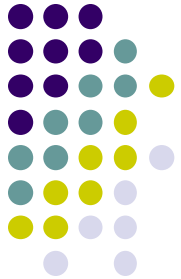


PHP и MySQL

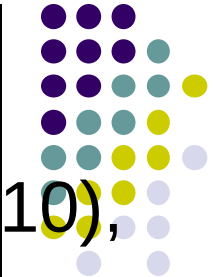
Виолета Божикова

- PHP може да работи с различни бази от данни, включително Oracle и Sybase, но основно се използва най-вече с MySQL бази от данни.
- Темата разглежда:
  - **MySQL - основни характеристики**
  - **Достъпа до MySQL Бази чрез драйвер mysql.dll**
  - **Много примери**

MySQL – Основни характеристики

- MySQL е Система за Управление на релационни БД, която предоставя достъп до данните посредством езика SQL. Синтаксисът на заявките към MySQL е почти идентичен със стандартния SQL синтаксис.
- MySQL е СУБД, която осигурява възможност за едновременен достъп до База от Данни на много потребители в мрежова среда.
- **MySQL** се появява в началото на 90-те, поради нуждата от бърза и гъвкава **база данни за целите на web базираните приложения.** Създава я Майкъл „Монти“ Видениус, под името MySQL. Тя бе дълго избор и основен компонент в широко използвания софтуерен пакет XAMPP(както и в другите „AMP“ пакети).
- От 2001 се разпространява и поддържа от шведската компания MySQL AB (основана от Дейвиз Аксмарк, Allan Larsson, Michael Widenins през 2001), която държи авторските права за голяма част от програмния код. След което става собственост на Sun Microsystems (от 2008), която добива MySQL AB. Понастоящем (от 2010), разработката и поддръжката е от корпорация Oracle, която поглъща Sun Microsystems, заедно с MySQL AB.

MariaDB – алтернатива на MySQL



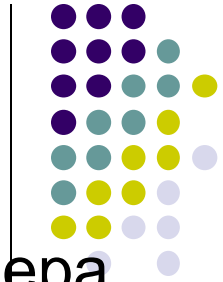
- След придобиването на MySQL от Oracle (2010), няколко от основните разработчици на базата данни MySQL, включително и Майкъл „Монти“ Видениус се оттеглят и създават MariaDB.
- **MariaDB е базирана на отворения изходен код на MySQL и е нейна алтернатива:**
  - **с повече функционалности**
  - **по-голяма сигурност и**
  - **по-добра производителност.**
- MariaDB е и типа БД при последните версии на xampp (н.р при xampp-win32-5.5.30-6-VC11-installer с версия на PHP - PHP 5.5.30).

- MariaDB е базирана на съответстващата версия на MySQL, ако съществува такава. Например MariaDB 5.1.53 е базирана на MySQL 5.1.53, с поправяне на грешките, нови функции, допълнителни storage engines, както и подобрения в производителността.
- Логично, **приликите между двете бази данни са доста**, тъй като MariaDB е базирана на отворения изходен код на MySQL и е нейно разклонение.
- Или казано накратко, ако **изтриете MySQL и инсталирате MariaDB безпроблемно ще върви всичко, което до момента е било в старата ви база данни**.
- Повече на (http://blog.icn.bg/pomoshtni-statii/mariadb-vs-mysql/)

MySQL/ MariaDB – Основни характеристики

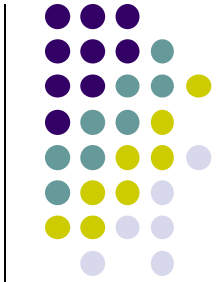
- MySQL/MariaDB е многоплатформена СУБД: има версии за най-разпространените ОС: Windows, Linux, Solaris и т.н., които могат да се изтеглят от www.mysql.com.
- Характеризира се с достатъчна скорост, надеждност и лекота в използването, особено подходяща за разработка на малки и средни по размер приложения.
- Смело може да се твърди, че понастоящем, MySQL/MariaDB е най-популярната система за управление на бази данни: поддържа програмни интерфейси (драйвери) за множество езици - C/C++, Eiffel, Java, Perl, Python, но най-често се използва в комбинация с езика PHP (За повече информация - www.php.net).
- Връзката между PHP и MySQL се осъществява чрез драйверите:
  - php_mysql.dll,
  - php_mysqli.dll и
  - php_pdo_mysql.dll.
- Тези драйвери на практика управляват достъпа до MySQL Бази от данни.

Драйвери на PHP за операции с MySQL/ MariaDB БД



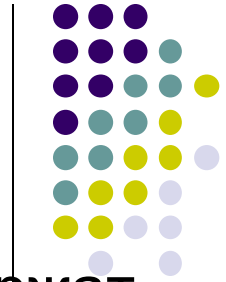
- В дистрибицията на PHP за Windows, 3 – те драйвера php_mysql.dll, php_mysqli.dll и php_pdo_mysql.dll (библиотеки на PHP за операции с MySQL/MariaDB БД) са в папка ext на php, тоест това са:
  - **...\ext\php_mysql.dll,**
  - **...\ext\php_ mysqli.dll и**
  - **...\ext\php_pdo_mysql.dll**
- **Драйверите mysqli** (т.е **mysql improved**) и **pdo (PHP Data Objects)** са изцяло разработени за 5-та версия на PHP - за да се предостави на програмистите достъп до новите възможности след MySQL 4.1.

Какво е MySQLi :



- php_mysqli.dll (mysql improved/подобрен mysql) се явява подобрена версия на стария драйвер php_mysql.dll.
- **Защо е бил създаден?** Защото php_mysql се базира на поддръжката на функционала MySQL 4.1.3, не поддържа транзакции, няма обектен интерфейс, налице е уязвимост към подмяна за стойности в заявката.
- Освен процедурен интерфейс, mysqli предоставя и ОО интерфейс.
- PDO предоставя само ОО интерфейс.

Кой драйвер да използваме, ако ни трябва ОО интерфейс?



- И двата - php_mysqli.dll и php_pdo_mysql.dll поддържат Prepared Statements, които предпазват от SQL injection, и са много важни за **сигурността** на web приложенията.
- Процедурният интерфейс, осигуряван чрез php_mysqli.dll е аналогичен на този на php_mysql.dll, тоест много лесно може да се реновира едно приложение, използващо php_mysql.dll.

Но:

- PDO разработи с 12 различни БД, докато MySQLi – само с MySQL БД.
- Тоест, ако вашият проект трябва да използва различни видове БД, то изберете PDO. Така ще трябва да промените само connection string-а си, а с MySQLi – трябва да пренапишете кода си.

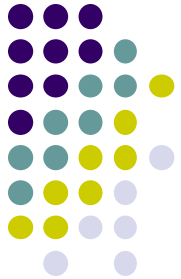
SQLite към MySQL/MariaDB БД

- Операциите с MySQL/MariaDB се изпълняват по същия сценарий като операциите с SQLite БД:
  - създаване на връзка,
  - изпълнение на операции и
  - затваряне на връзка.
- Има обаче една съществена разлика, която се състои в това, че при операции с MySQL се създава връзка между клиент (PHP модул) и сървър (MySQL сървър), което изисква задаване на допълнителни параметри на връзката, а с SQLite – няма сървърен процес.
- Твърди се, че ако web приложението има по - малко от 100,000 посетителя на ден, SQLite е по – добрия избор: 10 пъти е по-бърз е от MySQL и неговия графичен клиент PhpLiteAdmin е много по-елементарен и бърз от PHPMyAdmin за

MySQL/ MariaDB – Клиентски програми:

- Съществуват различни клиентски програми, които се използват за осъществяване на връзка със сървъра за MySQL/MariaDB БД. Например:
  - MySQL Query Browser,
  - MySQL Administrator,
  - MySQL WorkBench,
  - MySql Front,
  - **phpMyAdmin** и др.
- Това са клиентски програми, които осигуряват графичен интерфейс за връзка с MySQL СУБД.
- Чрез тях можем да създаваме БД и таблици в тях, да задаваме потребители и права, да въвеждаме данни и използване данните в БД.

MySQL/ MariaDB – Клиентски програми

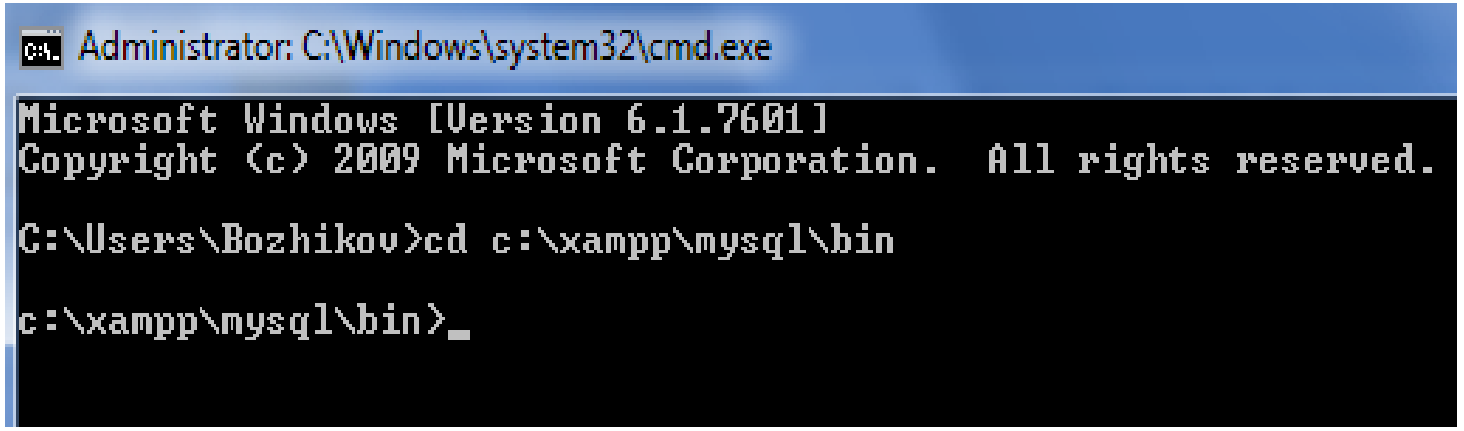


- phpMyAdmin е един много популярен графичен интерфейс (написан на PHP) за работа с MySQL. Работата с phpMyAdmin е удобна и интуитивна, това е програма, включена в пакета **xampp.**
- Със СУБД **MySQL/MariaDB** се разпространява и вграден **MySQL/MariaDB конзолен клиент**. При пакета **xampp,** това е програма **mysql.exe**, която се намира в **папка bin** на папка **mysql**, с която се работи в команден режим.
- Следващите екрани визуализират резултати, свързани с използването на **MySQL конзолен клиент** (под ОС Windows 7, при използване на пакета xampp, инсталиран в конкретния случай в **C:\xampp**).
- И така:

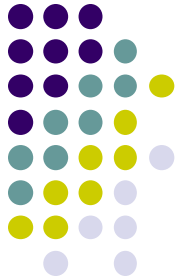
Стартирането на MySQL/ MariaDB – клиент:

- Стартирането на MySQL клиент става по следния начин:
  - Start бутон  : cmd

  - Позиционираме се в папка bin на mysql, тоест C:\xampp\mysql\bin



Създаване на връзка със сървъра за MySQL/ MariaDB :



- За създаване на връзка със **сървъра за** MySQL/MariaDB БД, е необходимо да се зададат: **host , port, protocol, shared-memory-base-name**
- **Параметър host**: задава IP адреса на компютъра, на който е инсталиран MySQL/MariaDB сървър.
- Ако MySQL/MariaDB сървър е инсталиран на същия компютър, то адреса е localhost или **127.0.0.1:**

-- host=127.0.0.1, или -- host=localhost

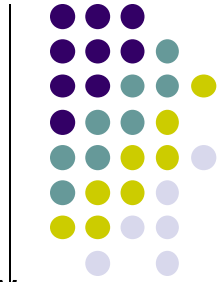
-- host= IP адреса, (дълък формат)

Или

–h 127.0.0.1

–h IP адрес, (кратък формат)

Създаване на връзка със сървъра за MySQL/ MariaDB :



- **Параметър port** – задава номера на порта, към който е свързван компютъра, на който е инсталиран MySQL сървър. Подразбиращият се порт е **3306** (за TCP/IP протокол).
- --port=номер_на_ порт, н.р --port=3306

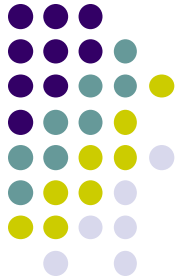
или

- -p номер_на_ порт

Ето как може да изглежда в ОС Windows командния ред за връзка със сървъра за MySQL/MariaDB БД, при задаване на параметри host и port и protokol:

C:\>mysql -- protocol=tcp --host=127.0.0.1 --port=3306

Създаване на връзка със сървъра за MySQL/ MariaDB :



- **Параметър protocol** – задава протокола, по който се осъществява връзката със сървъра.

--protocol=име_на_протокол

- Възможни стойности са:
  - tcp – за **TCP/IP** (подразбиращ се протокол);
  - pipe – за връзка по именуван канал към локален сървър (само за Windows OS);
  - memory – за връзка към локален сървър със споделена памет (само за Windows OS);
  - socket – за връзка чрез Unix сокет (само за Unix OS).

Създаване на връзка със сървъра за MySQL/ MariaDB:

- **Параметър shared-memory-base-name:** задава име на споделен блок памет (само за Windows OS и само за протокол тип memory):

-- shared-memory-base-name=име

Пример:

Ето как може да изглежда в ОС Windows командния ред за връзка със сървъра за MySQL/MariaDB БД, при задаване на параметри host и port и protokol:

C:\>mysql -- protocol=tcp --host=127.0.0.1 -- port=3306

Идентификация на потребител (**-u**)

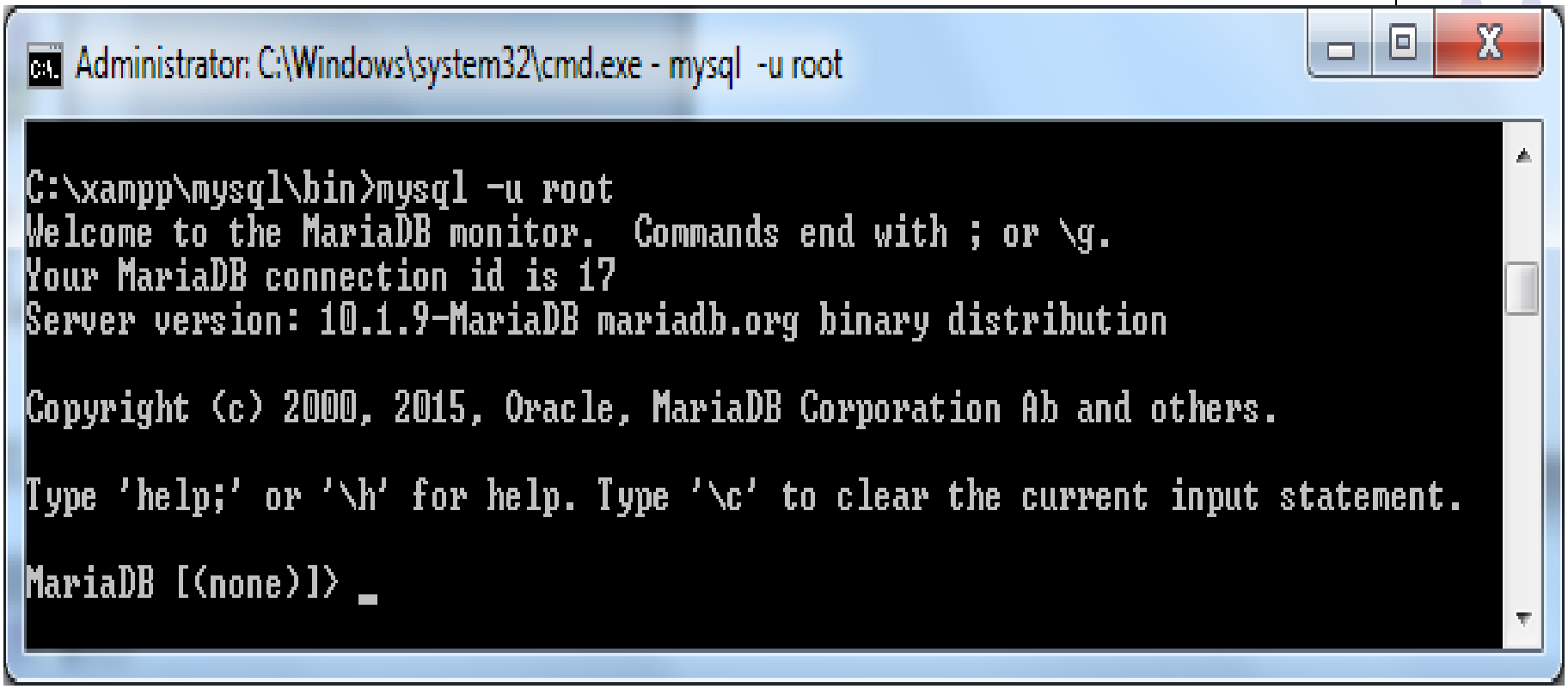
- При инсталиране на MySQL сървър, инсталиращата програма xampp (PHP Версия: 5.5.30) създава потребител **root** без парола, с глобални привилегии.
- Достъпът до MySql/MariaDB сървър за този потребител, използвайки вградения конзолен клиент (mysql.exe) става по следния начин:

Забележка: при инсталация на xampp-win32-5.5.30-6-VC11 с PHP Version 5.5.30 имаме:

cmd
**cd **
cd C:\xampp\mysql\bin
C:\xampp\mysql\bin>mysql -u root
Появява се промт-а: **MariaDB [<none>]>**

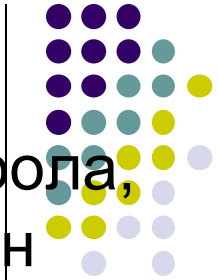

XAMPP Installers and Dow...
XAMPP Download Success
https://www.apachefriends.org/index.html
Search
Apache Friends
Download
Add-ons
Hosting
Community
About
Search..
Search
EN
XAMPP Apache + MariaDB + PHP + Perl
What is XAMPP?
XAMPP is the most popular PHP development environment
XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.
XAMPP
Download
Click here for other versions
XAMPP for Windows
v5.6.15 (PHP 5.6.15)
XAMPP for Linux
v5.6.15 (PHP 5.6.15)
XAMPP for OS X
v5.6.15 (PHP 5.6.15)

При инсталиране на MySQL/MariaDB сървър (чрез Xampp), инсталиращата програма създава **потребител root без парола**. Достъп до MySql/MariaDB сървър (от команден ред) за user=root, без парола (No password):



Забележка: *Всички команди оттук нататък са валидни и за двете БД, съответно - за MySql/MariaDB сървър!*

Идентификация на потребител



- Не се препоръчва създаването на аккаунти без парола, защото това създава предпоставки за неоторизиран достъп.
- Ето общия вид на командният ред при зададени параметри за: user и password.

...mysql\bin>mysql -u someus -p somepass

...mysql\bin>mysql --u=someus --p=somepass

- Параметри user и password определят съответно потребителското име и паролата на акаунта, които ще се използват за достъп до MySQL сървъра.

Пример: ако имаме **user „nina“** и **password „123“**

C:\xampp\mysql\bin>mysql –u nina –p

Enter password: 123

Появява се промпт-а: **MariaDB [<none>]>**

Идентификация на потребител

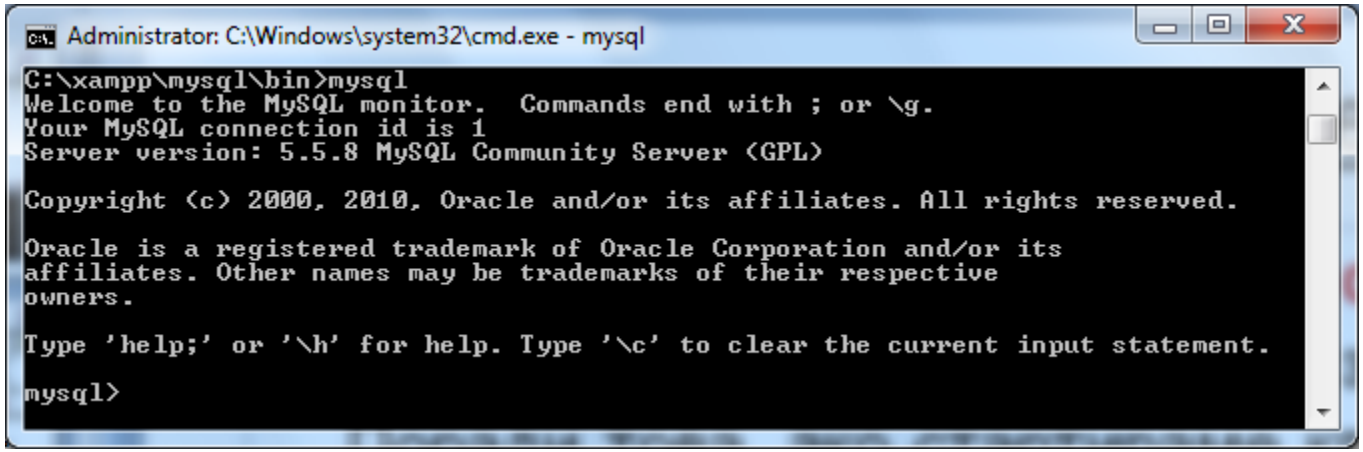
- Таблицата user (тоест, mysql.user), която се създава по подразбиране в БД mysql на сървъра (MySql/MariaDB) съдържа всички регистрирани потребители.
- Връзката (с MySql/MariaDB сървър) ще се осъществи само, ако в таблицата mysql.user има регистриран потребител с такова име и парола.

За xampp, БД mysql е в папка C:\xampp\mysql\data.

- Забележка: Сървърът използва стойността на параметъра user в комбинация с host, н.р root@localhost. Това означава, че може да има аккаунти с едно и също потребителско име user, но с различни стойности на host.

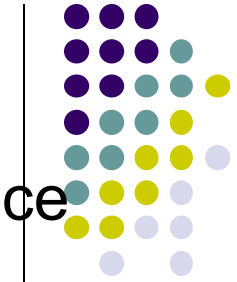
Идентификация на потребител

- За Windows ОС, **подразбиращото се име за потребител** към MySQL сървър е **ODBC**, а за **host – localhost**. Поради това, ако стартираме клиентската програма (**mysql.exe**) без параметри, то MySQL ще създаде връзка: **ODBC@localhost**.



- **Не съществува подразбираща се парола**. Може обаче този параметър да не се зададе (както бе показано в примера за user=root) и тогава в **mysql.user** трябва да имаме регистриран аккаунт със зададеното име на user, позволяващ свързване без парола.

Идентификация на потребител



- Съдържанието на таблица **mysql.user** може да се изведе с командата:

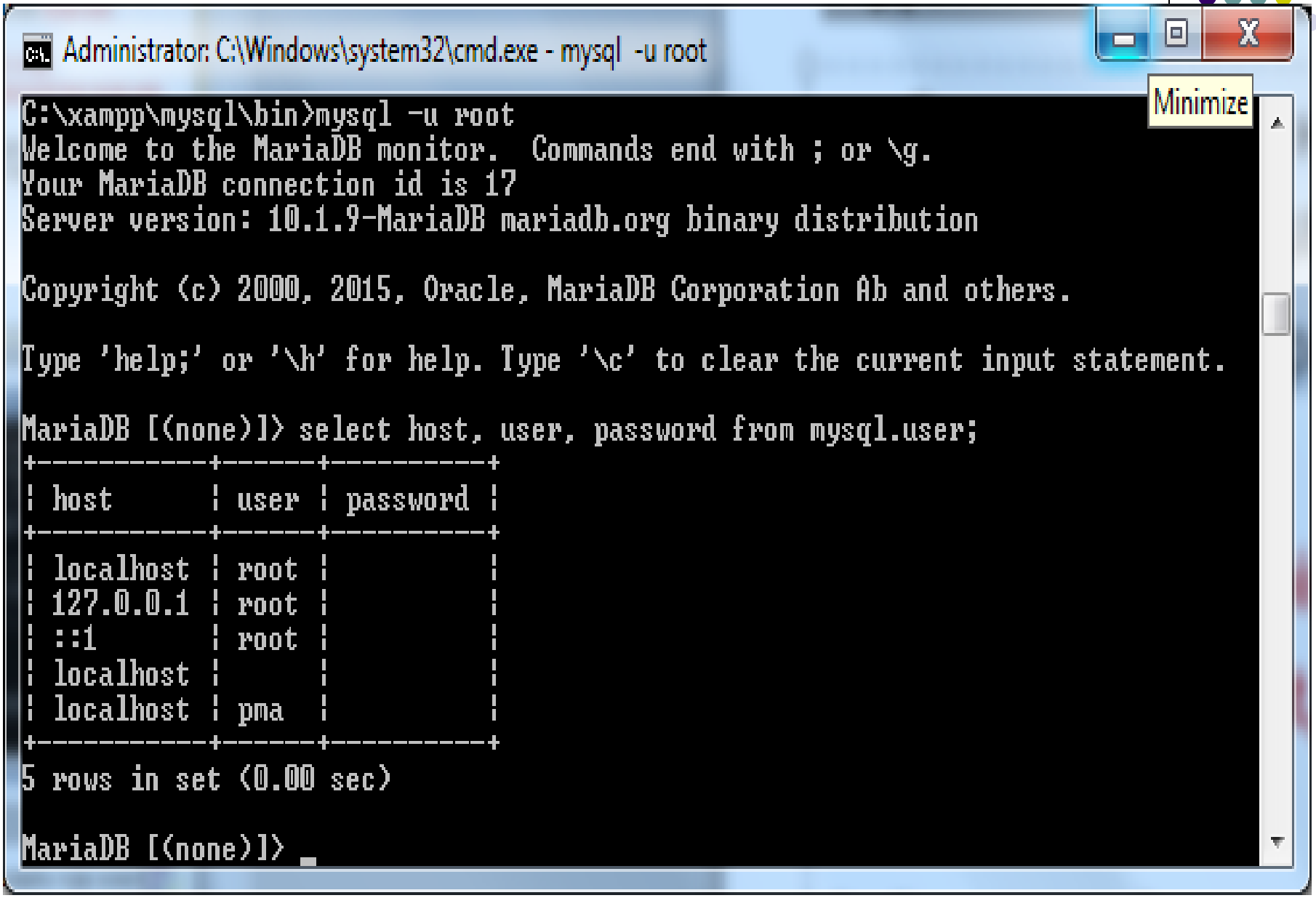
select host, user, password from mysql.user;

- **Пример:**

C:\xampp\mysql\bin> mysql -u root

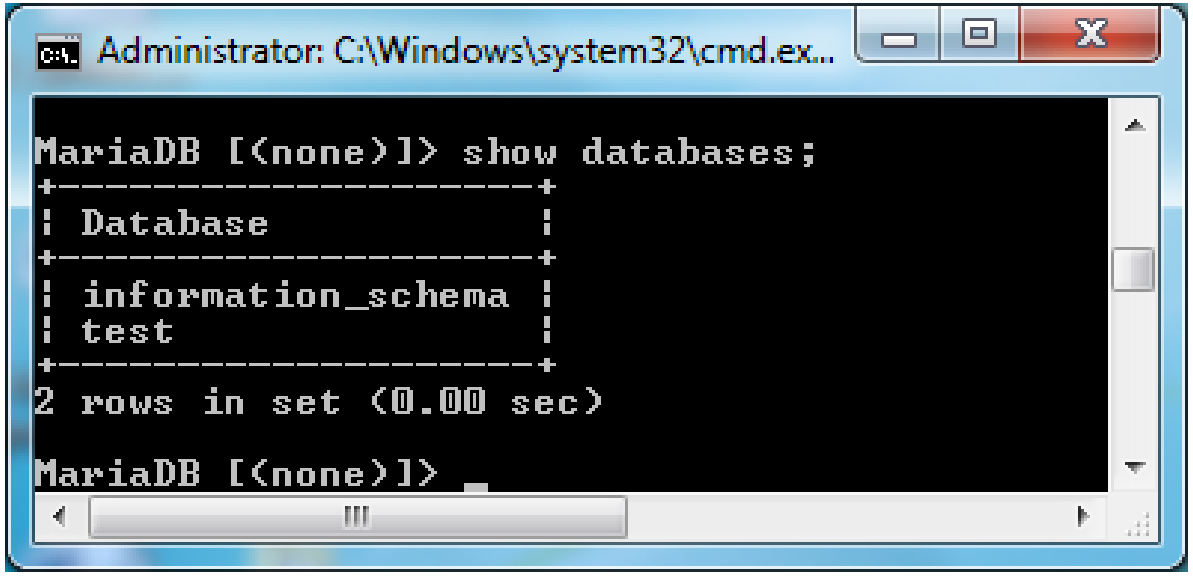
MariaDB [<none>]>select host, user, password from mysql.user;

Забележка: За да се стартира тази команда, аккаунта трябва да има съответни привилегии!


Administrator: C:\Windows\system32\cmd.exe - mysql -u root
Minimize
C:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 17
Server version: 10.1.9-MariaDB mariadb.org binary distribution
Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> select host, user, password from mysql.user;
+-----------+------+----------+
| host | user | password |
+-----------+------+----------+
localhost	root	
127.0.0.1	root	
::1	root	
localhost		
localhost	pma	
+-----------+------+----------+
5 rows in set (0.00 sec)
MariaDB [(none)]>

Да припомним няколко SQL команди, използвайки mysql клиент:

- Команда **show databases** - извежда всички бази от данни, до които въпросният аккаунт има достъп:



Още SQL команди:

- Командата **create име_база_от_данни** създава база от данни с указаното име:

Пример:

MariaDB [<none>]> create database test_db;

- Командата use име_база_от_данни задава подразбираща се база от данни с указаното име:

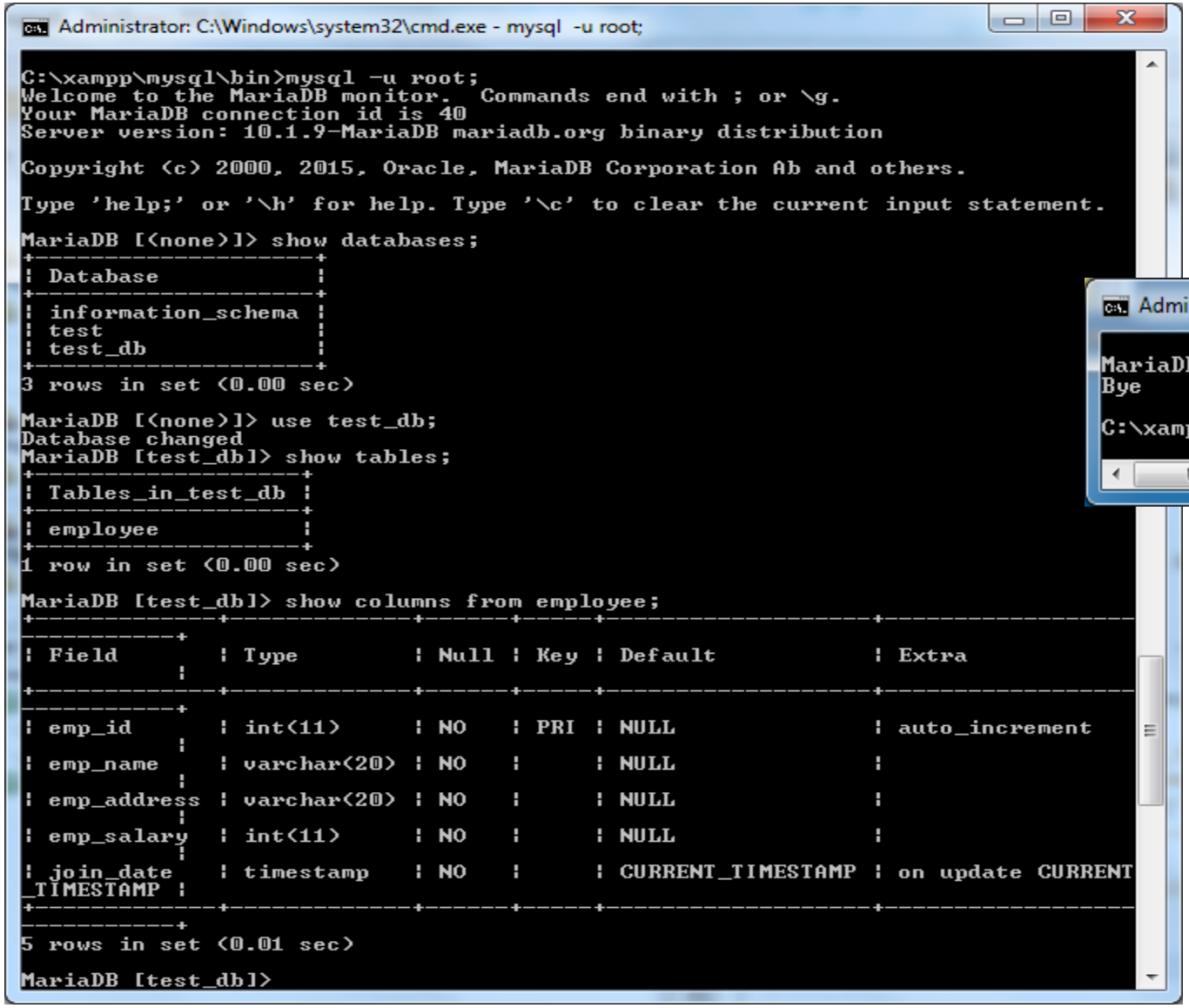
Пример: **MariaDB [<none>]> use test_db;**

MariaDB [<none>]>show tables;

- Командата **show tables** извежда списък с достъпните таблици на подразбиращата се БД, а
- Командата **show columns from име_на_таблица** – извежда колоните на избрана таблица:

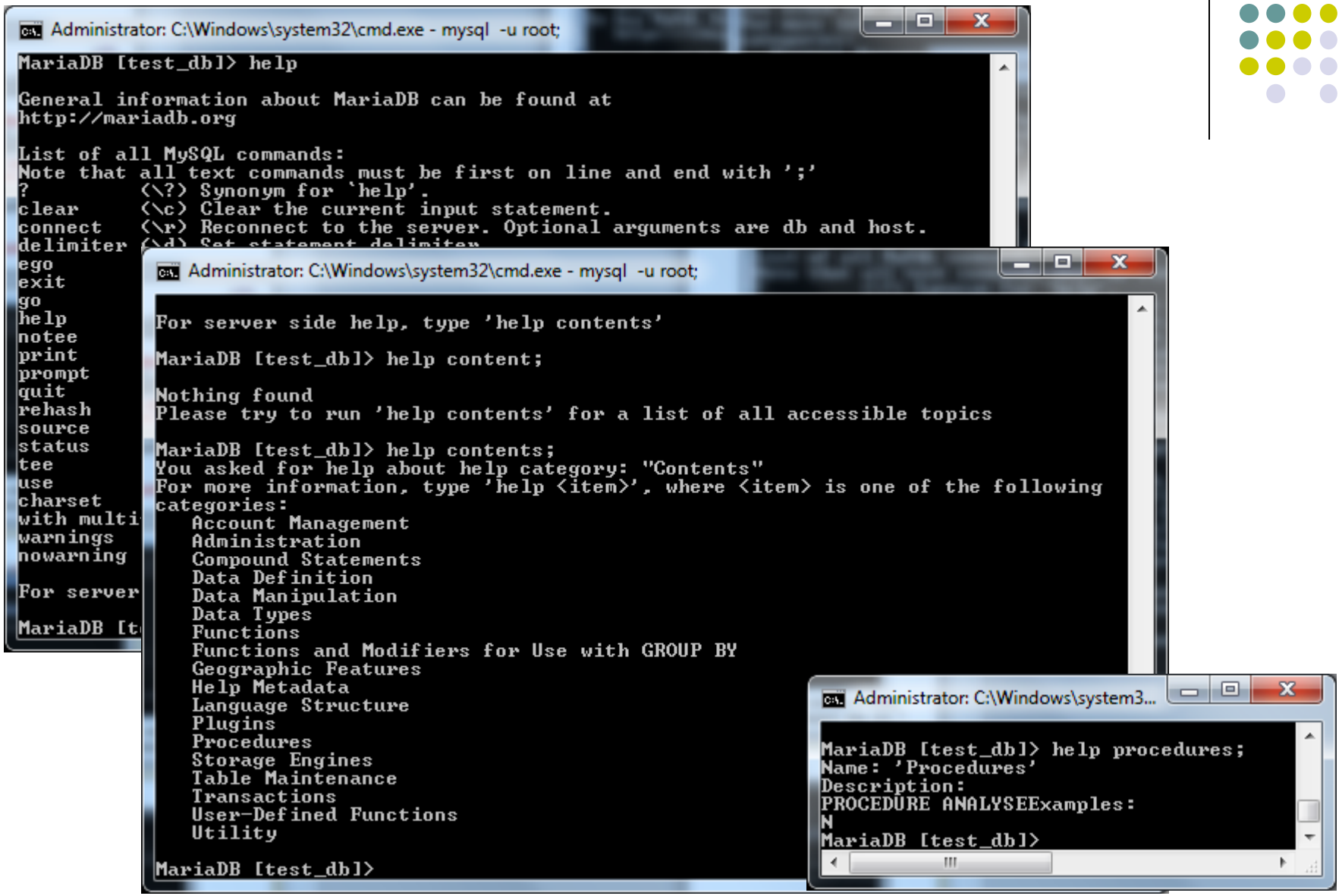
Пример:

MariaDB [<none>]> show columns from employee;



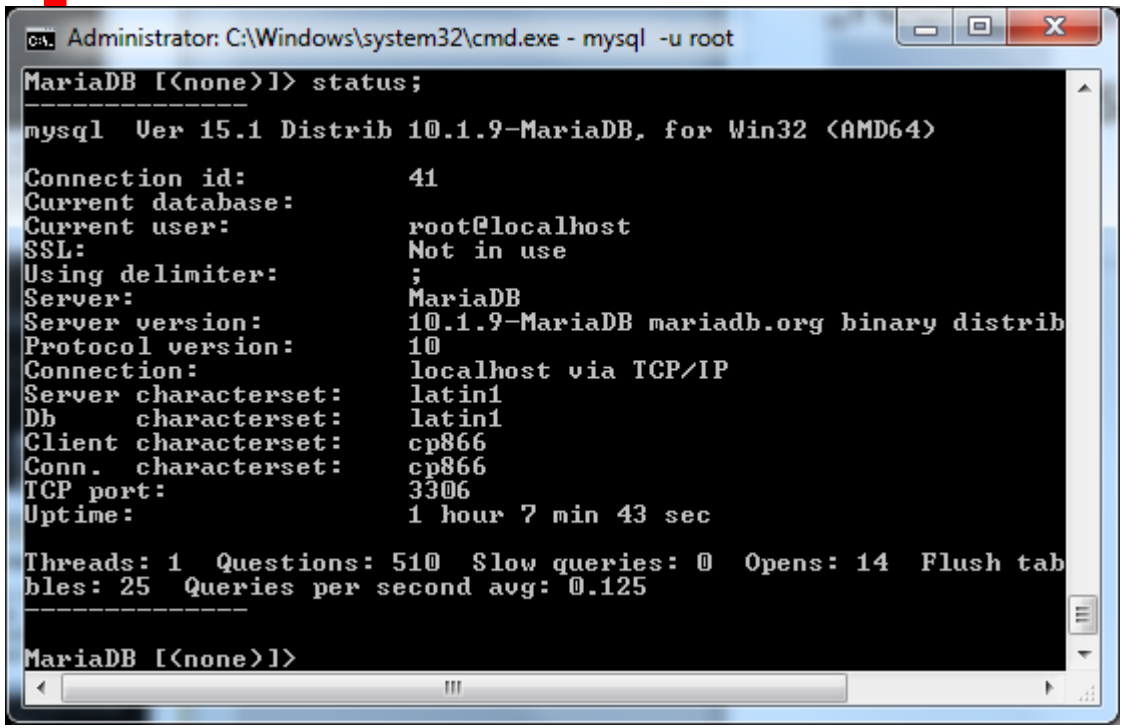
- С командите **exit** (Ctrl - C) и **quit** се прекъсва връзката с mysql/MariaDB сървър.
- Помощ се получава с комадата **help** или **?**.

Help, help contents, help <item>:

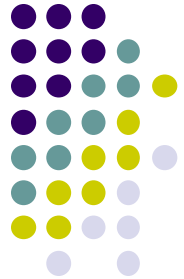


Извеждане параметрите на връзката

- Параметрите на връзката, версията, потребителите и други се извеждат с команда status. И така, за потребител „root@localhost”, No password (без парола):
- **MariaDB [<none>] >status**



Администриране на потребител



- Администрирането на потребител включва:
  - **създаване на потребител**

  и
  - **задаване на права на потребителя**.
- Създаването на потребител става със SQL командата **CREATE USER**.
  - Тази команда може да бъде изпълнена от потребител с **глобални привилегии** или с привилегии **INSERT за БД.**
  - Тази команда създава нов запис в таблицата **mysql.user**, който запис обаче не включва задаване на привилегии. Ако аккаунта вече съществува, то се генерира грешка.

Създаване на потребител

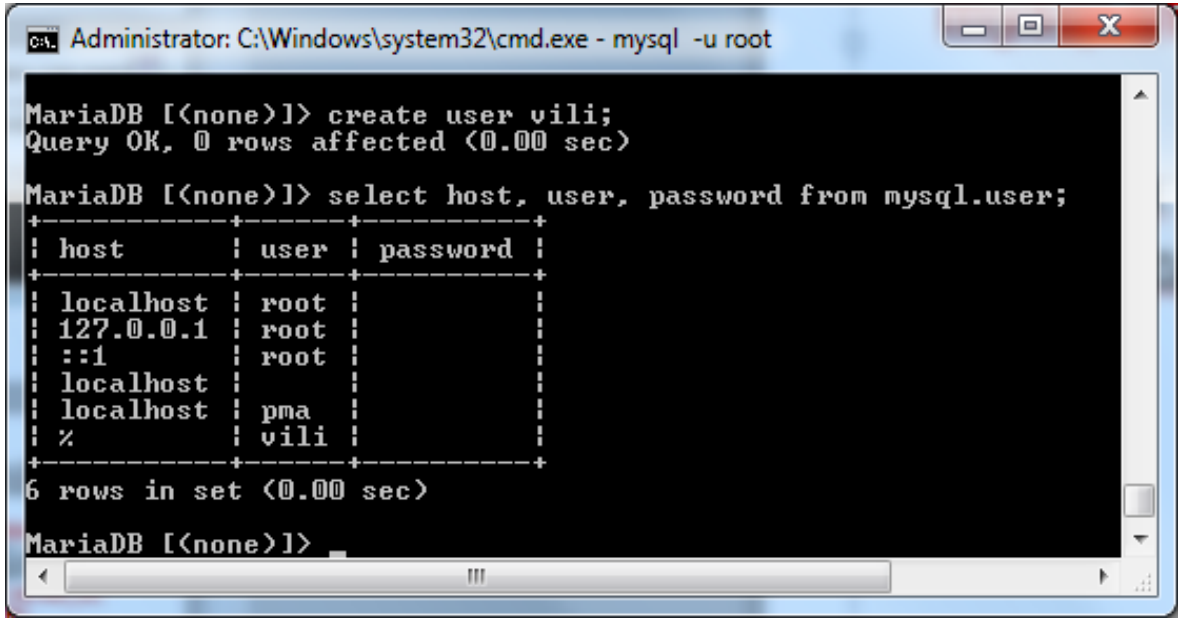
- Създаване на потребител (От root@localhost):

CREATE USER 'user_name'@'host_name'
[IDENTIFIED BY 'password']

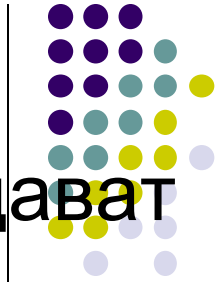
- Пример: Създаване на потребител vili без парола.

MariaDB [<none>]> create user vili;

- Без клаузата IDENTIFY се създава потребител vili без парола



Задаване на права на потребител



- Права/привилегии на потребител се задават с командата:

GRANT priv_type [(column_list)]

[,priv_type [(column_list)]]…

ON [object_type] priv_lavel

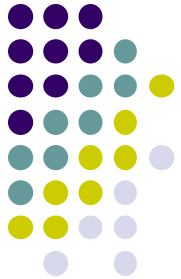
TO user_name [IDENTIFIED BY ‘password’]

[REQUIRE NONE | [{SSL | X509}]]

[WITH [GRANT OPTION|MAX QUERIES PER HOUR count|…]]

- Където:

Задаване на права на потребител



- priv_type – типа на задаваните привилегии, например:
  - **ALL** – всички привилегии без привилегията „задаване права на потребители” – тоест без WITH **GRANT OPTION**.
  - **SELECT**
  - **INSERT**
  - **UPDATE**
  - **CREATE**
  - **ALTER** (променя структурата на таблица)
  - **Execute** – изпълнение на съхранени процедури.
  - **LOCK TABLES** – заключване на таблици

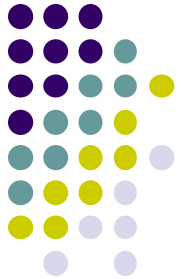
Задаване на права на потребител

- column_list – списък от полета, за които важат привилегиите.
- [object_type] **priv_lavel** – задава се обекта от БД (TABLE, FUNCTION, PROCEDURE), за който се задават привилегии и нивото на привилегия priv_lavel. Имаме:
  - Привилегия на глобално ниво, тоест ***** - означава за всички БД на сървъра. С команда REVOKE ALL ON *.* премахваме само този вид привилегии.
  - Привилегия на ниво на БД: указва се като **db_name.*.** Премахването на тази привилегия става с команда: REVOKE ALL ON db_name.*.
  - Привилегия на ниво на таблица от БД: указва се като **db_name.table_name**. Премахването на тази привилегия става с команда: REVOKE ALL ON db_name.table_name.
  - Привилегия на ниво на полета от таблица в БД: полетата се задават **column_list**, а таблицата се указва **db_name.table_name**. Тези привилегии се записват в таблицата mysql.column_priv.

Задаване на права на потребител

- TO user_name [IDENTIFIED BY ‘password’] – задава потребителя
- REQUIRE – задава допълнителни изисквания: NONE – означава, че не изисква криптирани връзки към аккаунта, SSL – разрешава само криптирани връзки, | X509 – изисква клиентът да притежава сертификат ‘cipher’.
- WITH - задава допълнителни опции, които магат да бъдат например: GRANT OPTION – право за задаване на права на потребител, MAX QUERIES PER HOUR count максимален брой заявки, които се обслужват за потребителя за един час и др.
- **Забележка**: тази команда не само задава права на потребител, но и създава потребителя, ако той не съществува. Така също – може да задава или променя паролата му.

Задаване на права на потребител



- Примери:

GRANT SELECT, INSERT, DELETE ON db1.* TO student@localhost IDENTIFIED BY abc;

- Задаване чрез команда GRANT на привилегии SELECT, INSERT, DELETE към db1.*, на потребител student@localhost с парола abc.

GRANT ALL ON *.* TO teacher@localhost;

- ON *.* - задава глобална привилегия на teacher@localhost.
- IDENTIFIED BY ‘pass’ – незадължителна, ако е включена присвоява парола на новосъздаден account или променя паролата на съществуващия.

Показване на права на потребител

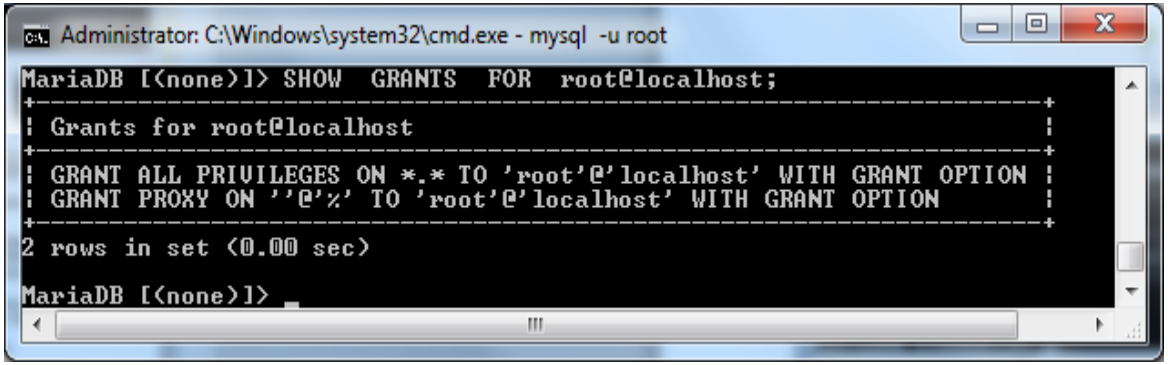
- За собствения : **SHOW GRANT**, в случая за **root@localhost:**
- За кой да е (например за ana@localhost или за за vili@%:

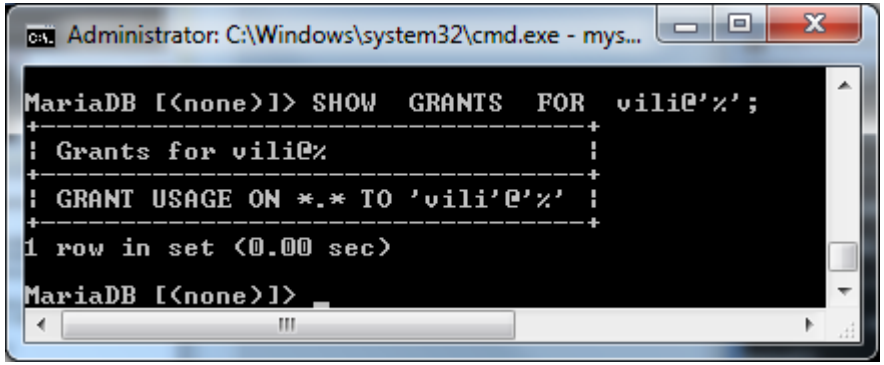
SHOW GRANTS FOR 'user_name'@'host_name';

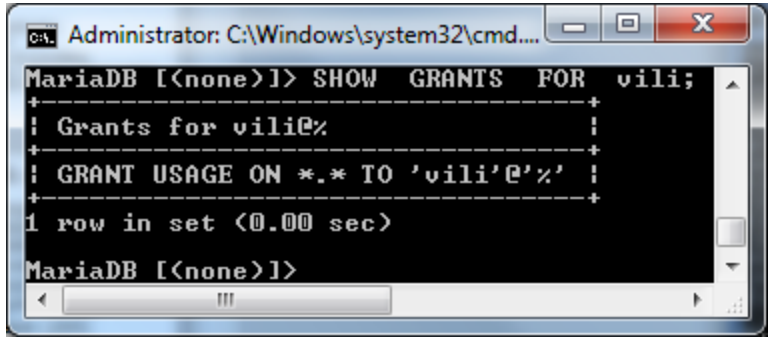
- Когато потребителят е без хост, то пишем %, например vili@% или само: **SHOW GRANTS FOR vili;**

Отменяне на права на потребител

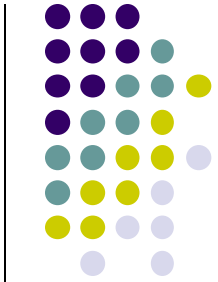
- REVOKE - списък ON ниво на привилегии FROM account.







За синтаксиса на SQL операторите в MySQl/ MariaDB СУБД:



- Всеки SQL оператор завършва с ;
- SQL операторите могат да се разполагат на произволен брой редове. Символът “\c” служи за отмяна на започната заявка
- При въвеждане на функция не се допуска интервал между името на функцията и скобата
- Операторите и имената на функциите могат да се изписват както с малки така и с големи букви, тоест СУБД не прави разлика.
- Пълното име на таблица в MySQl/ MariaDB е database_name.table_name
- Пълното име на атрибут в MySQl/ MariaDB таблица е database_name.table_name.field_name
- MySQl/ MariaDB поддържа всички оператори за създаване, модифициране и изтриване на данни, таблици и бази от данни: SELECT, CREATE, INSERT, DELETE, REPLACE, ALTER, DROP

Още за синтаксиса на SQL операторите...

- **MySQl/ MariaDB** поддържа **LIMIT** клауза (поставя се в края, след ORDER BY), например: **select * from mytable limit 0,50** – извлича първите 50 записа от резултата, като номерацията е от 0.
- **MySQl/ MariaDB** поддържа GROOP BY и ORDER BY клаузите, както и групиращите функции
- Поддържа външни свързвания (JOIN)
- Поддържа транзакции и заключване на таблици
- За получаване на списък на базите от данни, както показахме и по-напред, се използва оператор: **SHOW databases**

Задаване на кодиране на ниво БД, таблица и колона (CHARACTER SET):

- При създаване на базата от данни, на таблица на БД и колона на таблица може да се посочи и в какъв код да бъдат съхранени данните (latin1, Windows-1251, UTF- 8 или друг). Това става чрез клаузата CHARACTER SET, която се поставя на последно място в оператор CREATE:

CREATE DATABASE <име на БД>

CHARACTER SET <име на кодова таблица>;

Пример:

MariaDB [<none>]> CREATE DATABASE universitet

CHARACTER SET cp1251;

- **Задаване кодиране на ниво таблица:**

CREATE TABLE <име на таблица> (...) CHARACTER SET <име на кодова таблица>;

Пример:

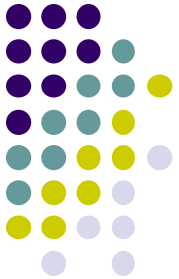
MariaDB [<none>]> use universitet;

MariaDB [<none>]> CREATE TABLE specialnosti (id int unsigned not null auto_increment, ime varchar(100) not null, primary key(id))

CHARACTER SET utf8;

Може и чрез: DEFAULT CHARSET=utf8;

Задаване на кодиране на ниво БД и таблица (CHARSET=...):

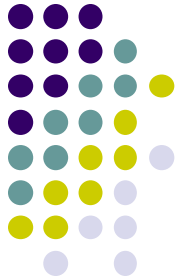


```
mysql_select_db('university');
CREATE TABLE students(
  id INT(10) NOT NULL,
  name VARCHAR(32) DEFAULT NULL,
  city VARCHAR(30) DEFAULT NULL,
  age SMALLINT(6) DEFAULT NULL,
  PRIMARY KEY  (id))
ENGINE=INNODB DEFAULT CHARSET=utf8;
```

Или:

```
CREATE TABLE students(id int(10) NOT NULL
   AUTO_INCREMENT,...)
ENGINE=MyISAM  DEFAULT CHARSET=latin1
   AUTO_INCREMENT=6, AVG_ROW_LENGTH=124 ;
```

Задаване на кодиране на ниво поле (character SET ...):



```
CREATE TABLE ae_gallery (
  id int(11) NOT NULL AUTO_INCREMENT,
  title varchar(64) character SET utf8 NOT NULL,
  ext varchar(8) character SET utf8 NOT NULL,
  ...
 data blob NOT NULL,
  PRIMARY KEY  (`id`)
);
```

Синтаксис на командата за създаване на таблица в MYSQL бази от данни

- Таблиците в MySql бази от данни се създават с SQL командата CREATE TABLE. Оператор CREATE TABLE създава таблица с зададено име в текущата БД.
- Ако няма активна текуща база от данни или указаната таблица вече съществува, то възниква грешка.
- Командата има следния синтаксис:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
[Име_на_БД].име_на_таблица[(дефиниция_на_колона,...)]
[машина_за_съхранение] [select_израз]

- С изрази „дефиниция_на_колона” се изброяват колоните, които трябва да съдържа таблицата.
- Формалното описание на „дефиниция_на_колона” изглежда така:

Име_на_стълб тип [NOT NULL | NULL] [DEFAULT стойност_по_подразбиране] [AUTO_INCREMENT] [UNIQUE [KEY]]|[PRIMARY KEY] [reference_definition]

Тип на стълб (тип в израза дефиниция_на_колона) може да бъде един от възможните MySQL типове.

За тип на стълб ...
при създаване на таблици в MYSQL бази от данни

MySQL поддържа всички типове данни, характерни за другите СУБД, но имената на някои от типовете са различни. С команда **help data types** (както показме по-напред) може да бъде изведен пълния списък.

Малко по-подробно, за целочислените типове:

TINYINT	**1 байт**	**- 128 до 127**
SMALLINT	**2 байта**	**- 2^{15} до 2^{15}**
MEDIUMINT	**3 байта**	**- 2^{15} до 2^{15}**
INT	**4 байта**	
BIGINT	**8 байта**	

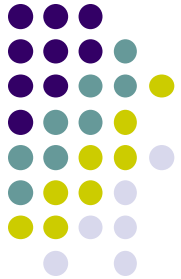
В кръгли скоби след типа може да се зададат и:

- брой позиции за показване, например zipcode smallint(5).
- спецификатори [UNSIGNED] и [ZEROFILL]: например INT [(length)] [UNSIGNED] [ZEROFILL] (**ZEROFILL – стойностите на колоната се допълват с водещи нули**), например **x INT(8) ZEROFILL NOT NULL**. Това означава, че стойността на **x** ще се покаже с не по-малко от 8 позиции, като незначещите нули в началото се показват с 0.

Тип **BIT(n)**, където n – брой битове, също може да се определи като целочислен тип. Той е най-икономичен. Например: flag BIT(1) – задаваме поле flag с размер само един бит.

За тип на стълб ...

при създаване на таблици в MYSQL бази от данни



- Реален тип: **FLOAT, DOUBLE** [(length,decimals)] [UNSIGNED] [ZEROFILL]
- Символен тип: с фиксирана дължина - **CHAR**(length) [BINARY] и с променлива дължина - **VARCHAR**(length) [BINARY] (VARCHAR - недвоичен, символен тип), **VARBINARY** - двоичен, символен тип.
- Дата: **DATE** и за дата и време: **DATETIME**, **TIMESTAMP**
- за работа с големи обекти: **BLOB** (на http://www.anyexample.com/programming/php/php_mysql_example__image_gallery_(blob_storage).xml е показан един хубав пример – фото галерия)
- текстов: **TEXT**
- Изброяване и множество: **ENUM**(value1,value2,value3,...) и **SET**(value1,value2,value3,...).

Пример: country ENUM (‘BG’, ‘FR’,’EN’); color SET (‘red’,’blue’,’pink’);
Декларира се поле ENUM със стойности: ‘BG’, ‘FR’,’EN’ и NULL, тъй като полето не е декларирано с ограничение NOT NULL.

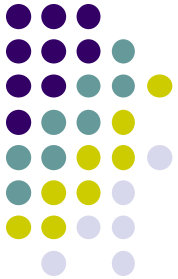
За тип на стълб ...

при създаване на таблици в MYSQL бази от данни

- Всеки стълб може **да няма стойност** (**NULL по подразбиране**), да бъде **NOT NULL,** да има стойност по подразбиране, да бъде ключ или автоматично да нараства (**AUTO_INCREMENT**).
- Ако полето е AUTO_INCREMENT, то неговата стойност автоматично се увеличава с 1 всеки път при добавяне на данни в таблицата. Автоинкремент в таблицата може да има **само един** и при това – обезателно е индексиран (**PRIMARY KEY** или **UNIQUE**). AUTO_INCREMENT започва от 1. **Наличието автоинкремент е една от особеностите на MySQL**.

Пример: **id INT PRIMARY KEY AUTO_INCREMENT**

- Вместо изброяване на колоните и техните свойства в дефиниция_на_колона може да се задават и списъци на ключовите и индексни полета, ограничения и проверки:



- **PRIMARY KEY** (име_на_индексируемия_стълб, ...)

Полето трябва да има уникална стойност!!! Не се допуска NULL.

Например:

```
CREATE TABLE Orders
    (O_Id int NOT NULL, P_Id int,
    PRIMARY KEY (O_Id),
    FOREIGN KEY (P_Id) REFERENCES Persons(P_Id)
    )
```

- **FOREIGN KEY(**име_на_индексируемия_стълб...). Забележете, че колона "P_Id" на таблица "Orders“ сочи "P_Id" колона на таблица "Persons".
- **KEY** (име_на_индексируемия_стълб,...)

Или синонима на KEY:

- **INDEX** (име_на_индексируемия_стълб,...)

```
CREATE TABLE if not exists my_db.tab2  (name VARCHAR(20), fnom VARCHAR(20),
    index(fnom));
```

- **UNIQUE** [INDEX] (име_на_индексируемия_стълб,...)

Полето трябва да има уникална стойност!!!

```
CREATE TABLE Persons ( P_Id int NOT NULL, ..., UNIQUE (P_Id))
```

Или (ако таблицата е вече създадена): **ALTER TABLE Persons ADD UNIQUE (P_Id)**

- **FULLTEXT** [INDEX] (име_на_индексируемия_стълб,...)

Н.р: **ALTER TABLE news ADD FULLTEXT(headline, story);**

- При задаване на всички тези елементи се указва списъка на полетата (стълбовете), които ще влязат в индекса;
- Освен изброените неща, при създаване на таблица могат да се укажат и:
  - Задаване на начална стойност на брояча за автоинкремент (различна от 1): **AUTO_INCREMENT = число**, например:

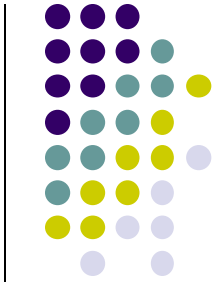
...)ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=6, AVG_ROW_LENGTH=124 ;

  - средна дължина на редовете на таблицата: **AVG_ROW_LENGTH = число**
  - коментари в таблицата (ред с дължина от 60 символав): **COMMENT = "ред“**

ALTER TABLE myTable CHANGE COLUMN myColumn myColumn BIGINT NOT NULL COMMENT 'This is the most important primary key ever';

  - максимално и минимално предполагаемо число на редовете: **MAX_ROWS = число** и **MIN_ROWS = число**
  - машина_за_съхранение (**ENGINE**):

Още за таблиците в MYSQL бази от данни (ENGINE)



- Машината за съхранение на данните (**ENGINE**) зависи от типа на таблицата, например **ENGINE = InnoDB е за таблици тип InnoDB**.
- Типът на таблицата може да е различен: **MYISAM, InnoDB** и др.
  - Тип **MYISAM** – това е енджина по default за MySQL. Ако повечето заявки към таблицата ще са SELECT, то този енджин е за предпочитане (т.е ако трябва бързо четене на данни, **MYISAM** е подходящия тип таблица). Заема малко памет, но **не поддържа транзакции**, заключването е на ниво таблица. Безплатните хостинги предлагат винаги MyISAM, за разлика от InnoDB енджин, който заема повече място. По всяко време може да се мигрира от MyISAM енджин към InnoDB, но обратната миграция е опасна, ако имаме Foreign Keys и транзакции…
  - **InnoDB** е по-нов енджин за MySQL и има предимства пред MyISAM. Поддържа **транзакции** и **заключване на ниво запис.** Това е добрият избор ако в приложението ще има много INSERT/UPDATE SQL заявки, ако има нужда от Foreign Keys support (т.е InnoDB поддържа и външни ключове за разлика от MyISAM). Заема повече памет от MyISAM.
  - **XtraDB** заменя InnoDB в MariaDB и е напълно съвместим с InnoDB, т.е «ENGINE = InnoDB» при MariaDB на практика означава тип XtraDB. Това е **оптимизирана по отношение на производителност** подсистема за съхранение на данните, базирана на InnoDB, разработена от Person. mysql>**CREATE TABLE if not exists my_db.tab1 (id INT PRIMARY KEY, name VARCHAR(20), email VARCHAR(20)) ENGINE=InnoDB;**

Още за таблиците в MYSQL бази от данни (ENGINE)

- И за последният елемент в командата CREATE TABLE, а именно SELECT(select_израз). Той е опционален. Синтаксисът му е:

[IGNORE | REPLACE] SELECT ...(всеки коректен израз SELECT)

Ако при създаване на таблицата се зададе SELECT, то всички полета, получени от извадката, се добавят в създаваната таблица.

Например:

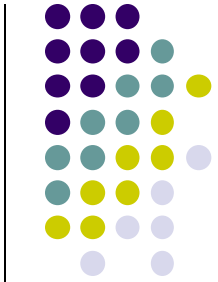
mysql>**CREATE TABLE test (a INT NOT NULL AUTO_INCREMENT, PRIMARY KEY (a), KEY(b))ENGINE=MyISAM SELECT b,c FROM test2;**

Създава се MyISAM таблица с 3 колони a, b, and c.

mysql>**CREATE TABLE new_tbl SELECT * FROM orig_tbl;**

Създава се таблица, съдържаща всички колони на оригиналната.

Още няколко примера за създаване на таблица:



```
mysql>CREATE TABLE my_db.persons (id INT PRIMARY KEY
    AUTO_INCREMENT, first_name VARCHAR(50), last_name
    VARCHAR(100), death_date INT, description TEXT, photo INT,
    citizenship CHAR(50) DEFAULT 'Bulgaria');

mysql>CREATE TABLE if not exists my_db.tab1 (id INT PRIMARY
    KEY, name VARCHAR(20), email
    VARCHAR(20))ENGINE=InnoDB;

mysql>CREATE TABLE if not exists my_db.tab2  (name
    VARCHAR(20), fnom VARCHAR(20), index(fnom));
```

ПОСЛЕДВАЩО ИНДЕКСИРАНЕ (ALTER TABLE)

- Освен индексиране с използване на някоя от клаузите PRIMARY KEY, KEY, UNIQUE, FULLTEXT в **CREATE TABLE**, MYSQL предоставя и други възможности за създаване на индекси, а именно - с **командите ALTER TABLE и CREATE INDEX :**

- При създадена вече таблица students, с командата **ALTER TABLE** може да се добави индекс за поле fnom:

mysql>ALTER TABLE students **ADD INDEX fnom(fnom);**

- **Индексиране с командата CREATE [UNIQIUE] INDEX**

mysql>**CREATE UNIQUE INDEX fnom ON students (fnom);**

Създаване на релации между таблици от тип InnoDB

- Ако две таблици са от тип InnoDB, то между тях може да се създаде релация, чийто интегритет се поддържа от MySQL.

- Задача: Да създадем две таблици, тип InnoDB, например:

Таблица **city** (реферираща таблица):

id (Primary key), Name, CountryCode (Foreign Key)

Таблица **country** (реферирана таблица):

Code (Primary Key) и Name

И да създадем релация между двете таблици

Създаване на релации между таблици от тип InnoDB

- Двете SQL команди за създаване на таблиците и за създаване на релацията между тях са:

1.) mysql>CREATE TABLE if not exists my_db.country

(**Code Char(3) NOT NULL PRIMARY KEY**,

name VARCHAR(20))ENGINE=InnoDB;

2.) mysql>CREATE TABLE if not exists my_db.city

(id INT PRIMARY KEY AUTO_INCREMENT,

name VARCHAR(20), **CountryCode Char(3),**

Index(CountryCode),

FOREIGN KEY(CountryCode) REFERENCES country(Code)

ON UPDATE CASCADE

ON DELETE CASCADE)ENGINE=InnoDB;

ON DELETE CASCADE- Ако един град с даден 'Code‘ бъде изтрит от реферираната таблица, то и записът (с CountryCode= Code) в реферираща таблица също ще бъде изтрит.

Създаване на релации между таблици - Още един пример

- Двете SQL команди за създаване на таблиците и за създаване на релацията между тях са:

Реферирана таблица:

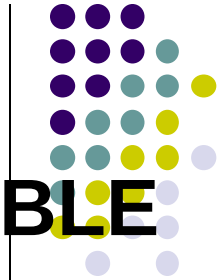
CREATE TABLE parent (id INT NOT NULL AUTO_INCREMENT, PRIMARY KEY (id));

Реферираща таблица:

CREATE TABLE child (id INT NOT NULL AUTO_INCREMENT, parent_id INT, INDEX parent_id, FOREIGN KEY (parent_id) REFERENCES parent(id) ON DELETE CASCADE);

ON DELETE CASCADE- Ако един запис с ‘id‘ бъде изтрит от реферираната таблица parent, то и записът (с parent_id = id) в рефериращата таблица child също ще бъде изтрит.

Изтриване на БД и таблица



Оператори DROP DATABASE и DROP TABLE

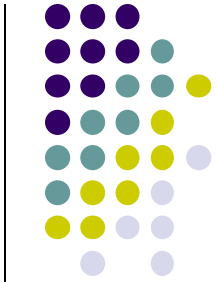
- Оператор DROP TABLE изтрива една или няколко таблици и трябва внимателно да се използва. Синтаксис:
- DROP TABLE [IF EXISTS] име_таблица [,име_таблица,...] [RESTRICT | CASCADE]
- Примери:

mysql>**DROP DATABASE test_db;**

mysql>**DROP TABLE IF EXISTS test_db.Persons, test_db.Artifacts, test_db.test;**

mysql>**DROP TABLE test_db.employee;**

Оператор SELECT

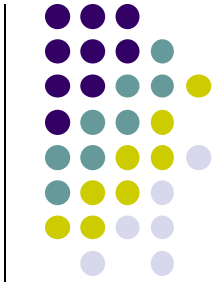


- mysql> SELECT * FROM Persons WHERE first_name='Александър';
- mysql> SELECT title,description FROM Artifacts WHERE id=10;

UPDATE, DELETE – примери:

- mysql> UPDATE Persons SET death_date=death_date+1;
- mysql> DELETE FROM Persons WHERE bday>2003;

Оператор INSERT – примери:



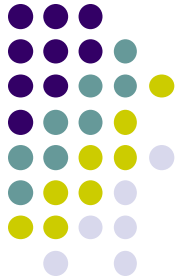
- **mysql> INSERT INTO Persons SET last_name='Петров', first_name='Иван';**

- **mysql> INSERT INTO Persons (last_name, bday) VALUES ('Иванов', '1934');**

Взаимодействие PHP и MySQL, чрез php_mysql.dll

- Казахме, че PHP предоставя 2 библиотеки или разширения (.dll) за операции с MySQL БД: php_mysql.dll и php_mysqli.dll
- Нека се запознаем първо с основни функции за работа с MySQL БД, посредством php_mysql.dll.
- Да вземем за пример таблица test_db.employee, в която се съдържа информация за служители:
  - Идентификатор: **emp_id**
  - Име: **emp_name**
  - Адрес: **emp_address**
  - Заплата: **emp_salary**
  - Дата на постъпване: **join_date**

Взаимодействие PHP - MySQL, чрез php_mysql.dll

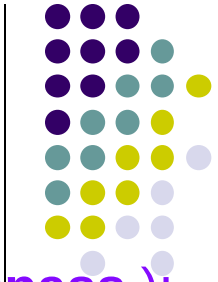


- Задача: Как да използваме PHP за създаване и използване на MySQL база от данни (**test_db**) и таблица (**employee**) в нея?

За целта ще разгледаме PHP функциите за:

- установяване и затваряне на съединение (на връзка) с MySQL сървър за БД;
- създаване, избор, изтриване на БД;
- получаване на списък на полетата на таблицата и за визуализирането им;
- за архивиране на БД и др.

Установяване, затваряне на конекция с MySQL сървър за БД: функции mysql_connect, mysql_close().



- Синтаксис на **mysql_connect**:

resource mysql_connect(string **$dbhost** , string **$dbuser**, string **$dbpass**);

Функцията установява връзка (конекция) с MySQL сървър и връща **идентификатор за това съединение** ($link_identifier, тип ресурс) или FALSE в случай на неуспех.

По подразбиране имаме:

- **$dbhost** - името или IP адреса на машината на която е инсталирана СУБД, например '**localhost**' (default) или '**localhost:3306**'. *Ако портът е различен от 3306, тогава той трябва задължително да се укаже след името на хоста, т.е имаме:* ***host:порт***.
- **$dbuser -** име на потребителя, достъпващ сървъра
- **$dbpass -** парола за достъп, по подразбиране – няма парола, т.е ''.

Н.р: **$conn = mysql_connect'localhost', 'root', 'admin');**

Ако функцията се извика втори път със същите параметри, то ново съединение не се установява, а се връща указател за старото съединение.

Създаване на MySQL БД

- За да създадете (или изтриете) една MySQL база от данни, вие трябва да имате администраторски права.
- За целта, PHP използва функция **mysql_query()**, която има два параметъра и връща TRUE при успех и FALSE при неуспех:

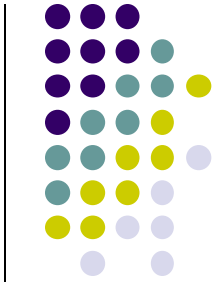
resouce mysql_query (string $query [,resouce $link_identifier]);

Н.р:

$retval = mysql_query('CREATE Database test_db', $conn);

която изпраща SQL- заявка ($query) към активната БД на MySQL сървъра, като връзката се идентифицира с помощта на $link_identifier. Ако $link_identifier е изпуснат, се използва последното направено съединение. Връща тип ресурс – тоест указател към резултата, иначе False.

Затваряне на съединение с MySQL сървър за БД: функции mysql_close().



- Затварянето на съединението: функция **mysql_close().** Общ вид:

bool mysql_close([resource **$link_identifier**])

$link_identifier е идентификатор на връзката, върнат от mysql_connect().

Функцията връща TRUE/FALSE.

Н.р: **mysql_close($conn);**

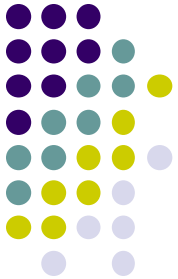
- Функция: **mysql_error().** Общ вид:

string mysql_error([resource $link_identifier])

Функцията връща текстово съобщение за възникнала грешка.

Н.р: **if(! $conn) {die('Could not connect: ' . mysql_error());}**

Установяване на съединение с MySQL сървър за БД и създаване на БД: функции mysql_connect, mysql_query.

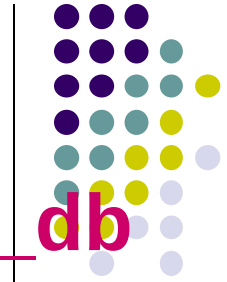


- Пример:

```
<?php
$dbhost = 'localhost'; $dbuser = 'root‘; $dbpass = 'admin';

$conn = mysql_connect($dbhost, $dbuser, $dbpass);
// $conn -  идентификатор на връзката
if(! $conn )
        {  die('Could not connect: ' . mysql_error());}
$sql = 'CREATE Database test_db';
$retval = mysql_query( $sql, $conn );
if(! $retval ){  die('Could not create database: ' .
   mysql_error());}
mysql_close($conn);
?>
```

Селектиране на MySQL база от данни

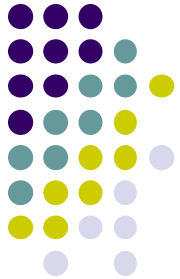


- PHP предоставя функция **mysql_select_db** за селектиране на БД, връщаща TRUE при успех и FALSE – при неуспех.
- Синтаксис:

bool mysql_select_db(string $db_name[, resource $link_identifier]); Където:

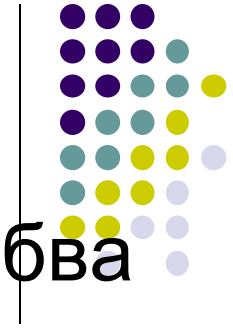
- $db_name - име на БД
- $link_identifier – идентификатор на връзката, върнат от mysql_connect(); ако не се зададе, то се използва последната отворена конекция със сървъра за MySQL БД.

Селектиране на MySQL база от данни - пример



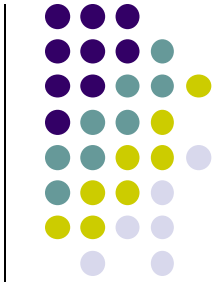
```
<?php
 mysql_connect("localhost", "root", "admin")
                                    or die(mysql_error());
echo "Connection to the server successful!<br/>";
mysql_select_db(test_db)
                                      or die(mysql_error());
echo "Database was selected!<br/>";
?>
```

Създаване на таблица в БД



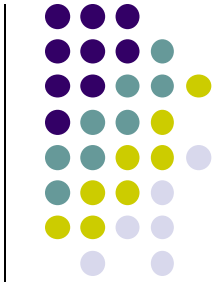
- За да създадете таблици в БД, вие трябва да постъпите по същия начин, както при създаване на БД.
- Първо създавате SQL заявка (SQL query) за създаване на таблиците, след което изпълнявате тази заявка, използвайки я като параметър на функция **mysql_query().**

Създаване на таблица employee в БД test_db



```
…
$conn = mysql_connect($dbhost, $dbuser,
   $dbpass);
$sql ="CREATE TABLE employee (emp_id INT
   NOT NULL AUTO_INCREMENT,  emp_name
   VARCHAR(20) NOT NULL,  emp_address
   VARCHAR(20) NOT NULL,  emp_salary INT
   NOT NULL,  join_date timestamp NOT NULL,
   PRIMARY KEY  (emp_id))";
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn);…
```

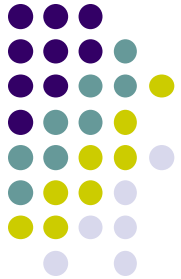
Запълване на таблица от БД с данни



- За да вмъкнем един запис в таблица employee на БД test_db съхранете SQL INSERT INTO … оператор в една променлива и я подайте като параметър на една mysql_query по следния начин:

```
$sql = 'INSERT INTO employee '.
   '(emp_name,emp_address, emp_salary,
   join_date) '. 'VALUES ( "guest", "XYZ", 2000,
   NOW() )';
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn);
```

Извличане на данни от база от данни



- Знаем, че чрез SQL оператора:

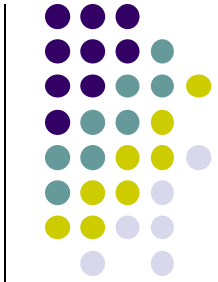
SELECT * FROM test_db;

се извличат всички записи от таблица **test_db** на БД.

- Ние се нуждаем да изпратим тази заявка на сървъра и да съхраним отговора. За да направим това ще използваме функцията на PHP - mysql_query() по следния начин:

$result =mysql_query("SELECT * FROM test_db");

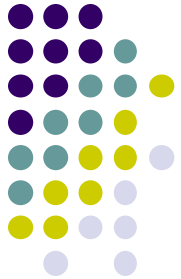
Един пример - извличане на данните от таблица employee



```
<?
php$dbhost = 'localhost';$dbuser = 'root'; $dbpass = 'admin';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )die('Could not connect: ' . mysql_error());
$sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )die('Could not get data: ' . mysql_error());
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{echo "EMP ID :{$row['emp_id']}  <br> ".
        "EMP NAME : {$row['emp_name']} <br> ".
        "EMP SALARY : {$row['emp_salary']} <br> ".
        "------------------------------<br>";}
mysql_close($conn);
?>
```

```
EMP ID :1
EMP NAME : guest
EMP SALARY : 2000
------------------------------
EMP ID :2
EMP NAME : Jane
EMP SALARY : 2000
------------------------------
```

Функции за визуализиране на резултата

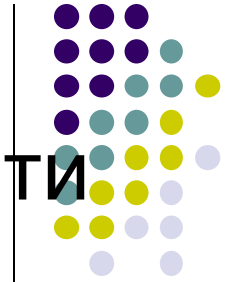


- **mysql_fetch_assoc (resource $result)** – Връща резултата = ред като **асоциативен масив**
- **mysql_fetch_row(resource $result)** - Връща резултата = ред като **номериран масив**
- **mysql_fetch_array (resource $result [, int $result_type])** - Връща резултата = ред като асоциативен, номериран масив или и двата.

$result е резултатът върнат от mysql_query() function

- **Всички тези функции конвертират един запис от резултата в масив, който след това може да се използва. В скрипта бе използван асоциативен масив.**
- **Всеки елемент на получения масив е поле от реда, като ключ на елемента е името на полето в реда.**

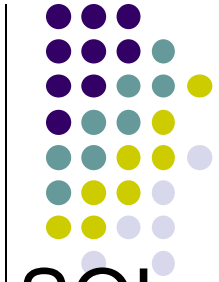
Модифициране на записи от БД



- Модифицирането на запис в БД е почти същото като INSERT.
- Изпълнява се SQL **UPDATE** команда, посредством PHP функция: **mysql_query()**.
- Парчетата код, които следват са за таблица employee:

```
$sql = "UPDATE employee SET
   emp_salary=4500 WHERE emp_name
   =‘Jane"";
$result =mysql_query($sql);
```

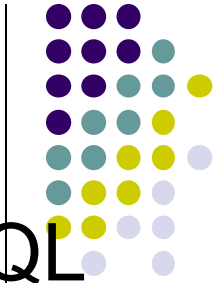
Изтриване на записи от БД



- Изтриване на запис от БД: изпълнява се SQL **DELETE** команда, чрез PHP функция mysql_query.
- За БД test_db, таблица employee :

```
$sql = "DELETE FROM employee WHERE
   name='Jane'";
$result =mysql_query($sql);
```

Изтриване на таблица от БД

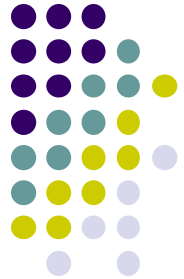


- В **mysql_query()** командата се подава SQL команда за изтриване на таблица (DROP TABLE test_db.employee), например:

...

```
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn ){  die('Could not connect: ' .
    mysql_error());}
$sql = 'DROP TABLE test_db.employee';
$retval = mysql_query( $sql, $conn );
if(!$retval )
      {die('Could not delete db_test: ' . mysql_error());}
```

Изтриване на базата от данни

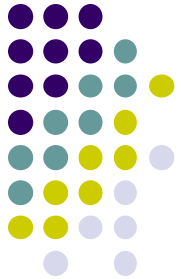


- В mysql_query командата се подава SQL команда за изтриване на БД (DROP DATABASE test_db), например:

...

```
$conn = mysql_connect($dbhost, $dbuser,
    $dbpass);
if(! $conn ){  die('Could not connect: ' .
    mysql_error());}
$sql = 'DROP DATABASE test_db';
$retval = mysql_query( $sql, $conn );
if(!$retval )
    {die('Could not delete db_test: ' . mysql_error());}
```

Други функции



- В PHP за получаване на списък на полетата на таблица от БД се използва функция **mysql_list_fields().**Синтаксис:

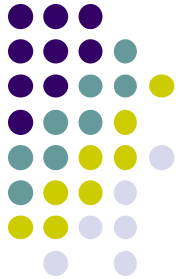
resource **mysql_list_fields**(string database_name,string table_name [,resource $link_identifier]);

където: $link_identifier - идентификатор на връзката, върнат от mysql_connect().

int **mysql_num_fields** (resource $result);

- връща броя на колоните в $result, където $result е указател към резултата върнат от mysql_query().

Други функции



- Функция:

int **mysql_num_rows** (resource $result);

връща броя на редовете в $result, където $result е указател към резултата върнат от mysql_query().

- Функция:

mysql_field_name() връща име на полето,

mysql_field_len() връща дължината на полето, а

mysql_field_type() – типа на поле.

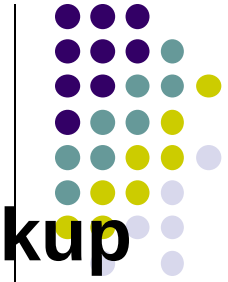
- Пример:

```
<?
$conn = mysql_connect("localhost", "nina","123") or
    die("Невъзможно да се установи връзка: ". mysql_error());
mysql_select_db("book"); // Persons е таблица в БД book
$list_f = mysql_list_fields ("book","Persons",$conn);
$n = mysql_num_fields($list_f);
for($i=0;$i<$n; $i++){
$type = mysql_field_type($list_f, $i);
$name_f = mysql_field_name($list_f,$i);
$len = mysql_field_len($list_f, $i);
echo "<br>Име на поле: ". $name_f;
echo "<br>Тип на поле: ". $type;
echo "<br>Дължина на поле: ". $len. "<hr>";}
?>
```

Резултат:

Съединението е установено! Правим book текуща!!!
Име на поле: id
Тип на поле: int
Дължина на поле: 11

MySQL backup, чрез PHP



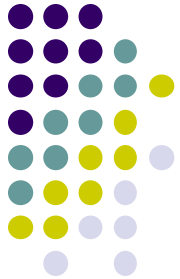
Винаги е полезно да се прави регулярно **backup на БД**. Има различни начини за тази цел:

- Използване на SQL **SELECT INTO OUTFILE** команда на PHP. Всяка таблица се съхранява в отделен текстов файл.
- Използване на MySQL **mysqldump** команда на PHP.
- Използване на **phpMyAdmin**.

- И др.

Ще се ограничим, до разглеждане на първия начин:

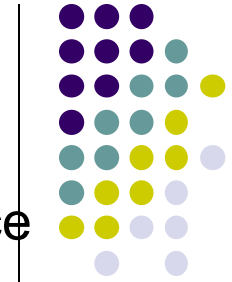
MySQL backup, чрез PHP



Изпълнение на SQL **SELECT INTO OUTFILE** команда на PHP.

```
...
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
mysql_select_db('test_db');
$table_name = "employee";
$backup_file  = "C:/xampp/tmp/employee.sql";
$sql = "SELECT *  FROM $table_name INTO OUTFILE
   '$backup_file'";
$retval = mysql_query( $sql, $conn );
if(! $retval ){  die('Could not take data backup: ' .
   mysql_error());}
echo "Backedup  data successfully\n";
mysql_close($conn);
```

MySQL backup, чрез PHP



За да се възстановят данните (restore the backup) - използва се **LOAD DATA INFILE** заявка:

```
...
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
$table_name = "employee";
$backup_file  = "C:/xampp/tmp/employee.sql";
$sql = "LOAD DATA INFILE '$backup_file' INTO TABLE
    $table_name";
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval ){  die('Could not take data backup: ' .
    mysql_error());}
mysql_close($conn);
```