

ЛАБОРАТОРНАЯ РАБОТА №6/1. РАБОТА С MYSQL.

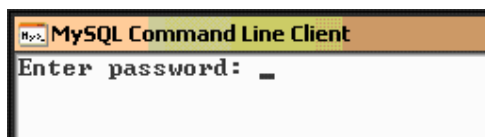
ЦЕЛЬ. Изучить основы работы с сервером MYSQL.

Теоретические сведения.

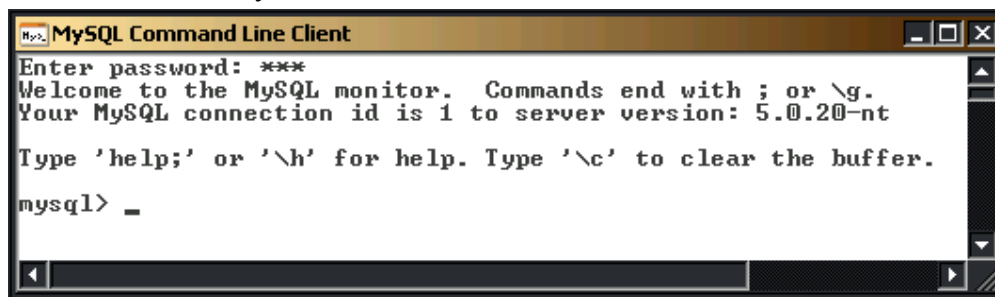
Монитор mysql запускаем из меню “Пуск” → “Программы” → “MySQL” → “MySQL Server 5.0” → “MySQL Command Line Client”, или через командную строку путем ввода:

mysql -u имя_пользователя -p.

В качестве имени пользователя введите **root**. После запуска программа запросит пароль:



Введите пароль, который вы установили при установке MySQL. Если введен верный пароль, система выдаст следующее сообщение:



Теперь можно приступить к работе с MYSQL. В настоящей работе создадим небольшую базу данных и несколько хранимых процедур и триггеров.

Посмотрим наличие существующих баз данных, как показано на рис.1.1. (команда show databases;) Не забывайте после команды ставить точку с запятой.

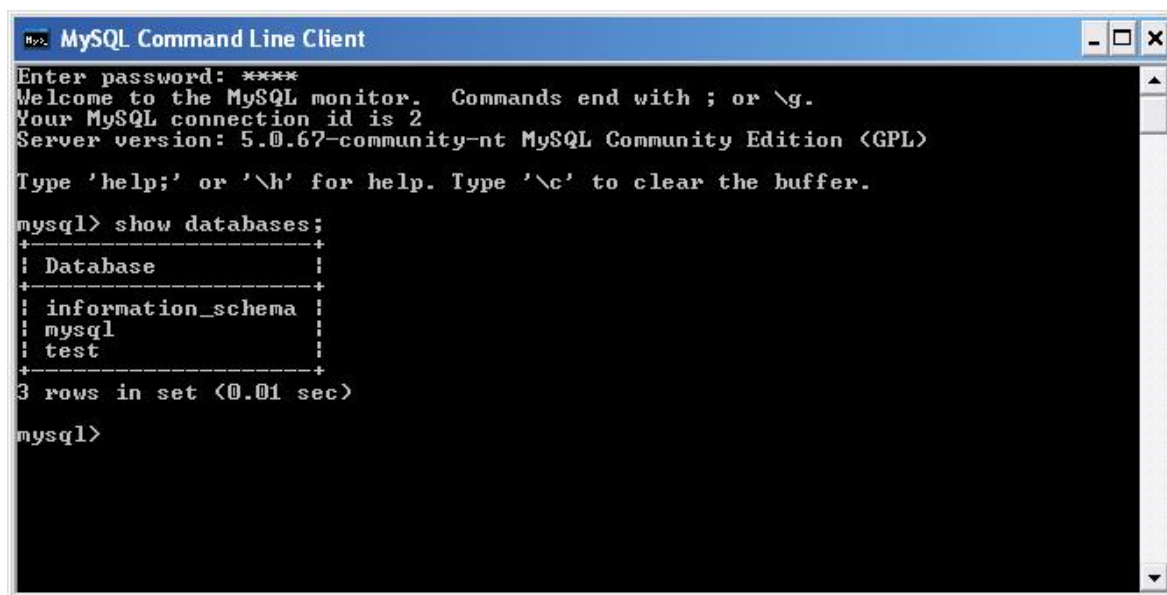


Рис.1.1

Создаем базу командой

```
>create database base1;
```

Подключаемся к базе командой

```
>use base1;
```

Создаем таблицу sklad:

```
>create table sklad(Tovar varchar(25),price int, kol int,postavka Date,Primary  
>KEY(tovar));
```

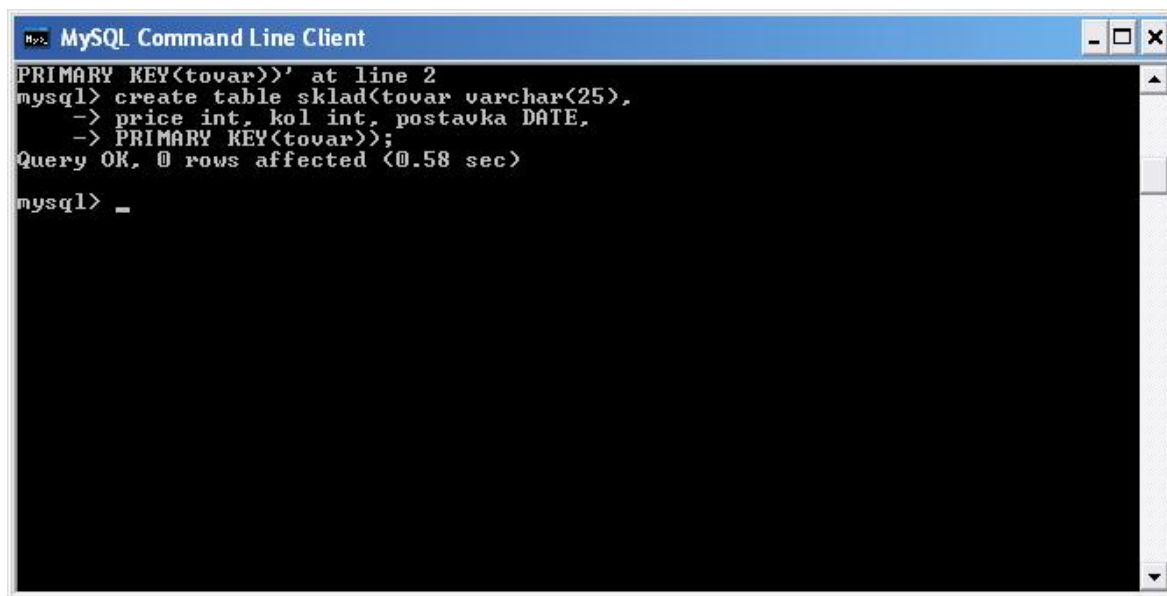


Рис.2

Теперь занесем в таблицу данные , набранные в текстовом файле. Пример команды загрузки:

```
mysql> LOAD DATA LOCAL INFILE '/path/pet.txt' INTO TABLE pet;
```

Записи в файле располагаются одна под другой и имеют слндующий примерный вид:

Whistler	Gwen	bird	\N	1997-12-09	\N
----------	------	------	----	------------	----

Значения полей отделяются символом табуляции (TAB). \N соответствует пустому значению. Наш файл такой

milk	200	130	2010-01-18
bread	1400	1000	2010-01-17
butter	1800	1200	2010-01-16
curd	3000	500	2010-01-10

Запишем его в таблицу:

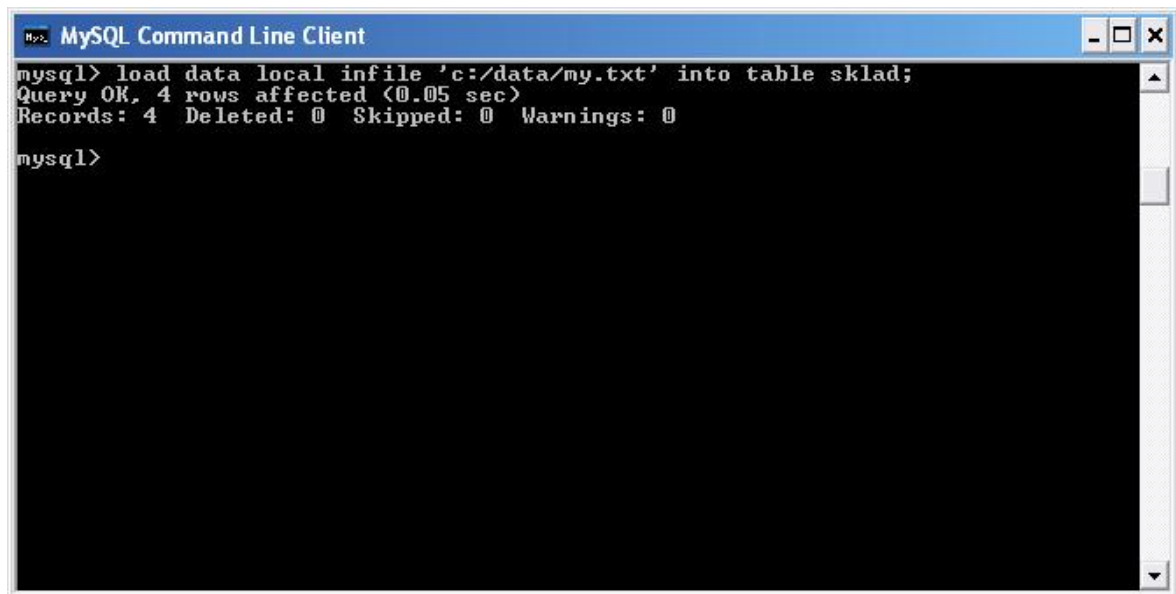
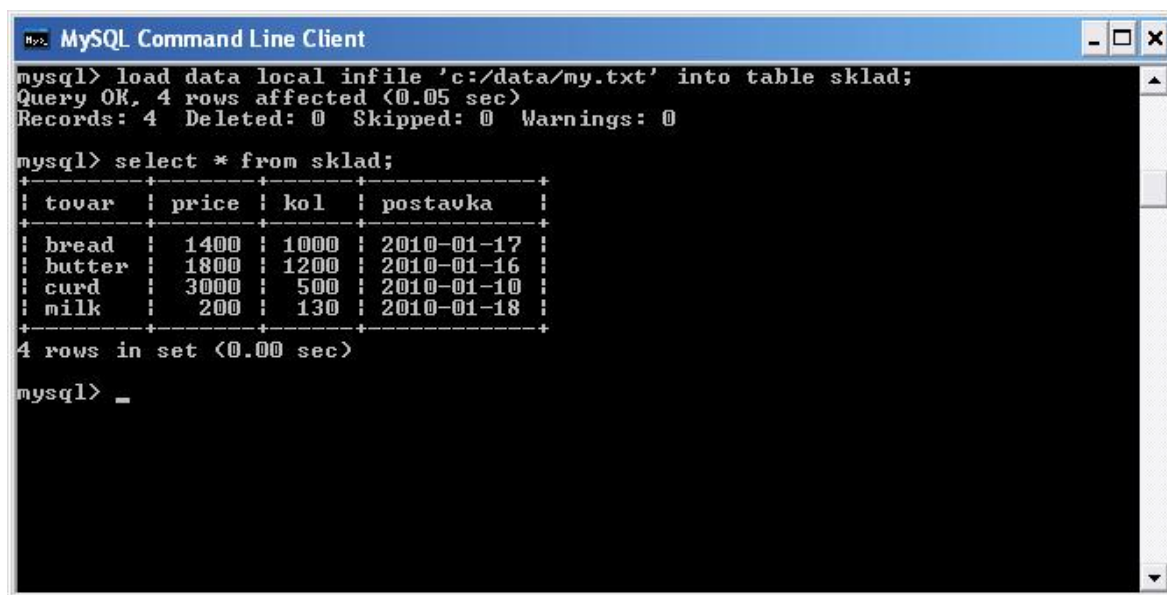


Рис.3

Теперь можно посмотреть содержимое файла:

```
>select * from sklad;
```



```
mysql> load data local infile 'c:/data/my.txt' into table sklad;
Query OK, 4 rows affected (0.05 sec)
Records: 4 Deleted: 0 Skipped: 0 Warnings: 0

mysql> select * from sklad;
+-----+-----+-----+-----+
| tovar  | price | kol  | postavka |
+-----+-----+-----+-----+
| bread  | 1400  | 1000 | 2010-01-17 |
| butter | 1800  | 1200 | 2010-01-16 |
| curd   | 3000  | 500  | 2010-01-10 |
| milk   | 200   | 130  | 2010-01-18 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> _
```

Рис.4

Приступим к созданию хранимых процедур (эта возможность имеется только для сервера MySQL версии 5 и выше). Наша первая процедура такова:



```
mysql> delimiter //
mysql> create procedure findcena(IN tov varchar(25), OUT cen INT)
-> begin
-> select price into cen from sklad where tovar=tov;
-> end;
-> //
Query OK, 0 rows affected (0.14 sec)

mysql> delimiter ;
mysql> call findcena('bread',@x);
Query OK, 0 rows affected (0.05 sec)

mysql> select @x;
+-----+
| @x    |
+-----+
| 1400  |
+-----+
1 row in set (0.00 sec)

mysql> _
```

Рис.5

Обратим внимание на то, что написанию процедуры предшествует изменение конечного символа-разделителя(delimiter):

```
>delimiter //
```

Затем идет собственно текст процедуры:

```
>create procedure findcena(IN tov varchar(25), OUT cen INT)
>begin
>select price into cen from sklad where Товар=tov;
>end;
>//
```

Затем восстанавливаем разделитель:

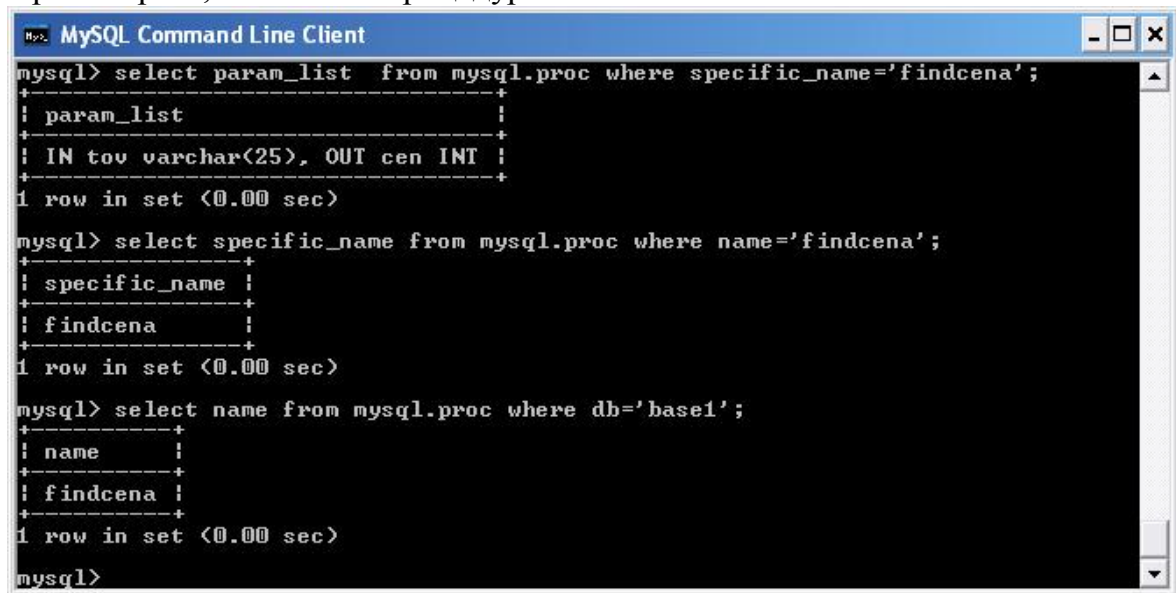
```
>delimiter ;
```

К сожалению, хранимую процедуру нельзя ввести из файла (как данные).

После этого можно вызывать процедуру и посмотреть результат:

```
>call findcena('bread',@x);
>select @x;
```

Просмотреть, какие есть процедуры в вашей базе можно так:



```
mysql> select param_list from mysql.proc where specific_name='findcena';
+-----+
| param_list |
+-----+
| IN tov varchar(25), OUT cen INT |
+-----+
1 row in set (0.00 sec)

mysql> select specific_name from mysql.proc where name='findcena';
+-----+
| specific_name |
+-----+
| findcena |
+-----+
1 row in set (0.00 sec)

mysql> select name from mysql.proc where db='base1';
+-----+
| name |
+-----+
| findcena |
+-----+
1 row in set (0.00 sec)

mysql>
```

Ри.6

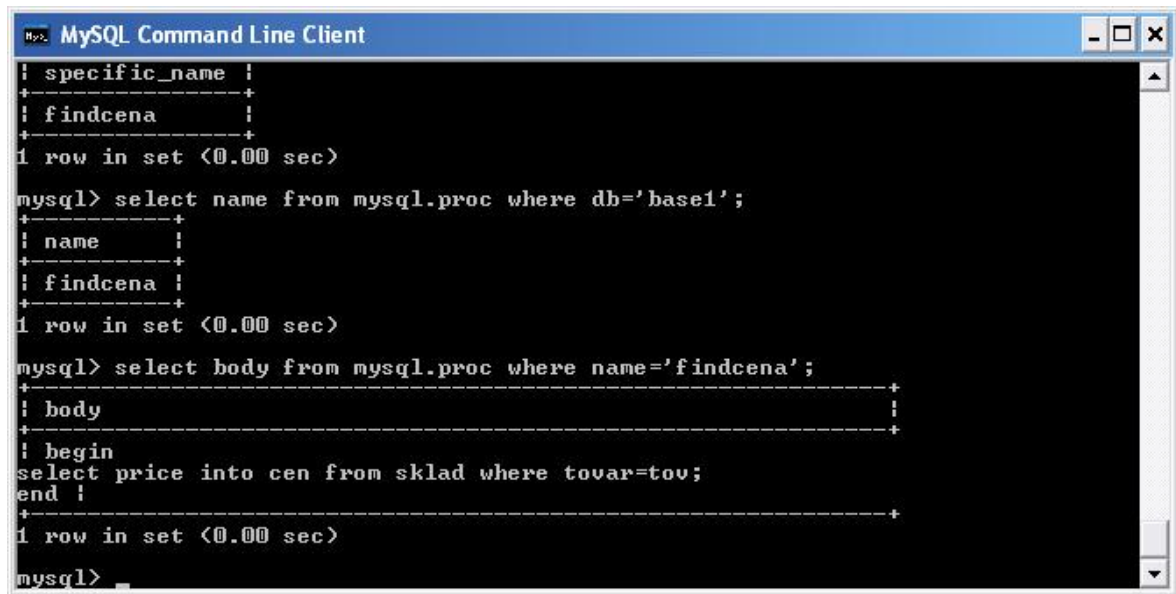
```
>select name from mysql.proc where db='base1';
```

Содержимое процедуры можно посмотреть таким образом:

```
>show create procedure findcena;
```

Или так:

```
>select body from mysql.proc where name='findcena';
```



```
MySQL Command Line Client
+-----+
| specific_name |
+-----+
| findcena      |
+-----+
1 row in set <0.00 sec>

mysql> select name from mysql.proc where db='base1';
+-----+
| name      |
+-----+
| findcena  |
+-----+
1 row in set <0.00 sec>

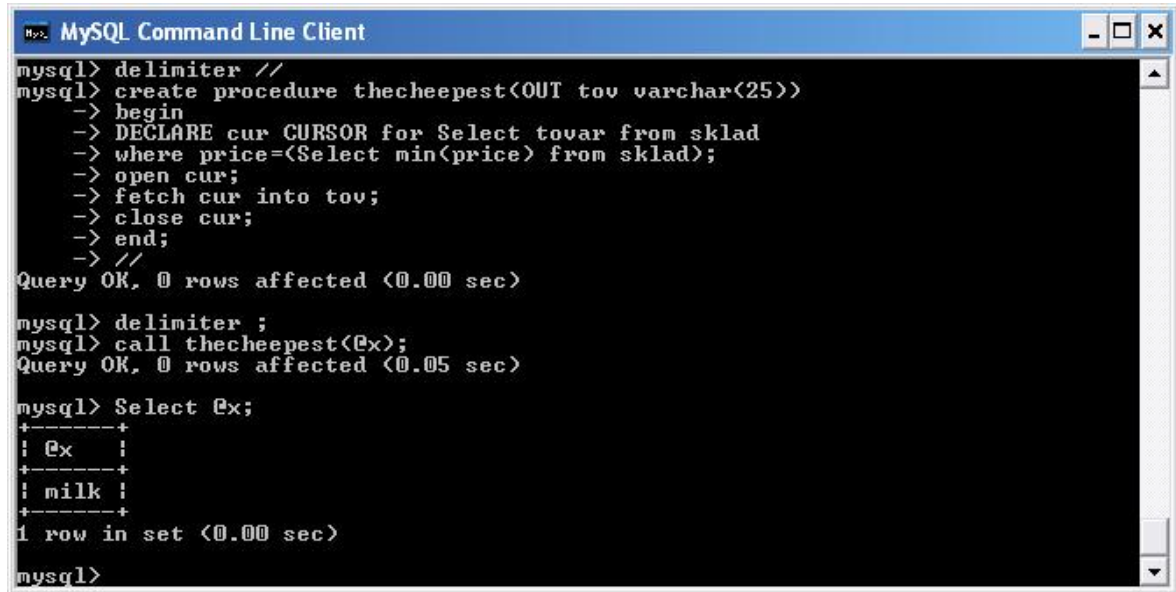
mysql> select body from mysql.proc where name='findcena';
+-----+
| body |
+-----+
| begin
select price into cen from sklad where tovar=tov;
end |
+-----+
1 row in set <0.00 sec>

mysql>
```

Рис.7

Продemonстрируем сейчас работу с курсором.

Следующая процедура использует курсор для выборки товара с минимальной ценой:



```
MySQL Command Line Client
mysql> delimiter //
mysql> create procedure thecheapest(OUT tov varchar(25))
-> begin
-> DECLARE cur CURSOR for Select tovar from sklad
-> where price=(Select min(price) from sklad);
-> open cur;
-> fetch cur into tov;
-> close cur;
-> end;
-> //
Query OK, 0 rows affected <0.00 sec>

mysql> delimiter ;
mysql> call thecheapest(@x);
Query OK, 0 rows affected <0.05 sec>

mysql> Select @x;
+-----+
| @x      |
+-----+
| milk    |
+-----+
1 row in set <0.00 sec>

mysql>
```

Рис.8

Курсор необходимо объявить как предложение SELECT:

DECLARE cur CURSOR for Select Tovar from sklad ... ;

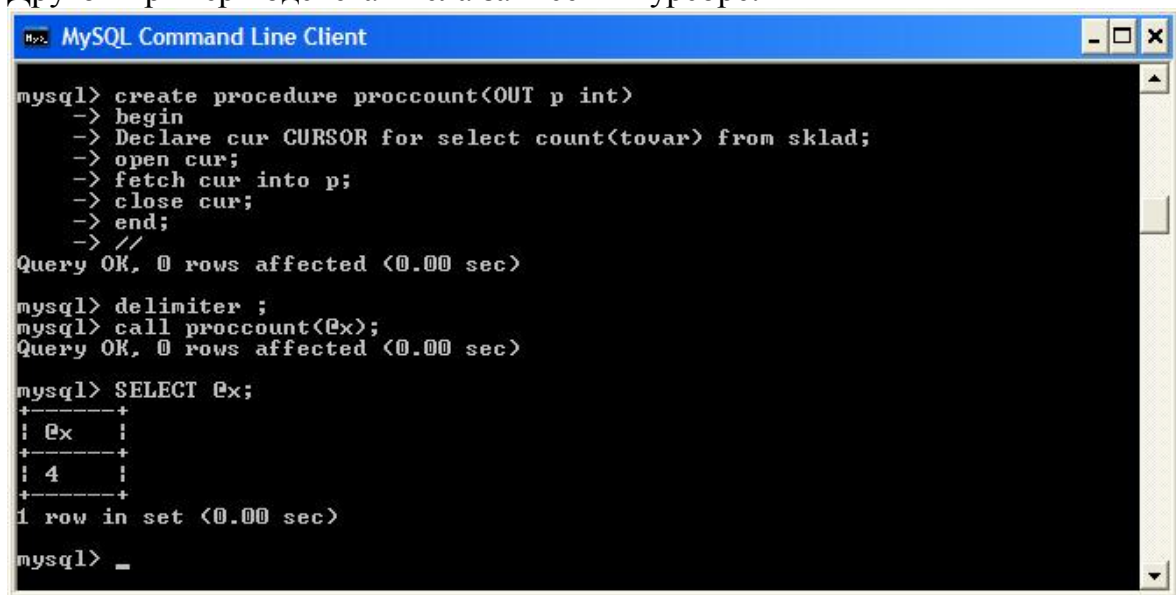
Для работы курсор следует открыть

Open cur;

А потом и закрыть

Close cur;

Другой пример подсчета числа записей в курсоре:



```
mysql> create procedure proccount(OUT p int)
-> begin
-> Declare cur CURSOR for select count(tovar) from sklad;
-> open cur;
-> fetch cur into p;
-> close cur;
-> end;
-> //
Query OK, 0 rows affected (0.00 sec)

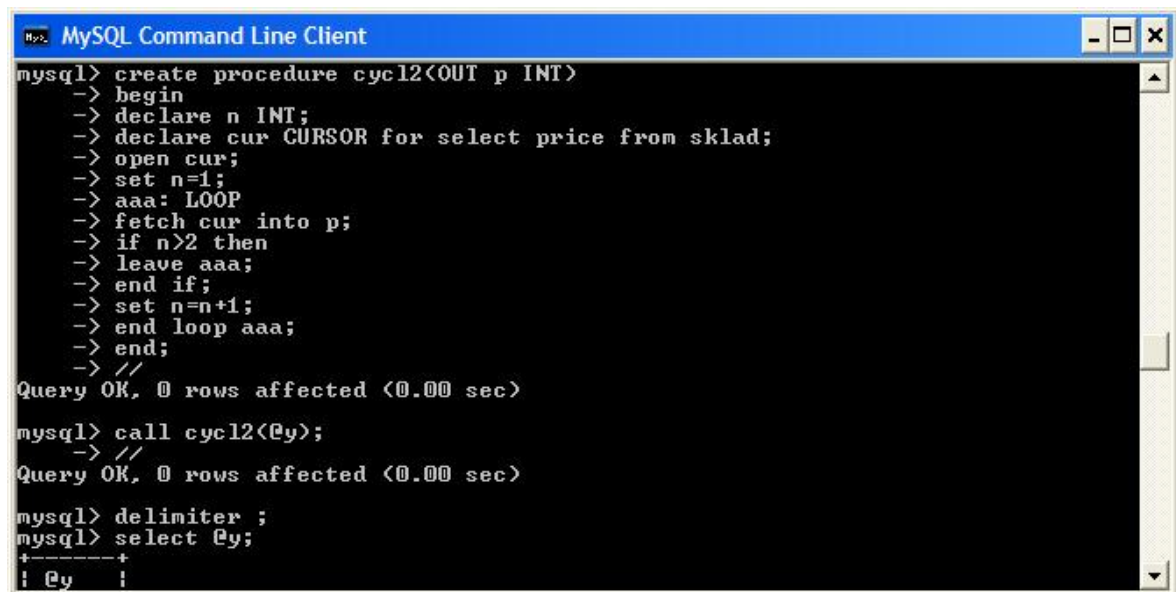
mysql> delimiter ;
mysql> call proccount(0x);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT 0x;
+-----+
| 0x    |
+-----+
| 4     |
+-----+
1 row in set (0.00 sec)

mysql> _
```

Рис.9

Пример с циклом:



```
mysql> create procedure cycl2(OUT p INT)
-> begin
-> declare n INT;
-> declare cur CURSOR for select price from sklad;
-> open cur;
-> set n=1;
-> aaa: LOOP
-> fetch cur into p;
-> if n>2 then
-> leave aaa;
-> end if;
-> set n=n+1;
-> end loop aaa;
-> end;
-> //
Query OK, 0 rows affected (0.00 sec)

mysql> call cycl2(0y);
-> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
mysql> select 0y;
+-----+
| 0y    |
+-----+
```

Рис.10

Цикл организуется между
aaa: LOOP

...

End LOOP aaa;

Внутри цикла выполняется преждевременный выход по команде

leave aaa;

если срабатывает условие:

If n>2.

Ясно, что набирать текст процедур крайне неудобно с консоли. Поэтому можно набрать текст в текстовом файле, например, такой:

```
delimiter //  
create procedure proc3(OUT p int)  
begin  
declare cur CURSOR for select count(tovar) from sklad;  
open cur;  
fetch cur into p;  
end;  
//  
delimiter ;  
call proc3(@z);  
select @z;
```

Назовем этот файл dat.txt. Теперь можно из консоли выдать следующую команду: source d:\work\dat.txt

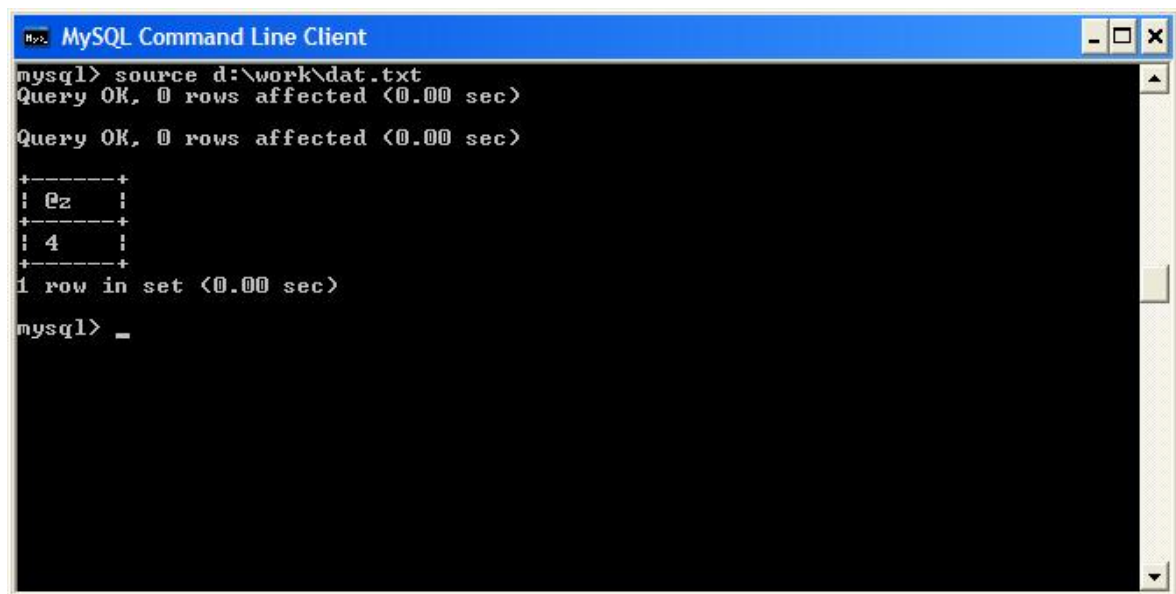


Рис.11

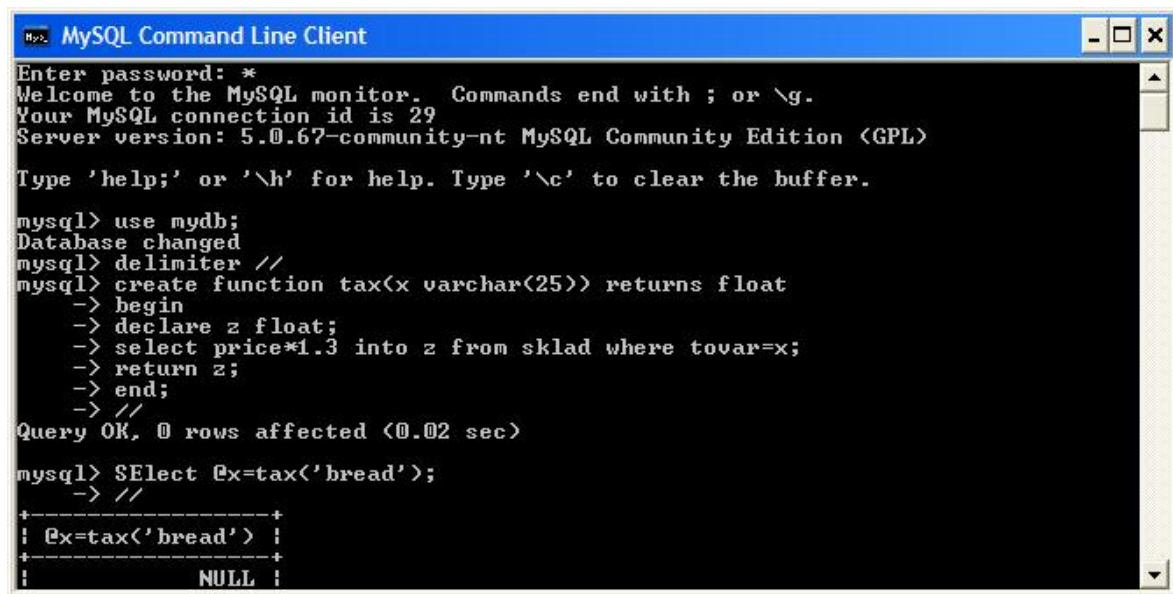
Таким образом можно облегчить работу по набору текста процедур.

Рассмотрим теперь, как создавать функции MySQL.

Общий формат таков:


```
>create function myfun(x INT) returns int
>begin
>...
>return val;
>end;
```

Значение функции возвращаем оператором return. Рассмотрим пример.



```
MySQL Command Line Client
Enter password: *
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 5.0.67-community-nt MySQL Community Edition (GPL)

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

mysql> use mydb;
Database changed
mysql> delimiter //
mysql> create function tax(x varchar(25)) returns float
-> begin
-> declare z float;
-> select price*1.3 into z from sklad where tovar=x;
-> return z;
-> end;
-> //
Query OK, 0 rows affected (0.02 sec)

mysql> $Eselect @x=tax('bread');
-> //
+-----+
! @x=tax('bread') !
+-----+
!                !
+-----+
```

Рис.12

ЗАДАНИЕ.

Создать хранимые процедуры для

- 1) Вставки записи и ее обновления.
- 2) Подсчета налога со всех товаров.
- 3) Получения точного названия товара по первым буквам.
- 4) Работы с курсором (подсчитать число записей с одинаковым названием товара).

Создать функции для:

- 1) Вычисления налога в зависимости от цены и количества товара
- 2) Вычисления средней цены товара по таблице

ЛАБОРАТОРНАЯ РАБОТА №6/2. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ В MYSQL.

ЦЕЛЬ. Изучить принципы работы с регулярными выражениями.

Теоретические сведения.

Регулярное выражение определяет множество строк, удовлетворяющих поисковому шаблону.

Рассмотрим следующую команду:

```
>Select 'fofo' REGEXP '^fo';
```

Эта команда выдаст значение true (Истина). Поисковый шаблон представлен строкой

```
'^fo'
```

Символ ^ означает, что поиск ведется с начала строки. Конструкция ^fo означает, что строка должна начинаться комбинацией fo.

Символ * означает повторение сколько угодно раз данного символа. Пример

```
>Select 'foofo' REGEXP '^fo*'
```

также вернет истину.

Знак вопроса соответствует любому одному символу:

```
>Select 'fofo' REGEXP '^f?fo'
```

также вернет истину. Заметим, что допустим только один знак вопроса.

Вместо вопроса можно использовать комбинацию \. (можно использовать сколько угодно таких комбинаций в поисковом шаблоне см. рис.1)

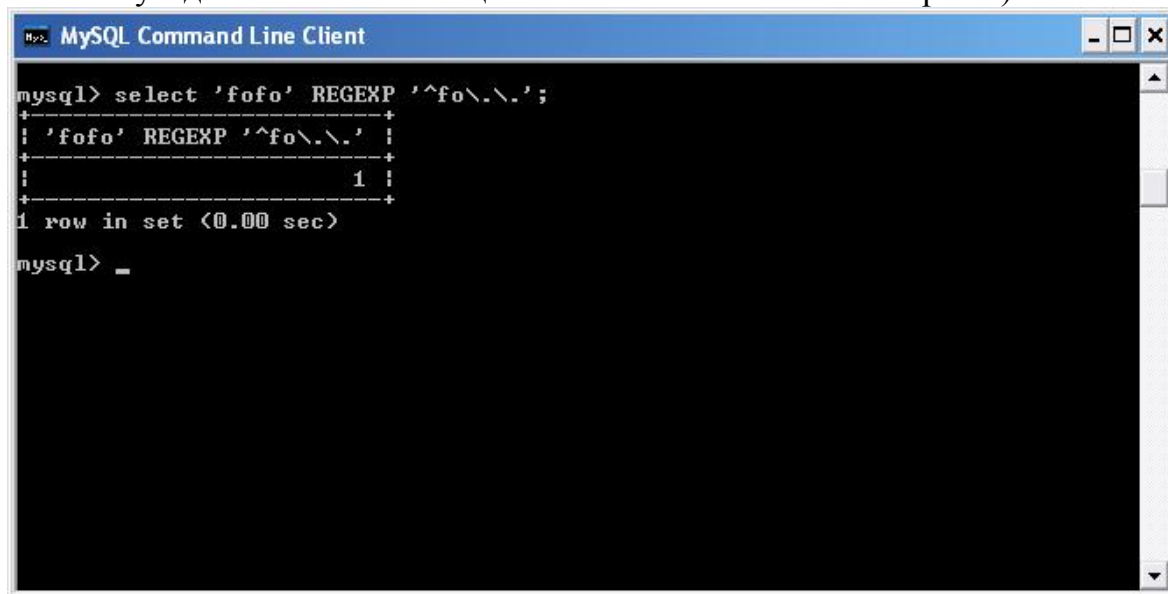
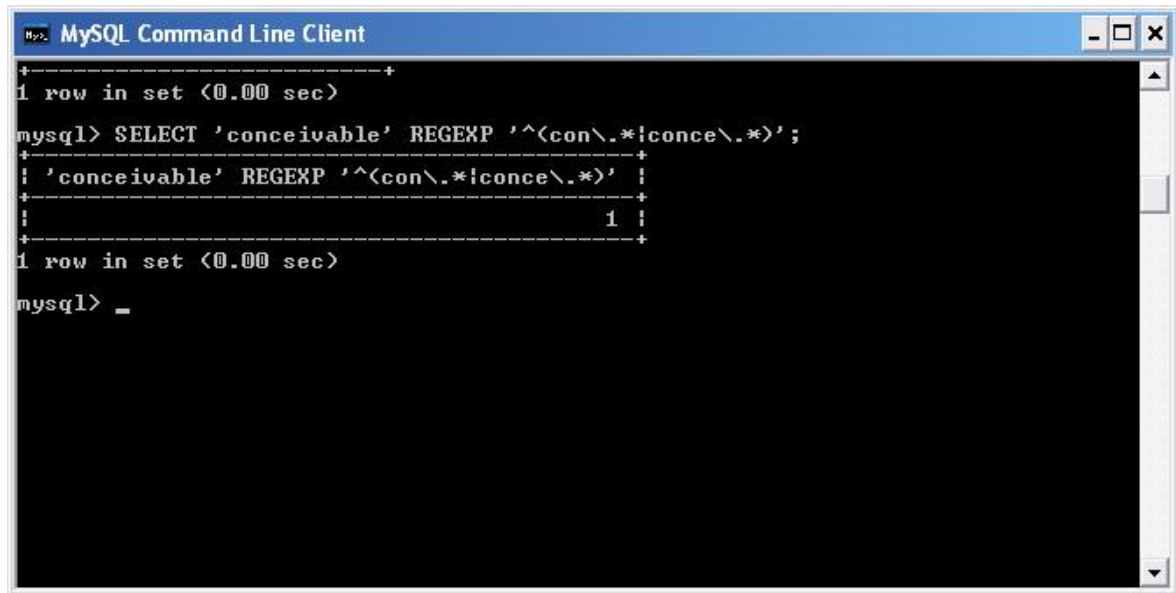


Рис.1

Можно проверить, что слово начинается с той или иной последовательности символов:

```
>Select 'conceivable' REGEXP '^<con\.*|concei\.*>'
```



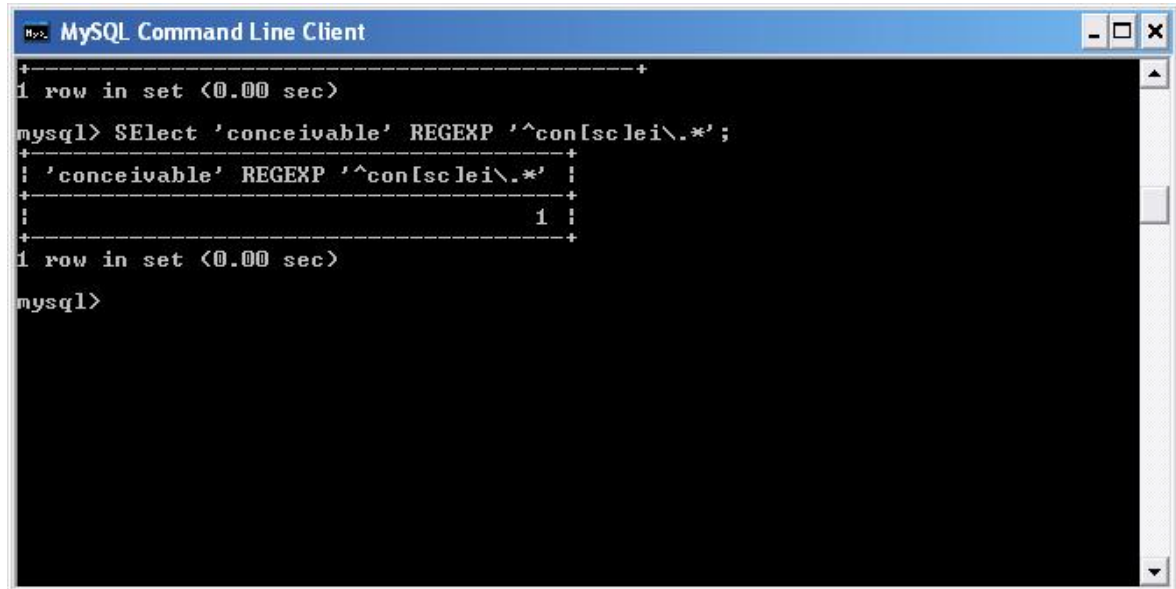
```
MySQL Command Line Client
+-----+
1 row in set (0.00 sec)

mysql> SELECT 'conceivable' REGEXP '^<con\.*|concei\.*>';
+-----+
| 'conceivable' REGEXP '^<con\.*|concei\.*>' |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)

mysql> _
```

Рис.2

Следующая конструкция



```
MySQL Command Line Client
+-----+
1 row in set (0.00 sec)

mysql> $Eselect 'conceivable' REGEXP '^con[sc]lei\.*';
+-----+
| 'conceivable' REGEXP '^con[sc]lei\.*' |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Рис.3

Проверяет, что слово стартует с concei или с concei. Конструкция в квадратных скобках [sc] означает, что может использоваться любой из указанных символов. Несколько видоизменим этот пример:

```
MySQL Command Line Client
1 row in set <0.00 sec>
mysql> $Select 'conceivable' REGEXP '^con(sei!cei)\.*';
+-----+
| 'conceivable' REGEXP '^con(sei!cei)\.*' |
+-----+
| 1 |
+-----+
1 row in set <0.00 sec>
mysql> _
```

Здесь уже проверяем, что слово содержит сочетание sei или sei.
Рассмотрим теперь следующий пример:

```
MySQL Command Line Client
mysql> SELECT 'abcd' REGEXP '[a-z]*';
+-----+
| 'abcd' REGEXP '[a-z]*' |
+-----+
| 1 |
+-----+
1 row in set <0.02 sec>
mysql> _
```

Рис.5

Шаблон REGEXP '[a-z]*' означает, что первая часть слова состоит из строчных английских букв от а до z. Вышеприведенный пример не работает в случае, показанном ниже:

```
MySQL Command Line Client
mysql> SELECT '2a' REGEXP '^[a-d]';
+-----+
| '2a' REGEXP '^[a-d]' |
+-----+
| 0 |
+-----+
1 row in set (0.00 sec)

mysql>
```

Следует отметить, что выражение в квадратных скобках означает любой один символ из указанного диапазона. Наличие символа ^ означает, что проверка выполняется с начала строки. Если убрать этот символ, то все сработает.

Проверить, что символ не содержится в последовательности можно поместив перед ним знак ^ (см. рисунок)

```
MySQL Command Line Client
Database changed
mysql> select '2a' REGEXP '^[^a-z]';
+-----+
| '2a' REGEXP '^[^a-z]' |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)

mysql> _
```

ЗАДАНИЕ.

Создать функции для:

- 1) Проверки, что название товара в базе состоит из букв, причем первая буква - согласная
- 2) Проверки, что код товара есть число
- 3) Проверки паспортных данных типа MD-77889900
- 4) Проверки, что адрес поставщика есть Минск, Москва, Киев, Петербург.
- 5) Проверки, что фамилия поставщика кончается на ов, ев, ий, ой, ова, ева, ин, ын, ина, ына.
- 6) Проверки, что в названии товара встречается сочетание букв пог, уф, баш, роч, юк.
- 7) Проверки, что в названии товара нет английских букв.