES topics (id_topic)
-> );
Query OK, 0 rows affected (0.04 sec)

mysql>
```

Обратите внимание, одну команду можно писать в несколько строк, используя клавишу Enter (MySQL автоматически подставляет символ новой строки ->), и только после разделителя (точки с запятой) нажатие клавиши Enter приводит к выполнению запроса.

Помните, если вы сделали что-то не так, всегда можно удалить таблицу или всю БД с помощью оператора DROP. Исправлять что-то в командной строке крайне неудобно, поэтому иногда (особенно на начальном этапе) проще писать запросы в каком-нибудь редакторе, например в Блокноте, а затем копировать и вставлять их в черное окошко.

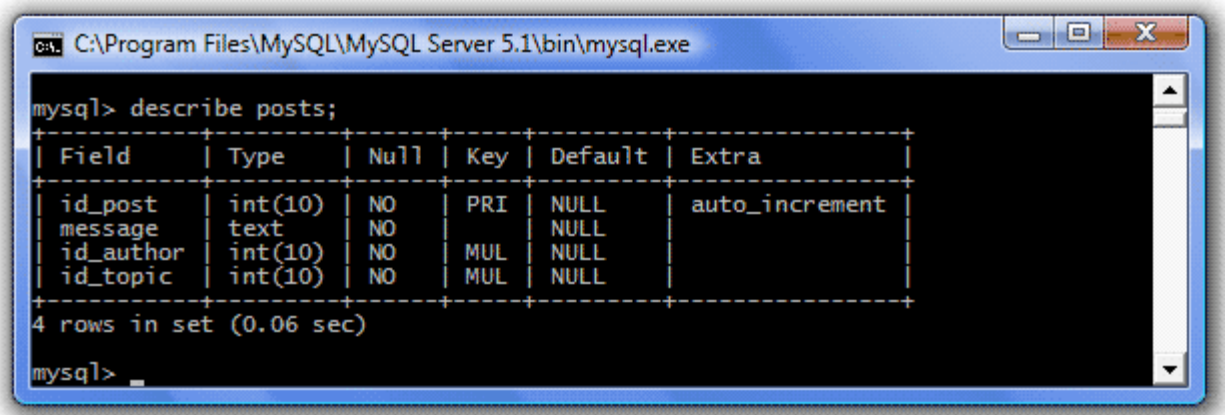
Итак, таблицы созданы, чтобы убедиться в этом вспомним о команде show tables:



```
mysql> show tables;
+-----+
| Tables_in_forum |
+-----+
| posts           |
| topics          |
| users           |
+-----+
3 rows in set (0.06 sec)

mysql>
```

И, наконец, посмотрим структуру нашей последней таблицы posts:



```
mysql> describe posts;
+-----+-----+-----+-----+-----+-----+
| Field      | Type   | Null | Key  | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id_post    | int(10) | NO   | PRI  | NULL    | auto_increment |
| message    | text   | NO   |      | NULL    |                |
| id_author  | int(10) | NO   | MUL  | NULL    |                |
| id_topic   | int(10) | NO   | MUL  | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.06 sec)

mysql> _
```

Теперь становятся понятны значения всех полей структуры, кроме поля DEFAULT. Это поле значений по умолчанию. Мы могли бы для какого-нибудь столбца (или для всех) указать значение по умолчанию. Например, если бы у нас было поле с названием «Женаты\Замужем» и типом ENUM ('да', 'нет'), то было бы разумно сделать одно из значений значением по умолчанию. Синтаксис был бы следующий:

```
1. married enum ('да', 'нет') NOT NULL default('да')
```

Т.е. это ключевое слово пишется через пробел после указания типа данных, а в скобках указывается значение по умолчанию.

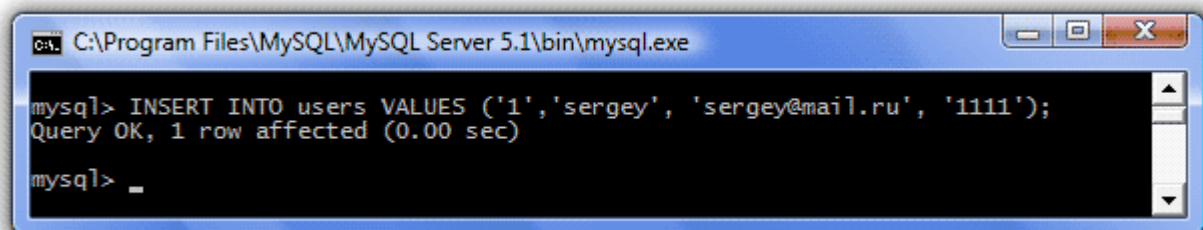
Но вернемся к нашим таблицам. Теперь нам необходимо внести данные в наши таблицы. На сайтах, вы обычно вводите информацию в какие-нибудь html-формы, затем сценарий на каком-либо языке (php, java...) извлекает эти данные из формы и заносит их в БД. Делает он это посредством SQL-запроса на внесение данных в базу. Писать сценарии на php мы пока не умеем, а вот отправлять SQL-запросы на внесение данных сейчас научимся.

Для этого используется оператор INSERT. Синтаксис можно использовать двух видов. Первый вариант используется для внесения данных во все поля таблицы:

```
1. INSERT INTO имя_таблицы VALUES
   ('значение_первого_столбца', 'значение_второго_столбца',
    ..., 'значение_последнего_столбца');
```


Давайте попробуем внести в нашу таблицу users следующие значения:

1. INSERT INTO users VALUES ('1','sergey', 'sergey@mail.ru', '1111');



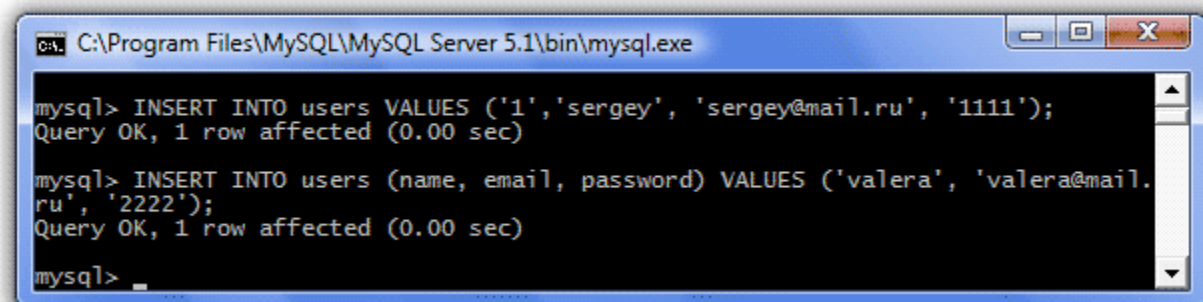
```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe
mysql> INSERT INTO users VALUES ('1','sergey', 'sergey@mail.ru', '1111');
Query OK, 1 row affected (0.00 sec)
mysql> _
```

Второй вариант используется для внесения данных в некоторые поля таблицы:

1. INSERT INTO имя_таблицы ('имя_столбца', 'имя_столбца') VALUES ('значение первого столбца', 'значение второго столбца');

В нашей таблице users все поля обязательны для заполнения, но наше первое поле имеет ключевое слово — AUTO_INCREMENT (т.е. оно заполняется автоматически), поэтому мы можем пропустить этот столбец:

1. INSERT INTO users (name, email, password) VALUES ('valera', 'valera@mail.ru', '2222');

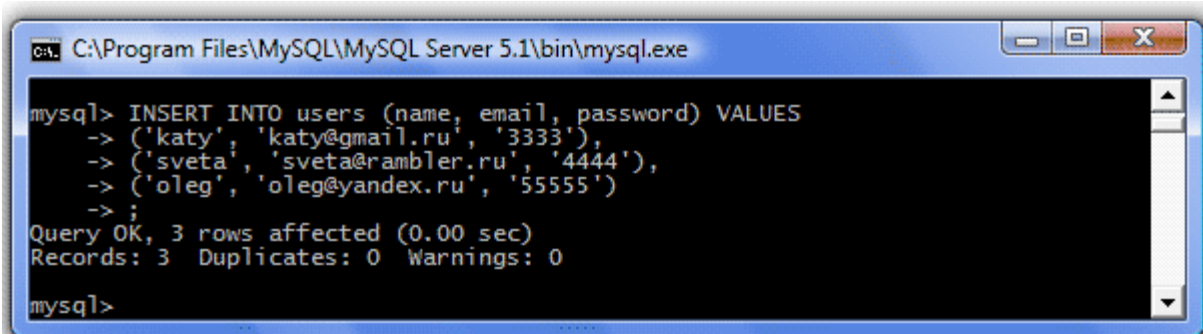


```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe
mysql> INSERT INTO users VALUES ('1','sergey', 'sergey@mail.ru', '1111');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO users (name, email, password) VALUES ('valera', 'valera@mail.ru', '2222');
Query OK, 1 row affected (0.00 sec)
mysql> _
```

Если бы у нас были поля с типом NULL, т.е. необязательные для заполнения, мы бы тоже могли их проигнорировать. А вот если попытаться оставить пустым поле со значением NOT NULL, то сервер выдаст сообщение об ошибке и не выполнит запрос. Кроме того, при внесении данных сервер проверяет связи между таблицами. Поэтому вам не удастся внести в поле, являющееся внешним ключом, значение, отсутствующее в связанной таблице. В этом вы убедитесь, внося данные в оставшиеся две таблицы.

Но прежде внесем информацию еще о нескольких пользователях. Чтобы добавить сразу несколько строк, надо просто перечислять скобки со значениями через запятую:

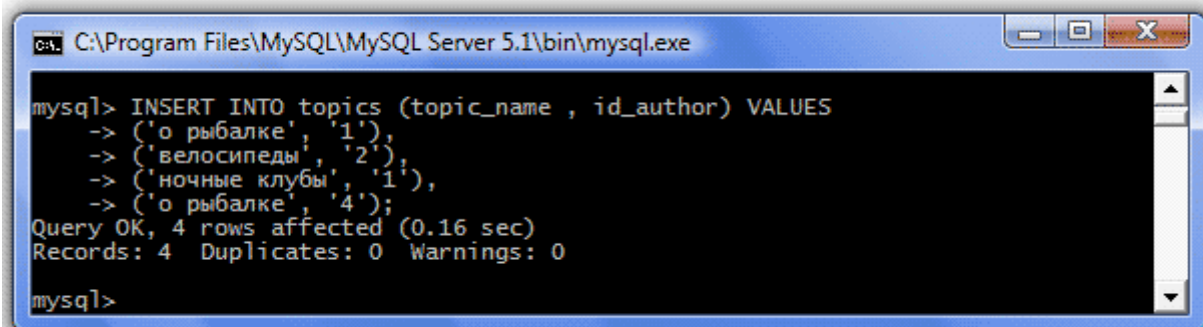


```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe

mysql> INSERT INTO users (name, email, password) VALUES
-> ('katy', 'katy@gmail.ru', '3333'),
-> ('sveta', 'sveta@rambler.ru', '4444'),
-> ('oleg', 'oleg@yandex.ru', '5555')
-> ;
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql>
```

Теперь внесем данные во вторую таблицу — topics (темы). Все тоже самое, но надо помнить, что значения в поле `id_author` должны присутствовать в таблице `users` (пользователи):

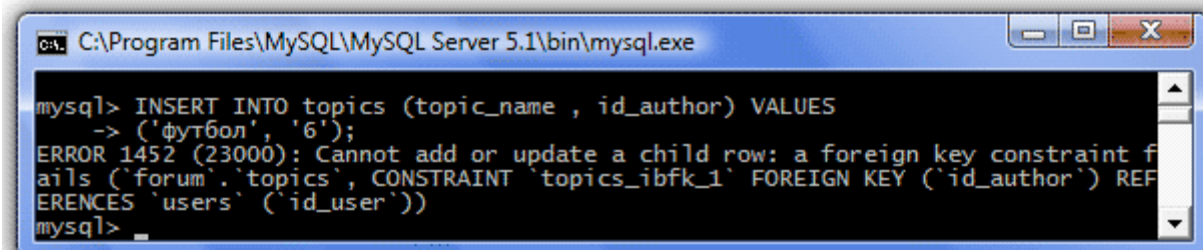


```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe

mysql> INSERT INTO topics (topic_name , id_author) VALUES
-> ('о рыбалке', '1'),
-> ('велосипеды', '2'),
-> ('ночные клубы', '1'),
-> ('о рыбалке', '4');
Query OK, 4 rows affected (0.16 sec)
Records: 4 Duplicates: 0 Warnings: 0

mysql>
```

Теперь давайте попробуем внести еще одну тему, но с `id_author`, которого в таблице `users` нет (т.к. мы внесли в таблицу `users` только 5 пользователей, то `id=6` не существует):



```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe

mysql> INSERT INTO topics (topic_name , id_author) VALUES
-> ('футбол', '6');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails ('forum`.`topics`, CONSTRAINT `topics_ibfk_1` FOREIGN KEY (`id_author`) REFERENCES `users` (`id_user`))
mysql>
```

Сервер выдает ошибку и говорит, что не может внести такую строку, т.к. в поле, являющемся внешним ключом, стоит значение, отсутствующее в связанной таблице `users`.

Теперь внесем несколько строк в таблицу `posts` (сообщения), помня, что в ней у нас 2 внешних ключа, т.е. `id_author` и `id_topic`, которые мы будем вносить должны присутствовать в связанных с ними таблицах:

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql> INSERT INTO posts (message, id_author, id_topic) VALUES
-> ('думаю, надо сделать так', '1', '1'),
-> ('согласен', '2', '4'),
-> ('а еще можно сделать так', '3', '1'),
-> ('согласен', '2', '1');
Query OK, 4 rows affected (0.07 sec)
Records: 4 Duplicates: 0 Warnings: 0
mysql>
```

Итак, в нашей БД forum есть таблицы: users (пользователи), topics (темы) и posts (сообщения). И мы хотим посмотреть, какие данные в них содержатся. Для этого в SQL существует оператор SELECT. Синтаксис его использования следующий:

1. SELECT что_выбрать FROM откуда_выбрать;

Вместо «что_выбрать» мы должны указать либо имя столбца, значения которого хотим увидеть, либо имена нескольких столбцов через запятую, либо символ звездочки (*), означающий выбор всех столбцов таблицы. Вместо «откуда_выбрать» следует указать имя таблицы.

Давайте посмотрим все столбцы из таблицы users:

1. SELECT * FROM users;

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql> SELECT * FROM users;
+----+-----+-----+-----+
| id_user | name  | email          | password |
+----+-----+-----+-----+
| 1       | sergey | sergey@mail.ru | 1111     |
| 2       | valera | valera@mail.ru | 2222     |
| 3       | katy   | katy@gmail.ru  | 3333     |
| 4       | sveta  | sveta@rambler.ru | 4444     |
| 5       | oleg   | oleg@yandex.ru | 5555     |
+----+-----+-----+-----+
5 rows in set (0.10 sec)
mysql>
```