

AZERBAIJAN CODERS' INSTITUTE



MySQL

# 10 dərəcə MySQL öyrən

“MySQL Development Training” təlimindən

**Etibar Vəzirov**

**Bakı - 2016**

# ÖN SÖZ

**Əziz tələbələr və proqramlaşdırma ilə maraqlanan həmvətənlər!**

Sizə təqdim etdiyim bu vəsait 02-12.07.2016 tarixində **Bakı Dövlət Universitetində** təşkil etdiyim "**MySQL Development Training**" təliminin materialları əsasında hazırlanmışdır. Doğma dilimizdə proqramlaşdırma və verilənlər bazası ilə əlaqədar tədris materiallarının azlığı səbəbindən və İT ixtisasında təhsil alan gənclərimizə yardım məqsədilə bu vəsaiti ərəsəyə gətirməyi qarşıma məqsəd qoydum. Düşünürəm ki bu kitabda bəhs olunan mövzuları diqqətlə oxuyub nümunələri test etməklə sizlərdə MySQL dili haqqında köklü biliklər formalaşacaqdır.

Kitabdan tələbələrlə yanaşı yuxarı sinif şagirdləri, müəllimlər həmçinin verilənlər bazası ilə maraqlanan hər kəs faydalana bilər.

**Hər birinizə müvəffəqiyyətlər arzu edirəm əzizlərim!**

**Hörmətlə : Etibar Vəzirov**

Mərhəmətli və Rəhimli Allahın adı ilə



## “MySQL Development Training”

### 1-ci dərş

*Verilənlər bazası. Relyasiyalı verilənlər  
bazası. SQL*

Təlimçi : *Etibar Vəzirov*

*Java Developer*

- **Data ilə informasiyanın fərqi nədir?**
- **Verilənlər bazası (database) nədir?**
- **Relyasiyalıq nə deməkdir?**
- **SQL -ə giriş.**
- **MySQL nədir?**

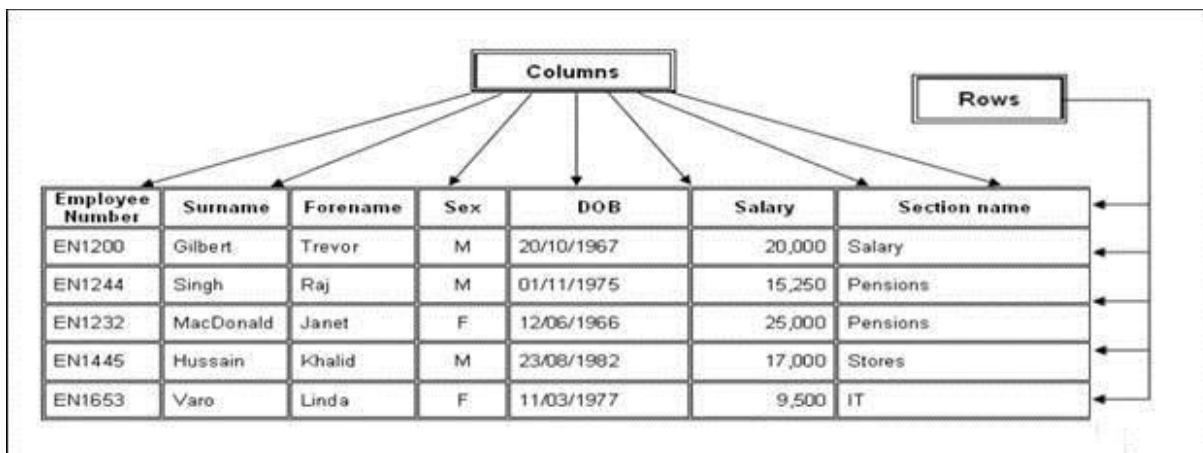


**Data konkret mənə kəsb etməyən faktlardır. Məsələn 27  
İnformasiya isə müəyyən bir mənaya malikdir. Məsələn  
məlumat cədvəlində yaş sütununda 27 yazılmışdır.**

**Verilənlər bazası bir  
biri ilə məntiqi  
əlaqəsi olan cədvəllər  
toplusunun saxlanma  
sistemidir.**

**Cədvəllərdə  
məlumatlar sətir və  
sütünlərdə saxlanılır.**





Bu cədvəllər müəyyən daxili xüsusiyyətlərinə görə əlaqədirlər. Məsələn :

Baker Table

Baker ID	Baker name	Address	Region ID
1	Ejder emi	N.Aliyev rd.	3
2	Gulu kishi	Jafarov st.	1
3	Saleh abi	Safarali ave	2
4	Ali dede	Ismail rd.	1

Region Table

Region ID	Settlement	District
1	Guzanli	Agdam
2	Gilazi	Khizi
3	Garacalar	Saatli
4	Vendam	Gabala

Bəs **RELYASİYALILIQ** nədir?

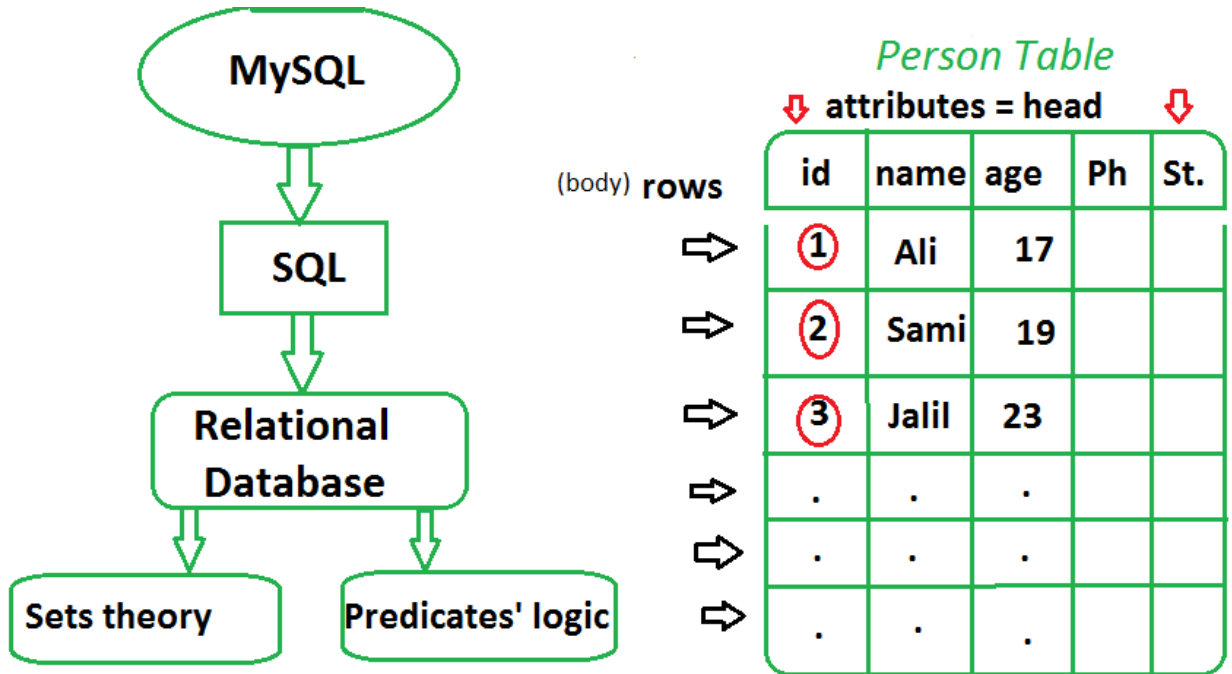
Relyasiyalı verilənlər bazası **çoxluqlar nəzəriyyəsi** (Sets theory) və **predikatlar məntiqinə** (predicates logic) əsaslanır.

Deyək ki aşağıdakı kimi *Person* cədvəlimiz var. Cədvəldə sütunlar atributları, sətirlər body hissəni əmələ gətirir.

Cədvəldə sətirlər **ixtiyari ardıcılıqla** yerləşir. Bu **çoxluqlar nəzəriyyəsinə** əsaslanır. Və bu cədvəldə müəyyən bir şərtə uyğun məlumatın alınmasına isə **predikat məntiqi** deyilir. Məsələn:

> *select name from person where id = 2*

bu sorğu bizə id-si 2 olan adı göstərəcək



Belə bir əlaqəliliyə **RELYASIYALILIQ** deyilir.

## SQL - ə giriş.

SQL - verilənlər bazasına manipulyasiya etmək üçün standart sorğu dilidir. Açılışı **Structured Query Language** şəklindədir.



SQL - lə nələr etmək mümkündür?

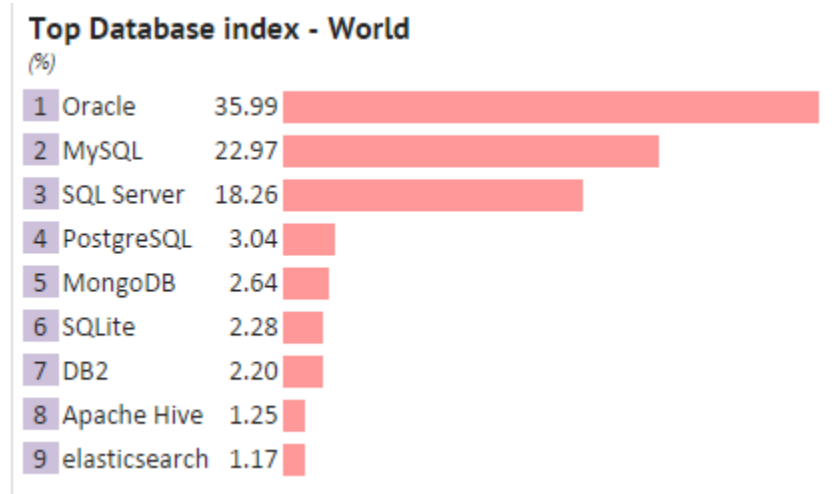
- ✓ SQL -lə baza yaradıla bilir
- ✓ SQL -lə bazaya sorğular göndərilir
- ✓ SQL -lə bazadan məlumatı kopyalamaq olur
- ✓ SQL -lə məlumatlar bazaya daxil edilir, oxunur, dəyişdirilir və silinir (*insert, select, update, delete*)
- ✓ SQL -lə bazada yeni cədvəllər yaradılır(*create*)
- SQL-lə bazada cədvəllər birləşdirilə bilir (*join*) və s.

SQL bir standartdır...və *American National Standards Institute* tərəfindən təyin olunmuşdur.

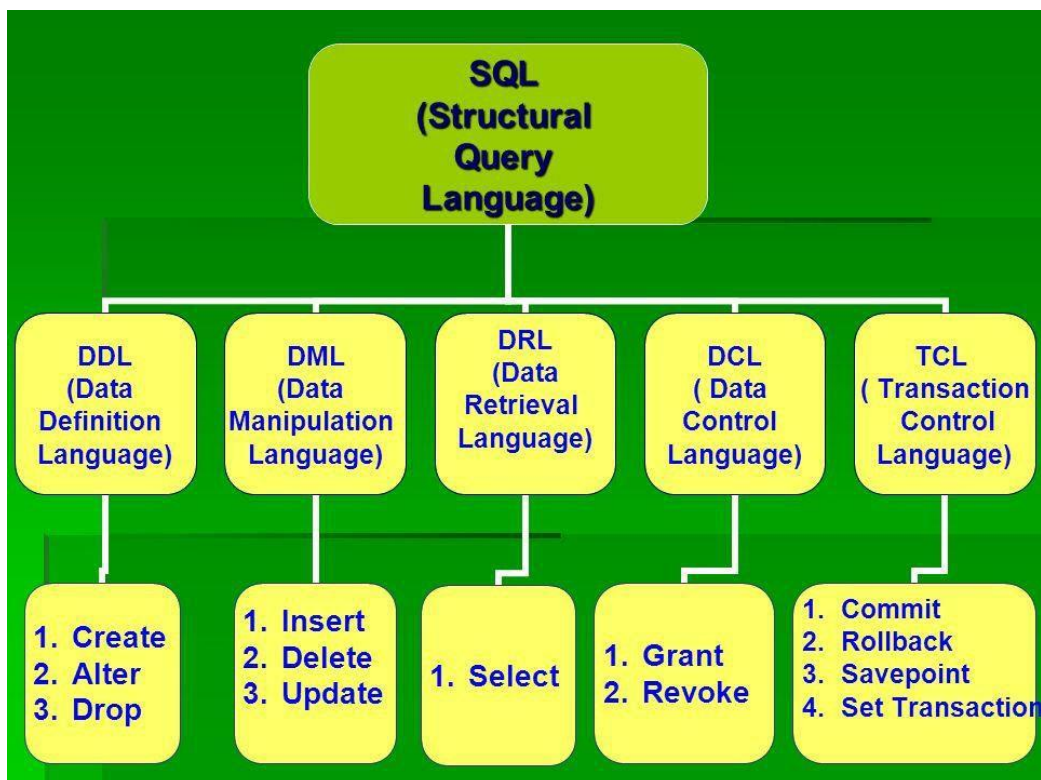
Hazırda SQL -in müxtəlif versiyaları mövcuddur. Onlar SQL- in modifikasiya olunmuş

növləridir və SQL -in əsas komandalarını dəstəkləyir (*select, update, delete, insert, where*)

Sağdakı diagramda **MySQL** -in digər SQL dilləri ilə müqayisədə istifadə reytingini görürük :



## SQL -də komanda qrupları (SQL statements)





## SQL -də açarlar (keys in SQL)

- primary key
- foreign key
- unique key
- composite key



Cədvəldə **primary key** sətirlərin yeganə təyinedicisidir, hər bir sətir üçün primary key olmalıdır. O **null** dəyəri ala bilməz. Həmçinin iki sətir eyni primary key-ə sahib ola bilməz!

**Foreign key** adından bəlli olduğu kimi başqa cədvəlin primary key - nə və ya unique key -nə istinad edir (başqa sözlə onu təmsil edir).

**Unique key** də cədvəldə hər bir sətirin yeganə təyinedici açarıdır, lakin primary key -dən fərqli olaraq o **NULL** dəyəri ala bilər. Bununla belə baza cədvəlinin birdən çox unique key -ləri ola bilər.

Əgər cədvəli yaradarkən 1-dən çox sütünü primary key (və ya foreign key) kimi veririksə bu

bağlığa composite key deyilir. Bir nümunəyə baxaq :

```
create table account (  
acc_num int,  
acc_type int,  
acc_desc char(500),  
primary key (acc_num , acc_type)  
)
```

burada **primary key (acc\_num , acc\_type)** **composite key** rolunu oynayır.

## MySQL nədir?

MySQL çox geniş yayılmış open source verilənlər bazası sistemidir.

MySQL SQL dilini başa düşən bir data baza dilidir.

MySQL dili **MySQL AB** adlı İsveç şirkəti tərəfindən yaradılmışdır. Bu struktur sorğu dili bir çox önəmli səbəblərə görə məşhurlaşmışdır :



- **MySQL açıq qaynaq kodludur (open source), bu o deməkdir ki istifadəsi tam pulsuzdur**
- **Çox bahalı və güclü databaza paketlərinin funksionallıqlarına sahibdir**
- **Bu dil SQL dilinin standartları üzərində qurulmuşdur və onları dəstəkləyir**
- **Bir çox platformalarda (OS) və bir çox dillərlə birgə çox asan integrasiya ola bilir(məs : PHP, Perl, C, C++, Java və s.)**
- **MySQL -lə böyük databazalar qurmaq olur, təxminən 50 milyondan çox sətiri olan cədvəllər(cədvəl üçün faylın standart ölçü limiti 4GB-dir, lakin artırmaq mümkündür)**
- **MySQL özəlləşdirilə biləndir, belə ki proqramçılar öz spesifik mühitlərinə uyğunlaşdırmaqla onu modifikasiya edə bilərlər. Bu da bazanın etibarlılığı baxımından çox önəmlidir.**

\*\*\*

## 1-ci darsin sonu

Növbəti darsin mövzusu :

*MySQL -in qoşulması və bazalarla iş*



**Diqqətiniz üçün təşəkkürlər**



Mərhəmətli və Rəhimli Allahın adı ilə



## MySQL Development Training

### 2-ci dər

### *MySQL -in qoşulması. Bazalarla iş*



Təlimçi : **Etibar Vəzirov**

*Java Developer*

## Windows ƏS -də MySQL -in yüklənməsi :

1. Öncə əməliyyat sisteminizin tipinə uyğun olan MySQL servisini aşağıdakı linkdən yükləyirsiniz :

**64 bitlik ƏS üçün :**

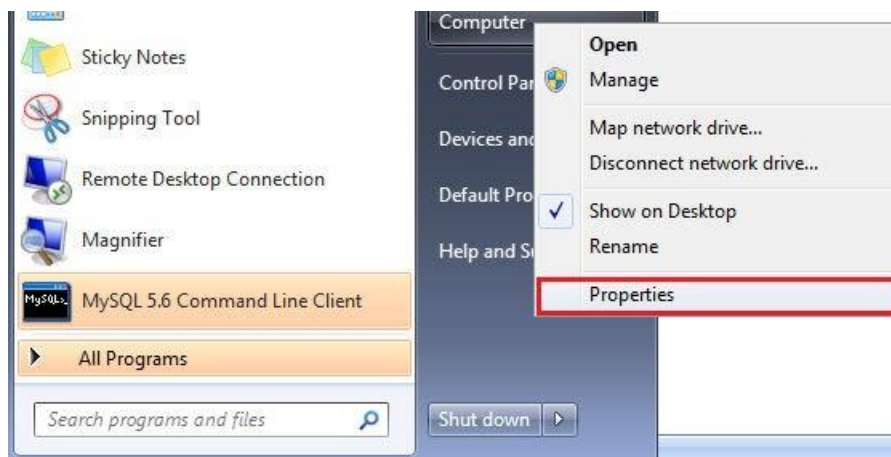
<https://drive.google.com/open?id=0B6zXK8pv4VnRMGFxMjIVeDBxMzQ>

**32 bitlik ƏS üçün :**

<https://drive.google.com/open?id=0B6zXK8pv4VnRb2VkekdnbVJFSDQ>

**Qeyd:** Əməliyyat sisteminizin tipini müəyyən etmək üçün aşağıdakı instruksiyaya əməl edin :

1. Start menyusunu açın və sağda **Computer** yazısı üzərində sağ düyməni klik edib **Properties** seçin:



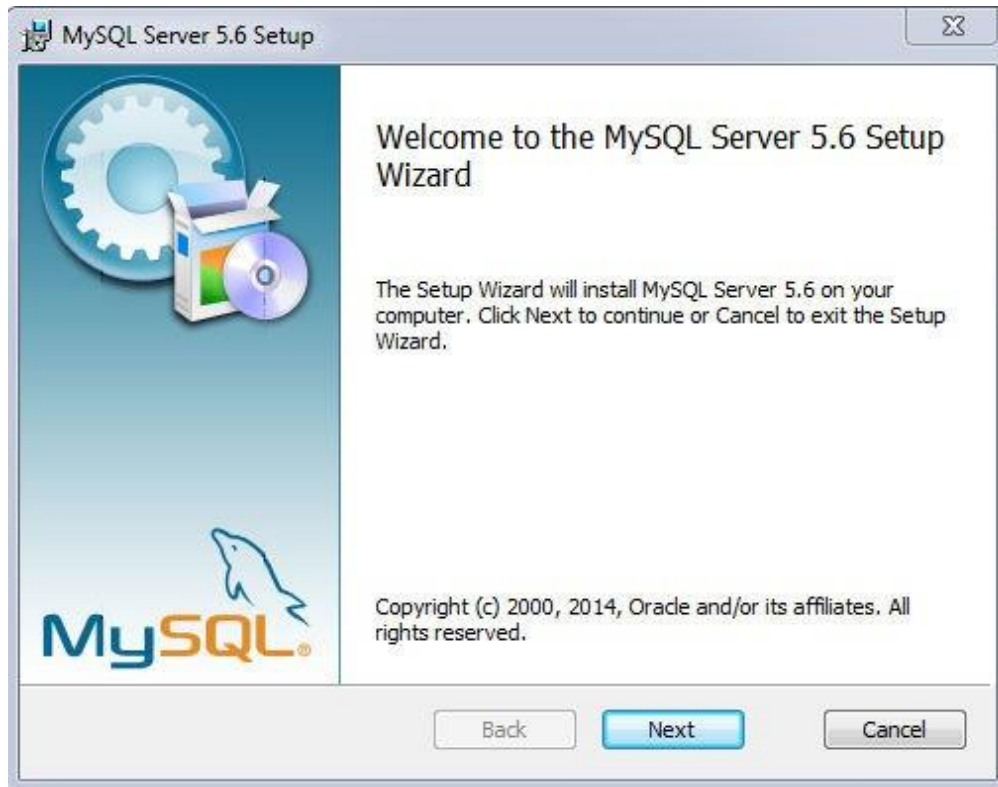
2. Açılan pəncərədə **System** başlığı altında **System type** yazılmış hissəyə nəzər yetirin:

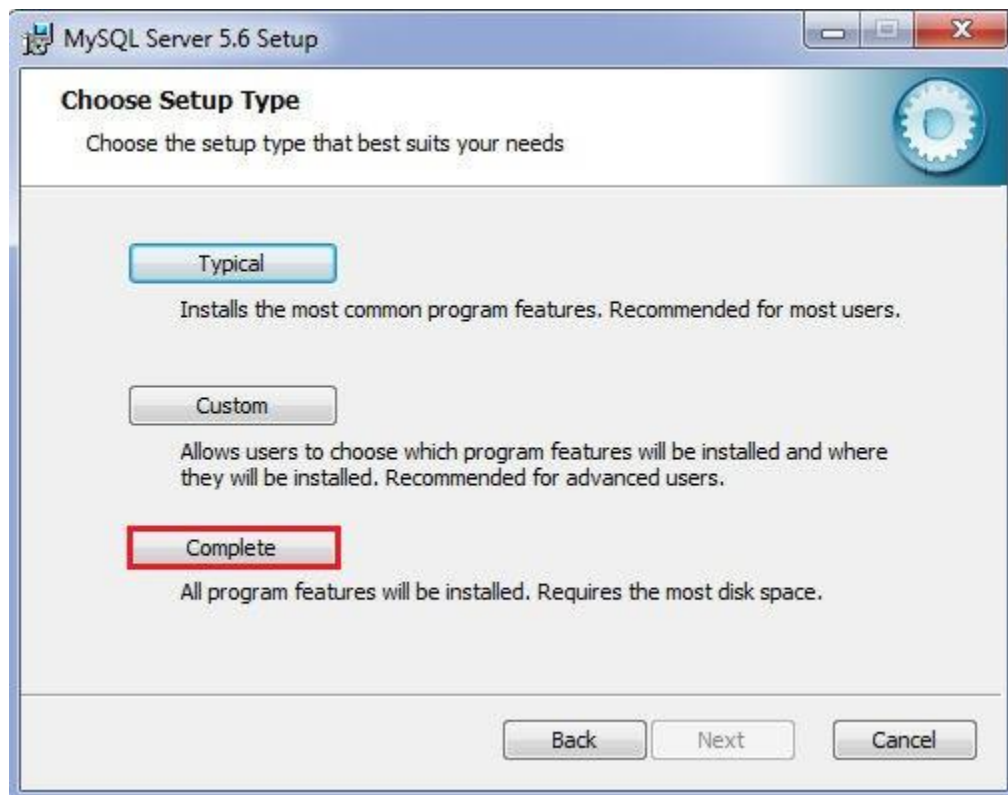
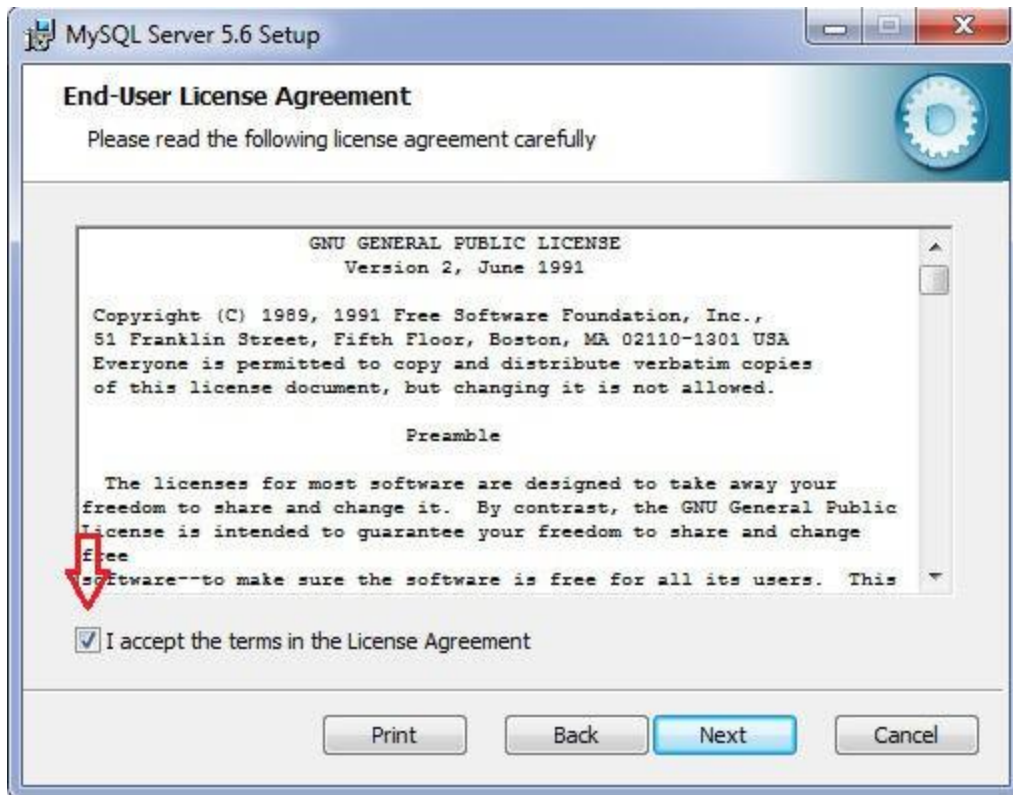
## System

Rating:	System rating is not available
Processor:	Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40 GHz
Installed memory (RAM):	8,00 GB (7,88 GB usable)
System type:	64-bit Operating System
Pen and Touch:	No Pen or Touch Input is available for this Display

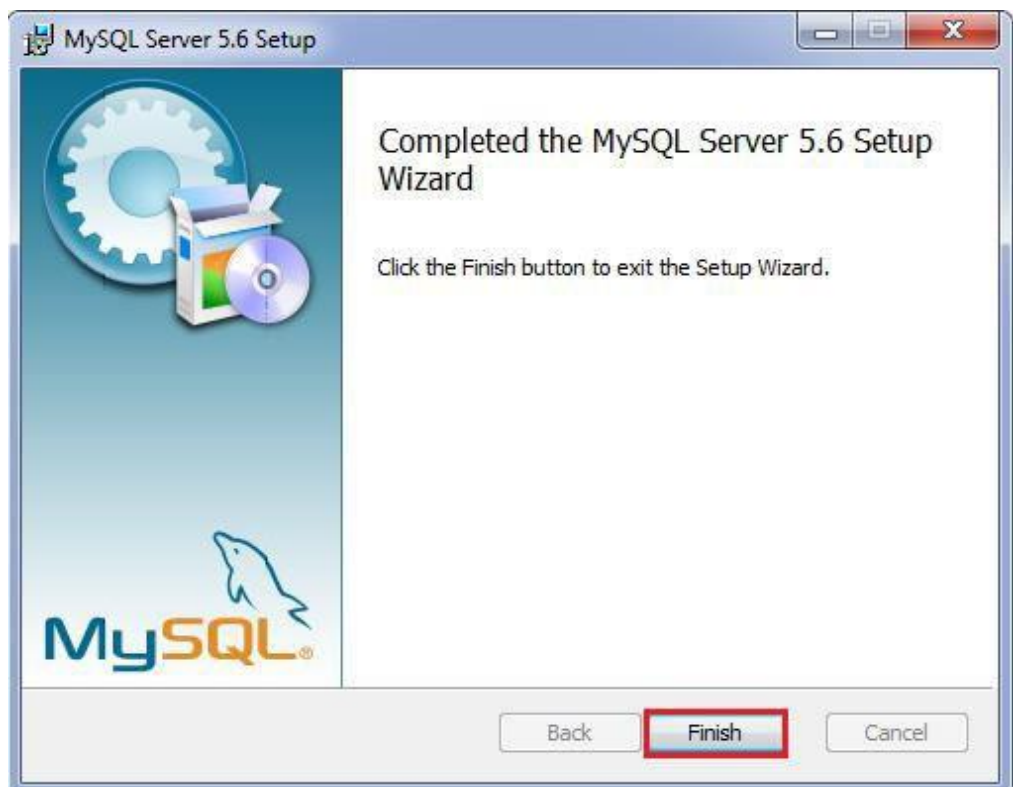
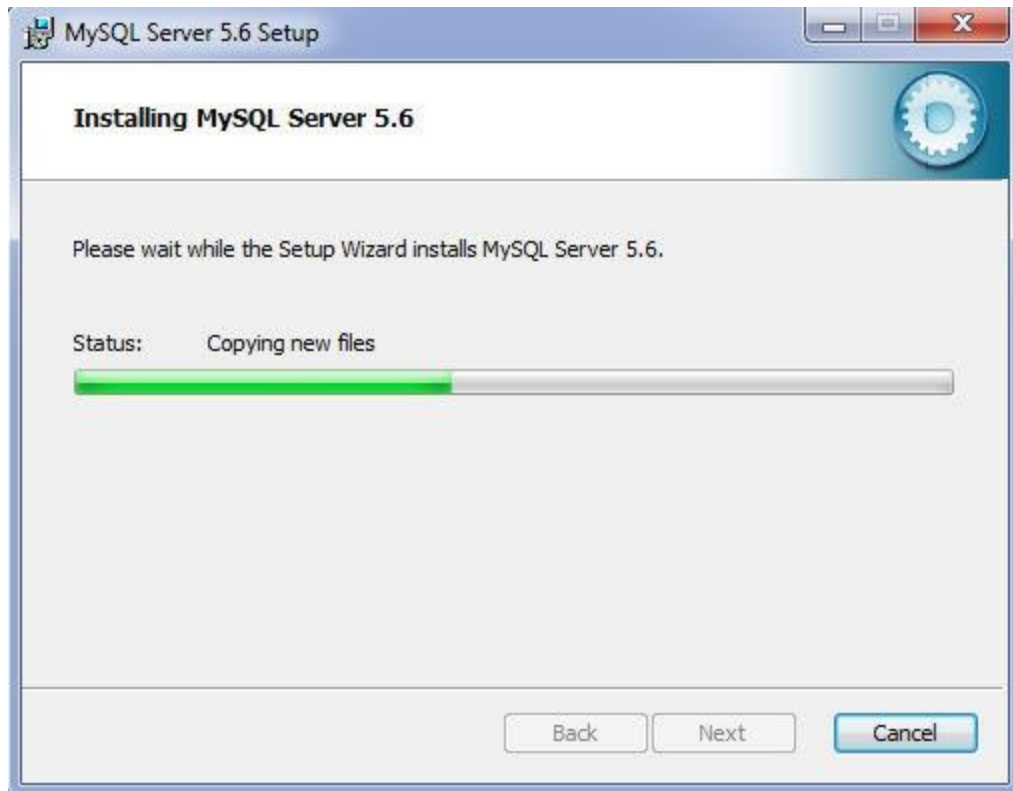
*Buradan əməliyyat sisteminizin 32 və ya 64 bit olduğunu müəyyən edəcəksiniz.*

**2. Bundan sonra MySQL-in qurulma əməliyyatı aşağıda şəkillərdə göstərildiyi kimi icra olunmalıdır :**



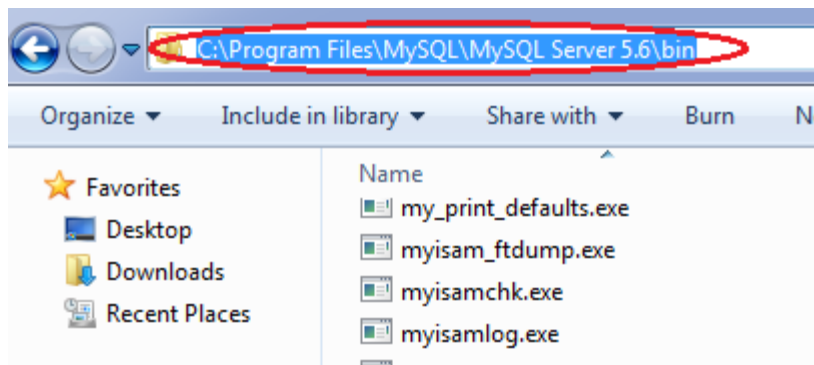




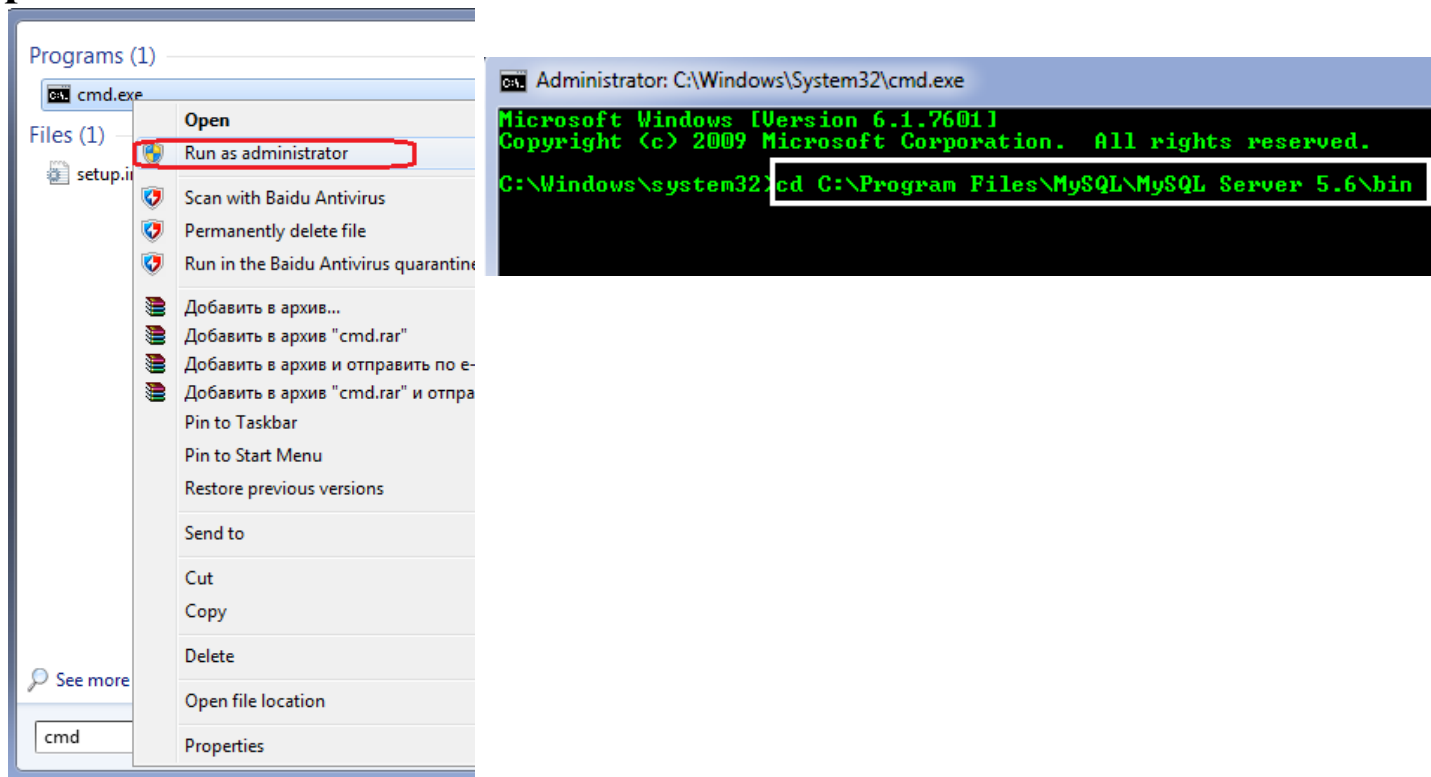


**MySQL -i install etdikden sonra MySQL qovluğunun yolu aşağıdakı kimi olacaq:**

**C:\Program Files\MySQL\MySQL Server 5.6**



**Siz öz kompüterinizdə bu yolu copy edib əmrlər sətrində paste edəcəksiniz. Belə ki**



**Bundan sonra MySQL servisini aktiv etmək lazımdır.**

Servisi aktiv etmək üçün əmrlər sətrində mysql daxilində **mysqld -install** yazmaq lazımdır :

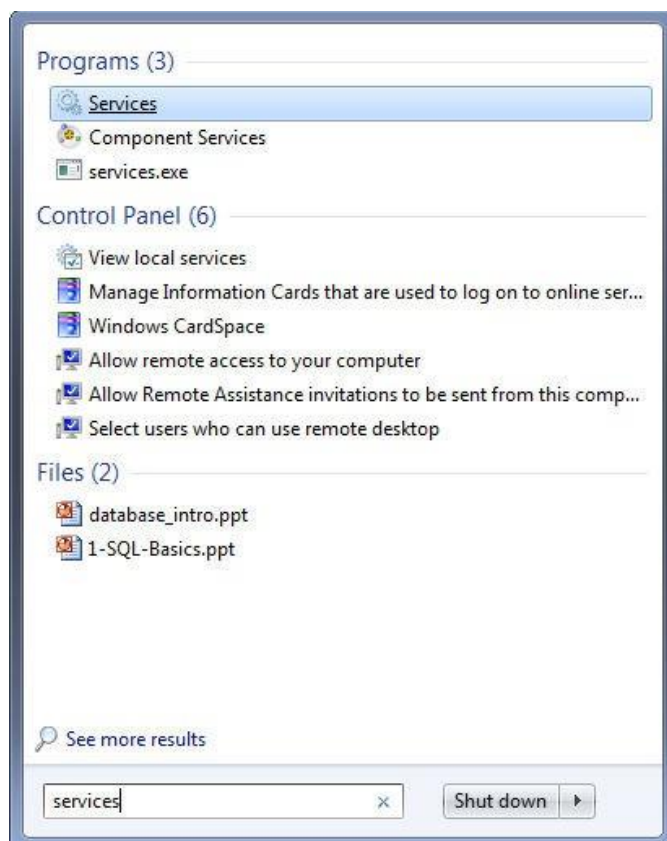
```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

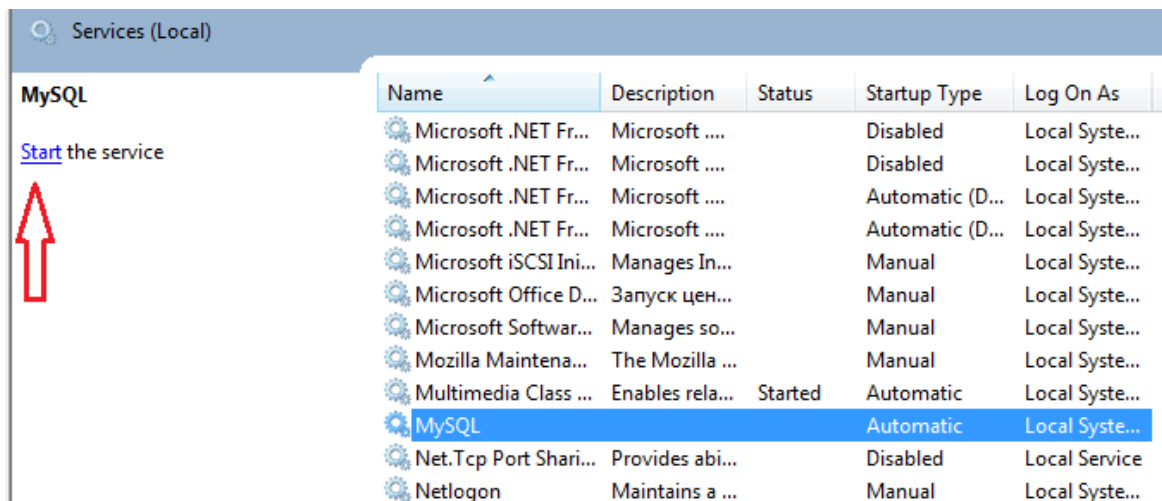
C:\Windows\system32>cd C:\Program Files\MySQL\MySQL Server 5.6\bin
C:\Program Files\MySQL\MySQL Server 5.6\bin>mysqld -install
Service successfully installed.
C:\Program Files\MySQL\MySQL Server 5.6\bin>_
```

Enter -ə vurduqda servisin müvəffəqiyyətlə qoşulması yazısını görəcəksiniz.

MySQL servisi bir çox rezident proqramlar kimi (məsələn antiviruslar) arxa planda işləyir. Onu start etmək üçün aşağıdakı addımları icra edirik: Start menyusunda **Services** yazıb ona daxil olursunuz.

Açılan pəncərədə MySQL servisini tapıb üzərinə klik edirsiniz.Sol tərəfdə **Start** yazısına klik etdikdən sonra MySQL servisi aktiv hala gələcəkdir. Bu o deməkdir ki siz artıq MySQL -də verilənlər bazasına manipulyasiya edə bilərsiniz.





## MySQL -də bazalarla iş.

MySQL yüklənən zaman default olaraq istifadəçi adı root qəbul olunur. Buna görə mysql-ə qoşulmaq üçün

**C:\Program Files\MySQL\MySQL Server 5.6\bin** qovluğu daxilində aşağıdakı əmri yazmaq lazımdır : *mysql -u root*

```

Administrator: C:\Windows\System32\cmd.exe - mysql -u root
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Program Files\MySQL\MySQL Server 5.6\bin
C:\Program Files\MySQL\MySQL Server 5.6\bin>mysqld -install
Service successfully installed.

C:\Program Files\MySQL\MySQL Server 5.6\bin>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>

```

Öncədən parol təyin olunmadığı üçün bu əmri yazıb enter-ə vuraraq mysql-ə daxil oluruq. Çıxış üçün də sadəcə **exit** (və ya **quit**) əmrini yazmaq kifayət edir:

```
Administrator: C:\Windows\System32\cmd.exe
mysql> exit
Bye
C:\Program Files\MySQL\MySQL Server 5.6\bin>
```

İstifadəçiyə parol təyin etmək üçün mysql-də aşağıdakı əmr sətirini yazmaq lazımdır:

*set password for 'root'@'localhost'=PASSWORD('abc');*

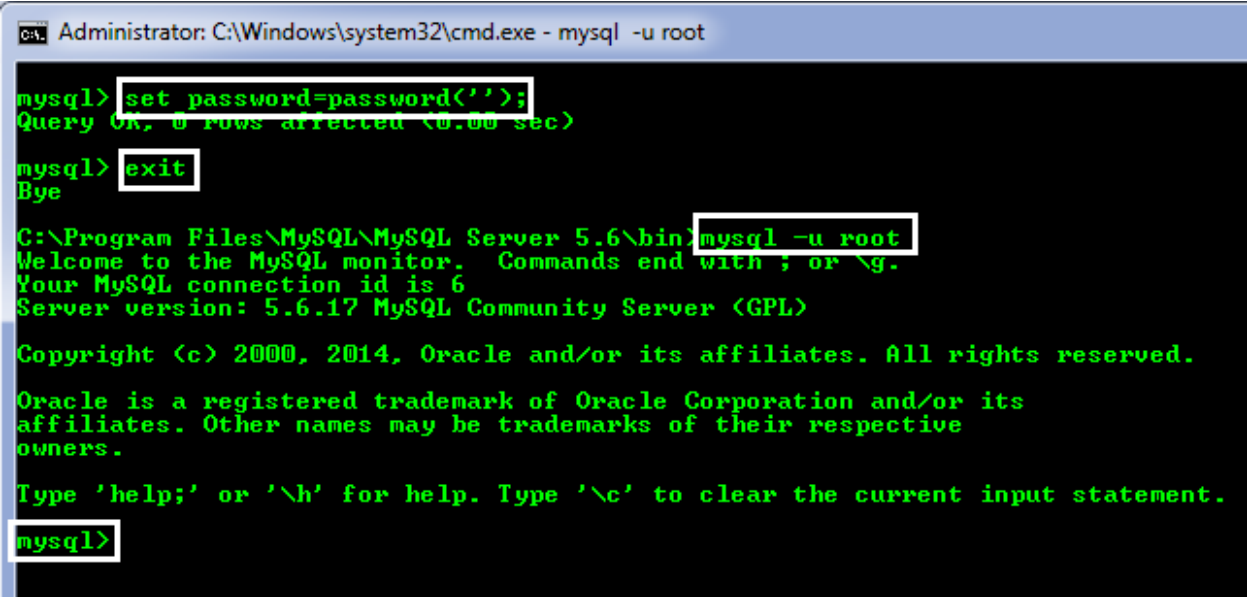
bu əmri yazmaqla biz lokalda mövcud olan root adlı istifadəçiyə *abc* şifrəsini təyin edirik.

```
Administrator: C:\Windows\system32\cmd.exe - mysql -u root -p
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Admin>cd C:\Program Files\MySQL\MySQL Server 5.6\bin
C:\Program Files\MySQL\MySQL Server 5.6\bin>mysql -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.17 MySQL Community Server (GPL)
Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> set password for 'root'@'localhost'=PASSWORD('abc');
Query OK, 0 rows affected (0.00 sec)
mysql> exit
Bye
C:\Program Files\MySQL\MySQL Server 5.6\bin>mysql -u root
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
C:\Program Files\MySQL\MySQL Server 5.6\bin>mysql -u root -p
Enter password: ***
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.6.17 MySQL Community Server (GPL)
Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

Şəkildən də görüldüyü kimi parol təyin edildikdən sonra mysql-ə parolsuz daxil olmaq (sadəcə *mysql -u root* yazmaqla) mümkün olmur. Artıq parol təyin olunduğuna görə onun tələb olunması üçün giriş əmrini *mysql -u root -p* kimi yazmaq lazımdır. Bu zaman " **Enter password :** " olan hissəyə öncədən təyin etdiyimiz parolu yazmaqla daxil ola bilirik.

Təyin olunmuş parolu sıfırlamaq üçün mysql -də aşağıdakı əmri yazmaq lazımdır :

*set password=password(' ');*



```
C:\Windows\system32\cmd.exe - mysql -u root

mysql> set password=password(' ');
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye

C:\Program Files\MySQL\MySQL Server 5.6\bin>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.6.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Göründüyü kimi parolu sıfırladıqdan sonra **exit** əmri ilə çıxış edib yenidən daxil olduqda (*mysql -u root* yazaraq) heç bir problemsiz yenidən mysql-ə daxil oluruq.

Eyni qaydada istifadəçi adını istəyimizə uyğun dəyişə bilərik. Bunun üçün əmri belə yazırıq:

*update mysql.user set user='etibar' where user='root';*

```
Administrator: C:\Windows\System32\cmd.exe - mysql -u etibar
mysql> update mysql.user set user='etibar' where user='root';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0 Changed: 0 Warnings: 0

mysql> quit
Bye

C:\Program Files\MySQL\MySQL Server 5.6\bin>mysql -u etibar
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 5.6.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

*show databases;* əmri ilə sistemdə hansı bazalar olduğunu görürük.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
| performance_schema |
+-----+
3 rows in set (0.00 sec)

mysql>
```

*create database if not exists mynewdb;* əmr sətirini yazmaqla **mynewdb** adında yeni baza yaratmış oluruq.

Burada *if not exists* yazmaqla biz sistem tərəfindən səhv almaq problemini aradan qaldırmış oluruq yəni əgər bu adda baza varsa yaradılacaq, yoxdursa sadəcə bizə xəbərdarlıq göndərəcəkdir.

Bundan əlavə hər hansı bazanı silmək üçün *drop database* əmrini istifadə edirik.

Desək ki elə bayaqki mynewdb bazasını silmək istəyirik, o zaman yazacağımız əmr sətiri belə olacaqdır :

*drop database if exists mynewdb;*

```
Administrator: C:\Windows\system32\cmd.exe - mysql -u root
mysql> create database if not exists mynewdb;
Query OK, 1 row affected (0.02 sec)
mysql> create database if not exists mynewdb;
Query OK, 1 row affected, 1 warning (0.00 sec)
mysql> create database mynewdb;
ERROR 1007 (HY000): Can't create database 'mynewdb'; database exists
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mynewdb |
| mysql |
| performance_schema |
+-----+
4 rows in set (0.00 sec)
mysql> use mynewdb;
Database changed
mysql> drop database if exists mynewdb;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
ear 'exists mynewdb' at line 1
mysql> drop database if exists mynewdb;
Query OK, 0 rows affected (0.02 sec)
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
+-----+
3 rows in set (0.00 sec)
mysql>
```

\*\*\*

## 2-ci darsin sonu

Növbəti darsin mövzusu :

*Cədvəllərdə sütun tipləri. Cədvəllərin yaradılması, silinməsi və məlumatların daxil edilməsi.*

Diqqətiniz üçün təşəkkürlər





Mərhəmətli və Rəhimli Allahın adı ilə



## MySQL Development Training

### 3-cü dərs

*Cədvəllərdə sütun tipləri. Cədvəllərin yaradılması, silinməsi və məlumatların daxil edilməsi.*



Təlimçi : *Etibar Vəzirov*

*Java Developer*

# Cədvəllərdə sütun tipləri

MySQL-də sütun tipləri 3 əsas kateqoriyaya ayrılır :

- *rəqəm tipli*
- *mətn (sətir) tipli*
- *vaxt və tarix*



## Rəqəm tipləri :

*tinyint*

*diapazonu : (-128; +127)*

*UNSIGNED (0..255)*

*smallint*

*diapazonu : (-32768; +32767) / 2 bayt*

*UNSIGNED (0..65535)*

*digər rəqəm ripləri :*

*mediumint, int, bigint, decimal, float, double*

## Mətn (Sətir) tipləri :

*CHAR(n)*

*VARCHAR(n)*

*binary*

*var binary*

*TEXT (böyük həcmli mətnlər üçün) /  
(tinytext, mediumtext və s)*

*BLOB (tinyblob, medium blob)*



## Vaxt və tarix tipləri :

*YEAR (year)*

*Date (day//month//year)*

*time (hour//minute//second)*

*datetime (date and time)*

*Timestamp (current time)*



## Cədvəllərin yaradılması və silinməsi

Öncəki dərslərdə əmrlər sətirindən (cmd-dən) mysql-ə daxil olmaq, mövcud bazaları göstərmək, baza yaratmaq və silməyi göstərmişdik. İndi isə bir baza yaradıb onun daxilində cədvəl yaradılması, silinməsi və həmçinin cədvəlin sətirinin çıxarılması işləmlərini edəcəyik.

***mysql -u root*** yazmaqla mysql-ə daxil oluruq və ***show databases;*** ilə mövcud bazaları göstəririk.

Aşağıdakı şəkildən görüldüyü kimi

***create database if not exists mysql\_training;***

sətiri ilə **mysql\_training** adlı bir baza yaratmışıq.

Bu bazaya qoşulmaq üçün **use mysql\_training;** yazırıq.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
+-----+
3 rows in set (0.00 sec)

mysql> create database if not exists mysql_training;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| mysql_training |
| performance_schema |
+-----+
4 rows in set (0.00 sec)

mysql> use mysql_training;
Database changed
```

İndi isə yeni bazamız daxilində çox sadə bir cədvəl yaradaq :

```
create table if not exists student(
name varchar(20),
surname varchar(30),
age tinyint unsigned,
birth_date date);
```

```
mysql> use mysql_training;
Database changed
mysql> create table if not exists student(
-> name varchar(20),
-> surname varchar(30),
-> age tinyint unsigned,
-> birth_date date)
-> ;
Query OK, 0 rows affected (0.29 sec)
```

Bundan sonra ***show create table student;*** yazmaqla yaratdığımız cədvəlin sintaksisini görə bilərik, həmçinin ***describe student;*** yazaraq da cədvəl görüntüsünə baxa bilərik :

```
mysql> show create table student;
+-----+-----+
| Table | Create Table |
+-----+-----+
| student | CREATE TABLE `student` (
  `name` varchar(20) DEFAULT NULL,
  `surname` varchar(30) DEFAULT NULL,
  `age` tinyint(3) unsigned DEFAULT NULL,
  `birth_date` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+-----+-----+
1 row in set (0.00 sec)

mysql> describe student;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name | varchar(20) | YES | | NULL | |
| surname | varchar(30) | YES | | NULL | |
| age | tinyint(3) unsigned | YES | | NULL | |
| birth_date | date | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```

Cari bazada olan cədvəlləri göstərmək üçün

*show tables;* yazmaq lazımdır:

```
mysql> show tables;
+-----+
| Tables_in_mysql_training |
+-----+
| student |
+-----+
1 row in set (0.00 sec)

mysql> show columns from student;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name | varchar(20) | YES | | NULL | |
| surname | varchar(30) | YES | | NULL | |
| age | tinyint(3) unsigned | YES | | NULL | |
| birth_date | date | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```

*show columns from student;* yazmaqla da **student** cədvəlini görüntüləmək olar.

Mövcud cəvəlin surətini çıxarmaq üçün

*create table copy\_student like student;*

yazmaqla *student* cədvəlinin *copy\_student* adında surətini çıxarmış oluruq :

```
mysql> create table copy_student like student;
Query OK, 0 rows affected (0.26 sec)

mysql> show tables;
+-----+
| Tables_in_mysql_training |
+-----+
| copy_student             |
| student                  |
+-----+
2 rows in set (0.00 sec)
```

Cədvəli silmək üçün drop table əmrini istifadə edirik :

*drop table if exists student;*

Bu əmr sətiri ilə biz *student* cədvəlini silmiş olacağıq :

```
mysql> drop table if exists student;
Query OK, 0 rows affected (0.07 sec)

mysql> show tables;
+-----+
| Tables_in_mysql_training |
+-----+
| copy_student             |
+-----+
1 row in set (0.00 sec)

mysql> _
```

Cədvəlin bazadan silindiğini yoxlamaq üçün yenidən *show tables;* yazıb buna əmin olmaq olar.

Cədvəlin adını dəyişmək də mümkündür, bunun üçün

*alter table copy\_student rename to employee;*

şəklində sorğu yazmaq lazımdır.Yəni cədvəlin yeni adı **employee** olacaqdır :

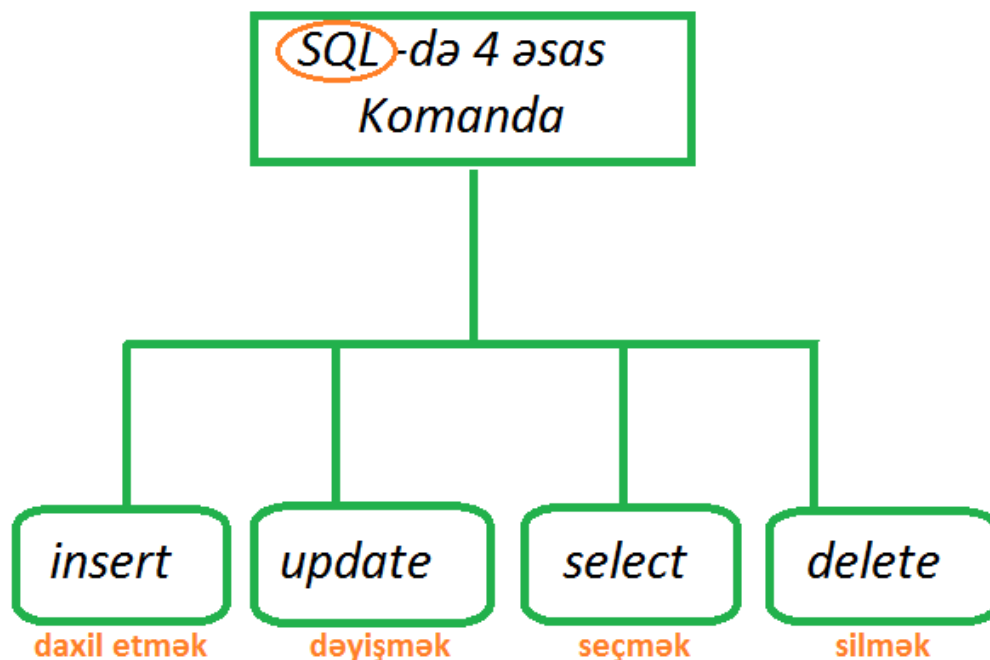
```
mysql> alter table copy_student rename to employee;
Query OK, 0 rows affected (0.10 sec)

mysql> show tables;
+-----+
| Tables_in_mysql_training |
+-----+
| employee                  |
+-----+
1 row in set (0.00 sec)

mysql>
```

## Cədvələ məlumatların daxil edilməsi (INSERT)

SQL dillərində (o cümlədən MySQL-də) 4 əsas əmr mövcuddur:



Bunlardan 1-cisi ilə (**INSERT**) bu dərs tanış olacağıq.

**INSERT** komandası ilə cədvələ məlumatlar daxil edilir.

Son yaratdığımız **employee** cədvəlinə məlumatlar daxil edək.

```
insert into employee(name,surname,age,birth_date)
values('Etibar', 'Vazirov', 27, '1989-07-29');
```

```
-----
insert into employee(name,surname,age,birth_date)
values('Fizuli', 'Ehmedov', 20, '1996-08-29');
```

Sonra daxil etdiyimiz məlumatları göstərmək üçün *select \* from employee;* yazmaq lazımdır.

```
mysql> insert into employee(name,surname,age,birth_date) values('Etibar','Vazirov',27,'1989-07-29');
Query OK, 1 row affected (0.06 sec)

mysql> insert into employee(name,surname,age,birth_date) values('Fizuli','Ehmedov',20,'1996-08-29');
Query OK, 1 row affected (0.07 sec)

mysql> select * from employee;
+-----+-----+-----+-----+
| name  | surname | age  | birth_date |
+-----+-----+-----+-----+
| Etibar | Vazirov | 27   | 1989-07-29 |
| Fizuli | Ehmedov | 20   | 1996-08-29 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Sintaksisdə (\*) cədvəldən bütün məlumatların görüntülənməsini təmin edir.



Əgər yalnız 1 sütün üçün məlumat daxil edərixsə o zaman digər sütün məlumatları *null* olaraq qəbul olunacaq (default deyer *null* olduğuna görə).

*insert into employee(surname) values('Alizade');*

```
mysql> insert into employee(surname) values('Alizade');
Query OK, 1 row affected (0.05 sec)

mysql> select * from employee;
+----+-----+-----+-----+
| name | surname | age | birth_date |
+----+-----+-----+-----+
| Etibar | Uazirov | 27 | 1989-07-29 |
| Fizuli | Ehmedov | 20 | 1996-08-29 |
| NULL | Alizade | NULL | NULL |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Cədvələ məlumatları hər sətir üçün ayrıca sorğu yazmadan da istədiyimiz qədər informasiyanı eyni zamanda daxil edə bilərik.Məsələn :

*insert into employee(name,surname,age,birth\_date) values('Cavid','Xelilov',20,'1996-08-12'), ('Azer','Mehdiyev',17,'1999-10-24');*

```
mysql> insert into employee(name,surname,age,birth_date) values('Cavid','Xelilov',20,'1996-08-12'),
-> ('Azer','Mehdiyev',17,'1999-10-24');
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> select * from employee;
+----+-----+-----+-----+
| name | surname | age | birth_date |
+----+-----+-----+-----+
| Etibar | Uazirov | 27 | 1989-07-29 |
| Fizuli | Ehmedov | 20 | 1996-08-29 |
| NULL | Alizade | NULL | NULL |
| Cavid | Xelilov | 20 | 1996-08-12 |
| Azer | Mehdiyev | 17 | 1999-10-24 |
+----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> _
```

\*\*\*

### **3-cü darsin sonu**

**Növbəti darsin mövzusu :**

*Verilənlərin saxlanılma sistemləri(Storage Engine).Primary key.Update və Delete əmrləri*

**Diqqətiniz üçün təşəkkürlər**



Mərhəmətli və Rəhimli Allahın adı ilə



## MySQL Development Training

### 4-cü dərs

*Verilənlərin saxlanılma sistemləri(Storage Engine).Primary key.Update və Delete əmrləri*

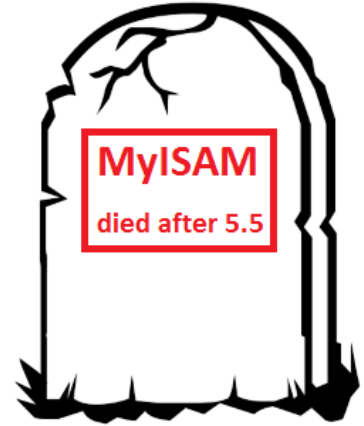


*Təlimçi : Etibar Vəzirov*

*Java Developer*

## Verilənlərin saxlanılma sistemləri (Storage Engine).

MySQL-in 3-cü versiyasından 5.5 versiyasına qədər əsas Storage Engine olaraq **MyISAM** istifadə olunurdu. MyISAM-in əsas üstün cəhəti o idi ki o platformalar arası keçid zamanı heç bir problem yaratmırdı və rahatlıqla inteqrasiya oluna bilirdi.



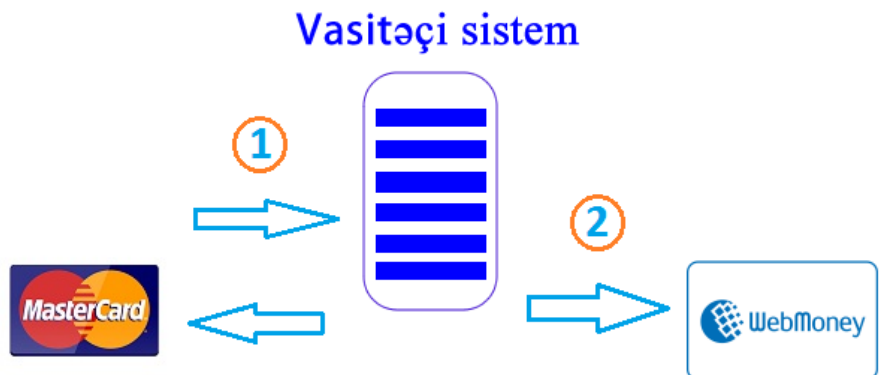
Həmçinin **MyISAM**-da select sorğusu çox sürətlə yerinə yetirilir. Bunun səbəbi **MyISAM**-in özünün **foreign key**-i dəstəkləmə qabiliyyətinin olmamasıdır.

**MyISAM**-in çatışmayan cəhəti isə onun **tranzaksiyaları** dəstəkləməməsidir.

### Tranzaksiya

haqqında biraz informasiya verək. Tranzaksiya birdən çox əməliyyatın vahid bir əməliyyat kimi qəbul

edilməsidir. Bu əməliyyatlar qrupundan hər hansı biri uğurla yerinə yetirilmədiyi halda **tranzaksiya** tamamlanmır.



MySQL-in 5.5 versiyasından sonra default engine olaraq **InnoDB** istifadə olunur. InnoDB-nin ən üstün cəhətləri özlüyündə **tranzaksiyaları** və **foreign key**-i dəstəkləməsidir.



Biz indi MySQL -in 5.6 versiyasını istifadə etdiyimizə görə bizim yaratdığımız cədvəl tipləri üçün storage engine InnoDB-dir. Məsələn employee cədvəlinin

**show create table employee;**

sorğusu ilə sintaksisinə nəzər salsaq engine olaraq InnoDB istifadə olduğunu görürük.

```
mysql> show create table employee;
+-----+-----+
| Table | Create Table |
+-----+-----+
| employee | CREATE TABLE `employee` (
  `name` varchar(20) DEFAULT NULL,
  `surname` varchar(30) DEFAULT NULL,
  `age` tinyint(3) unsigned DEFAULT NULL,
  `birth_date` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+-----+-----+
1 row in set (0.06 sec)
```

Bundan əlavə həmçinin cədvəlin özünün kodlaşma sistemi də olur. Məsələn **utf8, utf16, latin1** və s.

MySQL-də cədvəllərin standart kodlaşma sistemi **latin1**-dir. Bu yuxarıdakı nümunədən də görünür.

Onun dəyişilməsi qaydası ilə gəlin tanış olaq. Bunun üçün cədvəli yaradarkən sonda **default charset=utf8;** yazmaq lazımdır:

```
create table if not exists student(
name varchar(20),
surname varchar(30),
age tinyint unsigned) engine=InnoDB default charset=utf8;
```

```
mysql> create table if not exists student(
-> name varchar(20),
-> surname varchar(30),
-> age tinyint unsigned) engine=InnoDB default charset=utf8;
Query OK, 0 rows affected (0.20 sec)

mysql> show create table student;
+-----+-----+
| Table      | Create Table
+-----+-----+
| student    | CREATE TABLE `student` (
  `name` varchar(20) DEFAULT NULL,
  `surname` varchar(30) DEFAULT NULL,
  `age` tinyint(3) unsigned DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-----+-----+
```

## Primary key haqqında.

Cədvəlləri yaradarkən onların unikallığını (vahidliyini) təmin etmək lazımdır.

Tutaq ki aşağıdakı kimi cədvəlimiz var. Və bu cədvəldə mümkündür ki eyni ad , soyad və yaşa malik şəxslər qeyd olunsun. Bu zaman onları dəyişik salmamaq üçün bizə bir fərqləndirici açar lazımdır.



ID	NAME	SURNAME	AGE
1	Saleh	Əliyev	24
2	Nicat	Süleyman	29
3	Fizuli	Əhmədov	20
4	Saleh	Əliyev	24

Cədvəlin unikallığını təmin etmək üçün əlavə bir sütün - ID sütünü yaradılır. Bu sadəcə nömrələmə funksiyasını yerinə yetirəcəkdir. Və məhz bu id ilə biz yuxarıdakı cədvəldə **id -si 1** olan **Saleh Əliyev** -lə **id-si 4** olanı ayırd edə bilirik. Bu ID sütünü **primary key** rolunu oynayır.

Cədvəllərdə id-lər heç vaxt istifadəçi tərəfindən əllə daxil edilmir. MySQL -də bunun üçün xüsusi artırma funksionallığı **auto\_increment** vardır.

*Cədvəllərdə hansı sütünü unikal olaraq təyin etmək istəyiriksə onu primary key edirik. ID sütünü cədvəllərdə hər zaman primary key olaraq yaradılır.*

Sütunların özünün xüsusi atributları olur. məsələn **null** və **not null** kimi.

Hansı sütuna informasiyanın daxil edilməsi mütləqdirsə ona **not null** yazmaq lazımdır. Əgər bunu yazmırıqsa o zaman avtomatik olaraq default dəyər **null** kimi qəbul edilir.

```
create table if not exists student(  
id int not null auto_increment,  
name varchar(30) default 'unknown',  
surname varchar(40) default 'unknown',  
age int default 0,  
primary key(id)) engine=InnoDB charset=utf8;
```

```
mysql> create table if not exists student(
-> id int not null auto_increment,
-> name varchar(30) default 'unknown',
-> surname varchar(40) default 'unknown',
-> age int default 0,
-> primary key(id)) engine=InnoDB charset=utf8;
Query OK, 0 rows affected (0.23 sec)

ERROR 1064 (42000): You have an error in your SQL syntax; check the
error 'table student' at line 1
mysql> describe student;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(30)	YES		unknown	
surname	varchar(40)	YES		unknown	
age	int(11)	YES		0	

Cədvəli yaradarkən primary key -i təyin etmək üçün başqa üsül də vardır :

```
create table if not exists student(
id int not null auto_increment primary key,
name varchar(30) default 'unknown',
surname varchar(40) default 'unknown',
age int default 0)
engine=InnoDB charset=utf8;
```

Bu halda da yenə eyni cədvəli yaratmış olacağıq. Yaratdığımız cədvələ məlumatlar əlavə etməklə id-nin necə avtomatik artdığını görə bilərik:

```
mysql> insert into student(name,surname,age) values('Eli','Eliyev',20),
-> ('Azer','Kerimov',30),('Gunay','Semedova',22);
Query OK, 3 rows affected (0.05 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select * from student;
```

id	name	surname	age
1	Eli	Eliyev	20
2	Azer	Kerimov	30
3	Gunay	Semedova	22

3 rows in set (0.02 sec)

```
insert into student(name,surname,age) values('Eli','Eliyev',20),
('Azer','Kerimov',30), ('Gunay','Semedova',22);
```



## Update və Delete əməlləri

Bazaya daxil edilmiş məlumatları yeniləmək, üzərində dəyişiklik etmək lazımdırsa bu halda **update** əmrindən istifadə olunur.

```
mysql> select * from student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Eliyev  | 20  |
| 2  | Azer  | Kerimov | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Kenan    | 21  |
| 5  | Hysel | Elekberli | 18  |
| 7  | unknown | Qarayev | 21  |
+----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> update student set surname='Salahov' where id=4
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Eliyev  | 20  |
| 2  | Azer  | Kerimov | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov  | 21  |
| 5  | Hysel | Elekberli | 18  |
| 7  | unknown | Qarayev | 21  |
+----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Tutaq ki ad və soyadı yalnışlıqla eyni yazılmış adamın soyadını düzgün yazmaq lazımdır. Bunun üçün sorğu aşağıdakı kimi olacaq:

```
update student set surname='Salahov' where id = 4;
```

Aşağıdakı şəkildə isə yazdığımız sorğu ilə soyadı **Eliyev** olan bütün şəxslərin soyadları **Agayev** olaraq dəyişdirilir:

```
update student set surname='Agayev' where  
surname='Eliyev';
```

```
mysql> select * from student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Eliyev  | 20  |
| 2  | Azer  | Eliyev  | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov | 21  |
| 5  | Aysel | Elekberli | 18  |
| 7  | unknown | Qarayev | 21  |
+----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> update student set surname='Agayev' where surname='Eliyev';
Query OK, 2 rows affected (0.03 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20  |
| 2  | Azer  | Agayev  | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov | 21  |
| 5  | Aysel | Elekberli | 18  |
| 7  | unknown | Qarayev | 21  |
+----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Cədvəldə silinməsi lazım olan informasiyadan yaxa qurtarmaq üçün **delete** əmrindən istifadə olunur. Məsələn yuxarıdakı **student** cədvəlindən adı məlum olmayan şəxsi silmək istəyirəm. Bunun üçün sorğunu aşağıdakı kimi yazmaq lazımdır: **delete from student where id=7;**

```
mysql> select * from student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20  |
| 2  | Azer  | Agayev  | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov | 21  |
| 5  | Aysel | Elekberli | 18  |
| 7  | unknown | Qarayev | 21  |
+----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> delete from student where id=7;
Query OK, 1 row affected (0.06 sec)

mysql> select * from student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20  |
| 2  | Azer  | Agayev  | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov | 21  |
| 5  | Aysel | Elekberli | 18  |
+----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Əgər yuxarıdakı sorğuda şərt qoymasaydıq (yəni **where id=7;**

yazmasaydıq) o zaman sadəcə **delete from student;** sorğusu ilə cədvəldən bütün məlumatlar silinərdi. Bunu bir nümunədə göstərək. Student cədvəlinin surətini çıxarıb içinə məlumat yazmaq və onun üçün yuxarıda dediyimiz sorğunu icra edək.

```
mysql> create table test_student like student;
Query OK, 0 rows affected (0.22 sec)

ERROR 1146 (42S02): Table 'mysql_training.like_student' doesn't exist
mysql> select * from test_student;
Empty set (0.00 sec)

mysql> insert into test_student select * from student;
Query OK, 5 rows affected (0.03 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> select * from test_student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20  |
| 2  | Azer  | Agayev  | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov | 21  |
| 5  | Aysel | Elekberli | 18  |
+----+-----+-----+-----+
5 rows in set (0.00 sec)
```

**create table test\_student like student;** sorğusu ilə student cədvəlinin sturuktur formasını çıxardıq.(copy etdik). Lakin kontent boş olduğu üçün onun daxilinə informasiya yazmaq lazımdır:

**insert into test\_student select \* from student;** bu sorğu ilə cədvəlin içini doldurduq.İndi isə **delete from test\_student;** əmrinin nə etdiyinə baxaq.

Sağdakı şəkildən də görüldüyü kimi son yazdığımız sorğu cədvəli kontentini tamamilə silmiş oldu.Ona görə də delete əmri ilə işləyərkən maksimum diqqətli olmaq lazımdır.

```
mysql> delete from test_student;
Query OK, 5 rows affected (0.06 sec)

mysql> select * from test_student;
Empty set (0.00 sec)

mysql>
```

\*\*\*

**4-cü darsin sonu**

**Növbəti darsin mövzusu :**

*Select sorgusu və onunla birlikdə  
işlənən əmrlər*

**Diqqətiniz üçün təşəkkürlər**



Mərhəmətli və Rəhimli Allahın adı ilə



## MySQL Development Training

### 5-ci dərs

*Select sorğusu və onunla birlikdə  
işlənən əmrlər*



*Təlimçi : Etibar Vəzirov*

*Java Developer*

## Cədvəldən məlumatların seçilməsi.

### SELECT sorgusu.

**SELECT** əmri ilə verilənlər bazasından məlumatlar seçilir.

**Select** sözündən sonra seçmək istədiyimiz sütünün adı yazılır. Əgər bütün sütunları seçmək istəyiriksə o zaman **select-**dən sonra (\*) yazılır. Məsələn :

**select \* from student;**

```
mysql> select * from student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20  |
| 2  | Azer  | Agayev  | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov  | 21  |
| 5  | Aysel | Elekberli | 20  |
+----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> _
```

Cədvəldən bütün sütunların deyil yalnız biz istədiyimiz konkret sütunları görmək istəyiriksə , **select** sözündən sonra həmin sütunların adları sadalanır :

**select name,age from student;**

```
mysql> select name,age from student;
+-----+-----+
| name  | age |
+-----+-----+
| Eli   | 20  |
| Azer  | 30  |
| Gunay | 22  |
| Kenan | 21  |
| Aysel | 20  |
+-----+-----+
5 rows in set (0.00 sec)
```

Select sorgusunu müəyyən bir şərtlə də vermək olar.

Məsələn biz **student** cədvəlindən soyadı **Agayev** olanları seçmək istəyirik. Bunun üçün bir şərt yazılmalıdır:

**select \* from student where surname='Agayev';**

```
mysql> select * from student where surname='Agayev';
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20  |
| 2  | Azer  | Agayev  | 30  |
+----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Select əmri ilə bir sıra sadə sorğular yazmaq olar.Məsələn

**student** cədvəlində adı

Kenan olan şəxsin

yaşını öyrənmək

istəyirik:

**select age from student  
where name='Kenan';**

```
mysql> select age from student where name='Kenan';  
+-----+  
| age |  
+-----+  
| 21 |  
+-----+  
1 row in set (0.00 sec)  
mysql>
```

**Select** sorğusu və **where** konstruksiyası ilə birlikdə müəyyən məhdudiyətlər(**constraints**) qoymaq mümkündür.

Məsən biz yuxarıdakı cədvəldən soyadı Agayev və yaşı 20 olan şəxsi seçək istəyiriksə o zaman sorğunu aşağıdakı kimi yazacağıq :

**select \* from student where surname='Agayev' and age='20';**

```
mysql> select * from student where surname='Agayev' and age='20';  
+----+-----+-----+-----+  
| id | name | surname | age |  
+----+-----+-----+-----+  
| 1 | Eli | Agayev | 20 |  
+----+-----+-----+-----+  
1 row in set (0.00 sec)  
mysql> _
```

Əgər

yuxarıdakı sorğuda **and** operatoru yerinə **or** yazsaydıq fərqli nəticə alardıq:

**select \* from student where surname='Agayev' or age='20';**

```
mysql> select * from student where surname='Agayev' or age='20';  
+----+-----+-----+-----+  
| id | name | surname | age |  
+----+-----+-----+-----+  
| 1 | Eli | Agayev | 20 |  
| 2 | Azer | Agayev | 30 |  
| 5 | Aysel | Elekberli | 20 |  
+----+-----+-----+-----+  
3 rows in set (0.00 sec)
```

İlk dərsdə demişdik ki relyasiyalıq çoxluqlar nəzəriyyəsi və predikatlar(şərtlər) məntiqinə əsaslanır. Burada **and** operatoru iki şərti bir araya gətirməklə çoxluqların kəsişməsini, **or** isə həmin çoxluqların birləşməsini göstərir. *(riyaziyyatçılar daha yaxşı anlayar :))*

Məlumdur ki **select \* from student;** komandası ilə cədvəldə olan bütün məlumatları görə bilərik.

(qeyd: izah üçün cədvələ bir neçə məlumat daxil etmişəm)

```
mysql> select * from student;
```

id	name	surname	age
1	Eli	Agayev	20
2	Azer	Agayev	30
3	Gunay	Semedova	22
4	Kenan	Salahov	21
5	Aysel	Elekberli	20
6	Nicat	Suleyman	29
7	Fizuli	Ehmedov	20
8	Nermin	Orucova	19

8 rows in set (0.00 sec)

Lakin tutuq ki bizə yalnız 6 istifadəçinin məlumatını görmək lazımdır. Bu halda ilk olaraq düşünülə bilər ki **select \* from student where id<7;** yazmaqla bunu etmək olar.

```
mysql> select * from student where id<7;
```

id	name	surname	age
1	Eli	Agayev	20
2	Azer	Agayev	30
3	Gunay	Semedova	22
4	Kenan	Salahov	21
5	Aysel	Elekberli	20
6	Nicat	Suleyman	29

6 rows in set (0.00 sec)

Lakin id sıralaması düzgün olmazsa, məsələn id-si 2 olan şəxs cədvəldən silinərsə o zaman yuxarıdakı sorğu bizə 6 deyil 5 sətir məlumat göstərəcək.(yəni yalnız id-si 7-dən kiçik olanlar 1,3,4,5,6 göstəriləcək).Amma bizə sorğu zamanı dəqiq olaraq 6 istifadəçinin göstərilməsi lazımdır. Bunun üçün sorğunu aşağıdakı şəkildə yazmaq daha düzgündür:

**select \* from student limit 6;**  
**limit** operatoru ilə hər hansı sətirin olub olmamasından asılı olmayaraq lazımi nəticəni ala bilərik.

```
mysql> select * from student limit 6;
```

id	name	surname	age
1	Eli	Agayev	20
2	Azer	Agayev	30
3	Gunay	Semedova	22
4	Kenan	Salahov	21
5	Aysel	Elekberli	20
6	Nicat	Suleyman	29

6 rows in set (0.00 sec)



**Limit** komandasının həmçinin **ikili sintaksisi** vardır.Məsələn **select \* from student limit 2,3;** yazsaq bu sorğu ilə 2-ci sətirdən sonra 3 sətiri göstərmiş oluruq.

```
mysql> select * from student limit 2,3;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov  | 21  |
| 5  | Aysel | Elekberli | 20  |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

## SELECT sorğusu ilə birlikdə işlənən digər əmrlər : DISTINCT, IN, BETWEEN, LIKE, ORDER BY

Sorğu yerinə yetirilərkən təkrarlanan məlumatlar ala bilərik. Bu təkrarlanmanın qarşısını almaq üçün **DISTINCT** operatoru istifadə olunur.

Cədvəlimizdə təkrarlanan Agayev soyadı vardır.Biz unikal olaraq soyadların listlənməsini istəyiriksə **select distinct surname from student;** yaza bilərik.

```
mysql> select * from student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20  |
| 2  | Azer  | Agayev  | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov  | 21  |
| 5  | Aysel | Elekberli | 20  |
| 6  | Nicat | Suleyman | 29  |
| 7  | Fizuli | Ehmedov | 20  |
| 8  | Nermin | Orucova | 19  |
+----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> select distinct surname from student;
+-----+
| surname |
+-----+
| Agayev  |
| Semedova |
| Salahov  |
| Elekberli |
| Suleyman |
| Ehmedov  |
| Orucova  |
+-----+
7 rows in set (0.00 sec)
```

**IN** operatoru vasitəsi ilə bir neçə **OR** komandasını əvəz etmək olur. Məsələn, id-si 3,4,6 olan istifadəçiləri görmək istəyirik. Bu halda sorğumuz belə olacaq:

```
mysql> select * from student where id in(3,4,6);
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 3  | Gunay | Samedova | 22 |
| 4  | Kenan | Salahov  | 21 |
| 6  | Nicat | Suleyman | 29 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from student where id=3 or id=4 or id=6;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 3  | Gunay | Samedova | 22 |
| 4  | Kenan | Salahov  | 21 |
| 6  | Nicat | Suleyman | 29 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

**select \* from student where id in(3,4,6);**  
və ya **select \* from student where id=3 or id=4 or id=6;**

Elementləri sadalamaq üçün həmçinin **OR** komandasından istifadə olunur ki **IN** operatoru da bu işi bizim üçün sadələşdirir.

MySQL-də həmçinin **>** (böyükdür), **<** (kiçikdir), **<>** (fərqlidir) kimi şərt komandalarından da istifadə etmək mümkündür.

```
mysql> select * from student where id>3;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 4  | Kenan | Salahov  | 21 |
| 5  | Aysel | Elekberli | 20 |
| 6  | Nicat | Suleyman | 29 |
| 7  | Fizuli | Ehmedov  | 20 |
| 8  | Nermin | Orucova  | 19 |
+----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> select * from student where id<3;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20 |
| 2  | Azer  | Agayev  | 30 |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from student where id<>3;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20 |
| 2  | Azer  | Agayev  | 30 |
| 4  | Kenan | Salahov  | 21 |
| 5  | Aysel | Elekberli | 20 |
| 6  | Nicat | Suleyman | 29 |
| 7  | Fizuli | Ehmedov  | 20 |
| 8  | Nermin | Orucova  | 19 |
+----+-----+-----+-----+
7 rows in set (0.00 sec)
```

Cədvəllərdə müəyyən aralıqda informasiyanı seçmək üçün

**BETWEEN** operatoru istifadə olunur.

```
mysql> select * from student where id between 3 and 5;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 3  | Gunay | Samedova | 22 |
| 4  | Kenan | Salahov  | 21 |
| 5  | Aysel | Elekberli | 20 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

**select \* from student where id between 3 and 5;**  
id-si 3-lə 5 arasında olan şəxsləri göstərəcək.

Bəzən cədvəldə olan məlumatlar dəqiq yadımızda olmaya bilər(və ya bir qismini unuda bilərik).Bu halda **like** komandası işimizə yarıya bilər.Axtardığımızı uyğun informasiya

sonda gəldikdə **like '%src\_word'** şəklində,

əvvəldə gəldikdə **like 'src\_word%'** şəklində,

ortada gəldikdə isə **like '%src\_word%'** formasında yazırıq.

Burada **src\_word** yerində axtardığımız informasiyanın yadımızda qalan hissəsini yazırıq.

```
mysql> select name,surname,age from student where surname like '%man';
+-----+-----+-----+
| name  | surname | age |
+-----+-----+-----+
| Nicat | Suleyman | 29 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select name,surname,age from student where name like 'ay%';
+-----+-----+-----+
| name  | surname | age |
+-----+-----+-----+
| Aysel | Elekberli | 20 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select name,surname,age from student where surname like '%med%';
+-----+-----+-----+
| name  | surname | age |
+-----+-----+-----+
| Gunay | Samedova | 22 |
| Fizuli | Ehmedov  | 20 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

Soyadının sonu man ilə bitən şəxslərin ad, soyad və yaşını göstərir :

```
select name,surname,age from student where surname like '%man';
```

Adının əvvəli ay ilə başlayan şəxslərin ad, soyad və yaşını göstərir :

```
select name,surname,age from student where name like 'ay%';
```

Soyadının içində med olan şəxslərin ad, soyad və yaşını göstərir :

```
select name,surname,age from student where surname like '%med%';
```

**ORDER BY** konstruksiyası cədvəldə müəyyən elementə görə sıralanmaq funksiyasını yerinə yetirir.Məsələn biz student cədvəlimizdə şəxslərin yaşa görə sıralanmasını istəyiriksə

onda sorğunu bu şəkildə yazacağıq:

```
select * from student order by age;
```

Əks qaydada sıralanma üçün isə sonda sadəcə **desc** (descending) yazmaq lazımdır:

```
select * from student order by age desc;
```

```
mysql> select * from student order by age;
```

id	name	surname	age
8	Nermin	Orucova	19
1	Eli	Agayev	20
5	Aysel	Elekberli	20
7	Fizuli	Ehmedov	20
4	Kenan	Salahov	21
3	Gunay	Semedova	22
6	Nicat	Suleyman	29
2	Azer	Agayev	30

8 rows in set (0.00 sec)

```
mysql> select * from student order by age desc;
```

id	name	surname	age
2	Azer	Agayev	30
6	Nicat	Suleyman	29
3	Gunay	Semedova	22
4	Kenan	Salahov	21
1	Eli	Agayev	20
5	Aysel	Elekberli	20
7	Fizuli	Ehmedov	20
8	Nermin	Orucova	19

Ümumiyyətlə MySQL-də **ASC** (**ascending/artan**) və **DESC** (**descending/azalan**) əmrləri vardır ki onlarla istənilən sütünun məlumatlarına görə sıralama aparmaq mümkündür.

```
mysql> select * from student order by name desc;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 6  | Nicat | Suleyman | 29  |
| 8  | Nermin | Orucova  | 19  |
| 4  | Kenan  | Salahov  | 21  |
| 3  | Gunay  | Semedova | 22  |
| 7  | Fizuli | Ehmedov  | 20  |
| 1  | Eli    | Agayev   | 20  |
| 2  | Azer   | Agayev   | 30  |
| 5  | Aysel  | Elekberli | 20  |
+----+-----+-----+-----+
8 rows in set (0.00 sec)
```

Burada **select \* from student order by name desc;** sorğusu ilə biz adların əlifba sırasına uyğun əks qaydada düzülüşünü aldıq.

### Aqreqat funksiyalar.(Aggregate functions).

MySQL-də **aqreqat funksiyalar** müvafiq sütündəki qiymətlərə görə hesablama aparıb nəticədə tək bir dəyər qaytaran funksiyalara deyilir. Praktikada ən çox əhəmiyyətli bir neçə aqreqat funksiyaya nəzər yetirək:

**COUNT()** - sətirlərin sayını verir

**AVG()** - orta qiymət verir

**FIRST()** və **LAST()** - ilk və son qiyməti qaytarır(mysql-də bunları **LIMIT** funksiyası əvəz edir)

**MAX()** və **MIN()** - ən böyük və ən kiçik qiyməti verir

```
mysql> select * from student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20  |
| 2  | Azer  | Agayev  | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov | 21  |
| 5  | Aysel | Elekberli | 20  |
| 6  | Nicat | Suleyman | 29  |
| 7  | Fizuli | Ehmedov | 20  |
| 8  | Nermin | Orucova | 19  |
+----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> select SUM(age) from student;
+-----+
| SUM(age) |
+-----+
| 181      |
+-----+
1 row in set (0.00 sec)

mysql> select AVG(age) from student;
+-----+
| AVG(age) |
+-----+
| 22.6250  |
+-----+
1 row in set (0.00 sec)

mysql> select MAX(age) from student;
+-----+
| MAX(age) |
+-----+
| 30       |
+-----+
1 row in set (0.00 sec)
```

sətirlərin sayını göstərir

**select count(DISTINCT surname) from student;**  
muxtəlif soyadların sayını qaytarır.

```
mysql> select count(*) from student;
+-----+
| count(*) |
+-----+
| 8        |
+-----+
1 row in set (0.00 sec)

mysql> select count(DISTINCT surname) from student;
+-----+
| count(DISTINCT surname) |
+-----+
| 7                        |
+-----+
1 row in set (0.00 sec)
```

**SUM()** - cəmi verir.

**select SUM(age) from student;**  
yaşların cəmini qaytarır  
**select AVG(age) from student;**  
yaşların ədədi ortasını qaytarır  
**select MAX(age) from student;**  
ən böyük yaşı seçir  
**select count(\*) from student;**

\*\*\*

**5-ci darsin sonu**

**Növbəti darsin mövzusu :**

*MySQL-də müvəqqəti cədvəllər. Index  
və foreign key*

**Diqqətiniz üçün təşəkkürlər**



Mərhəmətli və Rəhimli Allahın adı ilə



## MySQL Development Training

### 6-cı dər

*MySQL-də müvəqqəti cədvəllər. Index  
və foreign key*



Təlimçi : **Etibar Vəzirov**

*Java Developer*



## Müvəqqəti cədvəllərin yaradılması.

MySQL-də **müvəqqəti cədvəllər** 3.23 versiyasından sonra əlavə olunmuşdur. Bu cədvəllər elə xüsusi cədvəl növüdür ki onlar **müvəqqəti informasiyanı saxlamaq üçün** nəzərdə tutulmuşdur. Bəzən böyük həcmdə informasiyamız ola bilər ki onları müvəqqəti yadda saxlamaq lazım olduğu üçün bazada daimi cədvələ yazmaq lazım gəlmir. Bu zaman müvəqqəti cədvəl yaratmaq çox işimizə yaraya bilər.

MySQL sessiya bitdikdə və ya mysql serverlə əlaqə kəsildikdə avtomatik müvəqqəti cədvəli silir.

Müvəqqəti cədvəl bazada mövcud olan daimi cədvəllə eyni ada malik ola bilər, daimi cədvəllər isə eyni ada malik ola bilməzlər. Lakin mövcud cədvəllə eyni adda müvəqqəti cədvəl yaratmaq məsləhət olunan deyil. Çünki ola bilər ki MySQL database serverində connection itib yenidən avtomatik bərpa olunsun və biz bunu bilmədən müvəqqəti cədvəli silmək üçün **drop table** komandası yazsaq. Bu zaman özümüz də bilmədən bazada eyni adlı daimi cədvəli silmiş olacağıq.

Müvəqqəti cədvəllər **create temporary table** sintaksisi vasitəsilə yaradılır. Silmək üçünsə adi qaydada **drop temporary table** yazmaq lazımdır.

Birlikdə bir müvəqqəti cədvəli yaradılmasına baxaq.

create temporary table  
**my\_temp\_table**(  
**id** int not null primary key auto\_increment,  
**name** varchar(20) not null,  
**surname** varchar(30) not null,  
**age** int not null);

```
mysql> create temporary table my_temp_table(  
-> id int not null primary key auto_increment,  
-> name varchar(20) not null,  
-> surname varchar(30) not null,  
-> age int not null);  
Query OK, 0 rows affected (0.16 sec)  
  
mysql> show tables;  
+-----+  
| Tables_in_mysql_training |  
+-----+  
| student                  |  
| test_student             |  
+-----+  
2 rows in set (0.00 sec)  
  
mysql> select * from my_temp_table;  
Empty set (0.00 sec)
```

**my\_temp\_table** adlı müvəqqəti cədvəlimiz yaratdıq. Lakin **show tables;** yazdıqda bu cədvəlin adını siyahıda görə bilmirik çünki o müvəqqəti cədvəldir.

**select \* from my\_temp\_table;** yazmaqla cədvəldə hələ heç bir məlumatın olmadığını görürük.

Bu cədvələ adi qaydada məlumatlar daxil edə bilərik hansı ki ki insert komandası ilə siz artıq tanış olmusunuz.

```
mysql> insert into my_temp_table(name,surname,age) values('Etibar','Uezirov',27),  
-> ('Nicat','Suleyman',29),('Fizuli','Ehmedov',20),('Cavid','Kelilov',20);  
Query OK, 4 rows affected (0.07 sec)  
Records: 4 Duplicates: 0 Warnings: 0  
  
mysql> select * from my_temp_table;  
+----+-----+-----+-----+  
| id | name  | surname | age |  
+----+-----+-----+-----+  
| 1  | Etibar | Uezirov | 27  |  
| 2  | Nicat  | Suleyman | 29  |  
| 3  | Fizuli | Ehmedov | 20  |  
| 4  | Cavid  | Kelilov | 20  |  
+----+-----+-----+-----+  
4 rows in set (0.00 sec)
```

**drop temporary table my\_temp\_table;** yazmaqla bu müvəqqəti cədvəli silirik.

MySQL-də əldə etdiyimiz müəyyən nəticələri qruplaşdırmaq üçün **GROUP BY** komadasından istifadə edirik. **GROUP BY** ilə **count(\*)** aqreqat funksiyası geniş istifadə olunur və o qruplaşdırılan elementlərin sayını göstərir. **count(\*)**- dan sonra **as column\_name** alyasından istifadə olunur. Çünki **count(\*)** yazmaqla qruplaşdırılan elementlərin sayı adsız olaraq bir sütunda göstəriləcək, lakin **as column\_name** yazmaqla həmin sütuna ad vermiş oluruq. Gəlin bunu praktik nümunədə göstərək.

```
mysql> select * from student;
+----+-----+-----+-----+
| id | name  | surname | age |
+----+-----+-----+-----+
| 1  | Eli   | Agayev  | 20  |
| 2  | Azer  | Agayev  | 30  |
| 3  | Gunay | Semedova | 22  |
| 4  | Kenan | Salahov  | 21  |
| 5  | Aysel | Elekberli | 20  |
| 6  | Nicat | Suleyman | 29  |
| 7  | Fizuli | Ehmedov  | 20  |
| 8  | Nermin | Orucova  | 19  |
| 9  | Eldar | Ehmedov  | 24  |
| 10 | Gulnar | Orucova  | 25  |
| 11 | Rasin  | Agayev  | 26  |
+----+-----+-----+-----+
11 rows in set (0.00 sec)

mysql> select name,surname,count(*) as say from student group by surname;
+-----+-----+-----+
| name  | surname | say |
+-----+-----+-----+
| Eli   | Agayev  | 3   |
| Fizuli | Ehmedov  | 2   |
| Aysel | Elekberli | 1   |
| Nermin | Orucova  | 2   |
| Kenan | Salahov  | 1   |
| Gunay | Semedova | 1   |
| Nicat | Suleyman | 1   |
+-----+-----+-----+
7 rows in set (0.00 sec)
```

**select name,surname,count(\*) as say from student group by surname;**

Sorğusu ilə **student** cədvəlindən soyada görə qruplaşdırma apardıq və qruplaşdırılan elementlərin sayını ayrıca **say** sütünündə göstərdik.

**count** funksiyasından əlavə **substring\_index** funksiyası da vardır ki sözün müəyyən simvola qədər yazılışını kəşib çıxarmaq üçün istifadə etmək olar. Məsələn deyək ki **student** cədvəlində **Semedova** soyadında **2-ci e-yə qədər hissəni** kəşib göstərmək lazımdır:

```
mysql> select substring_index(surname,'e',2) as cut_surname from student where surname='Semedova';
+-----+
| cut_surname |
+-----+
| Sem         |
+-----+
1 row in set (0.00 sec)
```

**select substring\_index(surname,'e',2) as cut\_surname from student where surname='Semedova';**

Sorğunun nəticəsi "**Sem**" olaraq görünəcəkdir.

## **INDEX və FOREIGN KEY (xarici açar)**

Cədvəllərdə axtarış prosesini sürətləndirmək üçün **indeks**lərdən istifadə olunur. Əgər cədvəldə **indeks**ləmə yoxdursa onda select sorğusu ilə verilən axtarış bütün sətir və sütunlara görə aparılacaq ki bu da sorğunun gec icra olunmasına səbəb olacaq. Lakin cədvəldə biz müvafiq **indeks**lər yaratdıqda *axtarış prosesini xeyli sürətləndirmiş*

oluruq. **İndeks**ləri bir növ **kitablardakı mündəricata** bənzətmək olar, belə ki kitabda mündəricat olduqda biz istədiyimiz kontenti tez tapa bilirik.

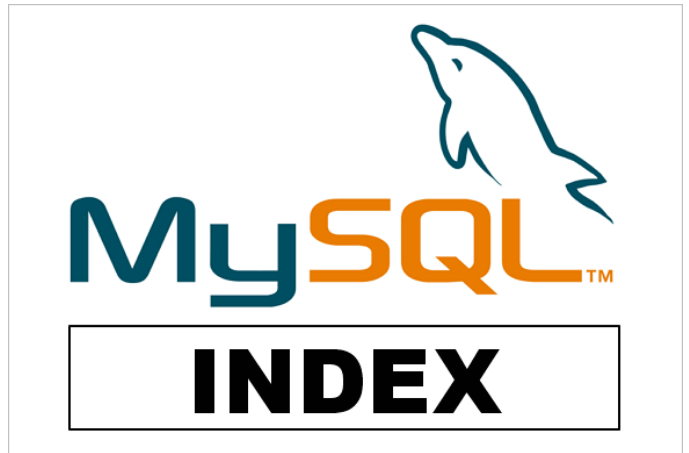
Bazada biz hər hansı bir cədvəl üçün **indeks** yaradan

zaman yeni bir cədvəl yaranır və öncəki cədvəldəki məlumatlar **indeks** yaradılan cədvəldə düzgün ardıcılıqla sadalanır. Yəni hansı sütun üçün **indeks** yaratmışıqsa həmin sütunda məlumatlar düzgün ardıcılıqla yerləşmiş olur. MySQL sorğu verilən zaman **indeks** olub olmadığını yoxlayır və əgər varsa **indeks** olan cədvələ gedib məlumatı daha tez tapır.

**İndeks**lərin həm **müsbət** həm də **mənfi** tərəfləri vardır.

Əgər biz öncəki cədvəl üçün **insert** və **update** əməliyyatları yerinə yetirəcəyiksə paralel olaraq həmin əməliyyatlar **indeks** yaradılmış cədvəldə də gedəcəkdir. *Lakin indeksə görə bu əməliyyatların sürəti xeyli zəifləyəcək. Beləliklə indekslərin müsbət tərəfi cədvəldə **select** sorğusunun daha sürətli icra olunması, mənfi tərəfi isə **insert** və **update** əməliyyatlarının zəif icra olunmasıdır.*

**İndeks**ləri daha yaxşı anlamaq üçün bir nümunə üzərində izah edək. **Employee** adında işçilər cəvəli yaradıb orada hər bir işçi üçün olan **kart nömrəsini** unikal indekslə verək.



create table if not exists

employee(  
id int not null  
auto\_increment primary  
key,  
name varchar(20) not  
null,  
surname varchar(30) not  
null,  
cardNumber varchar(25) not null,  
unique index(cardNumber));

sorğusu ilə employee cədvəli yaratdıq.

Həmçinin **show create table employee;** yazmaqla cədvəlin sintaksisində cardNumber-in **unikal indeks** olduğunu görə bilərik.

Bu cədvələ müxtəlif işçilər üçün eyni kart nömrəsi daxil etmək istədikdə error-la qarşılaşacağıq.

```
mysql> create table if not exists employee(  
-> id int not null auto_increment primary key,  
-> name varchar(20) not null,  
-> surname varchar(30) not null,  
-> cardNumber varchar(25) not null,  
-> unique index(cardNumber));  
Query OK, 0 rows affected (0.26 sec)  
mysql> show create table employee;  
+-----+-----+  
| Table           | Create Table  
+-----+-----+  
| employee        | CREATE TABLE `employee` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `name` varchar(20) NOT NULL,  
  `surname` varchar(30) NOT NULL,  
  `cardNumber` varchar(25) NOT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `cardNumber` (`cardNumber`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |  
+-----+-----+
```

```
mysql> insert into employee(name,surname,cardNumber) values('Nicat','Suleyman','123-456-78-abcd');  
Query OK, 1 row affected (0.06 sec)  
mysql> insert into employee(name,surname,cardNumber) values('Fizuli','Ehmedov','123-456-78-abcd');  
ERROR 1062 (23000): Duplicate entry '123-456-78-abcd' for key 'cardNumber'  
mysql>
```

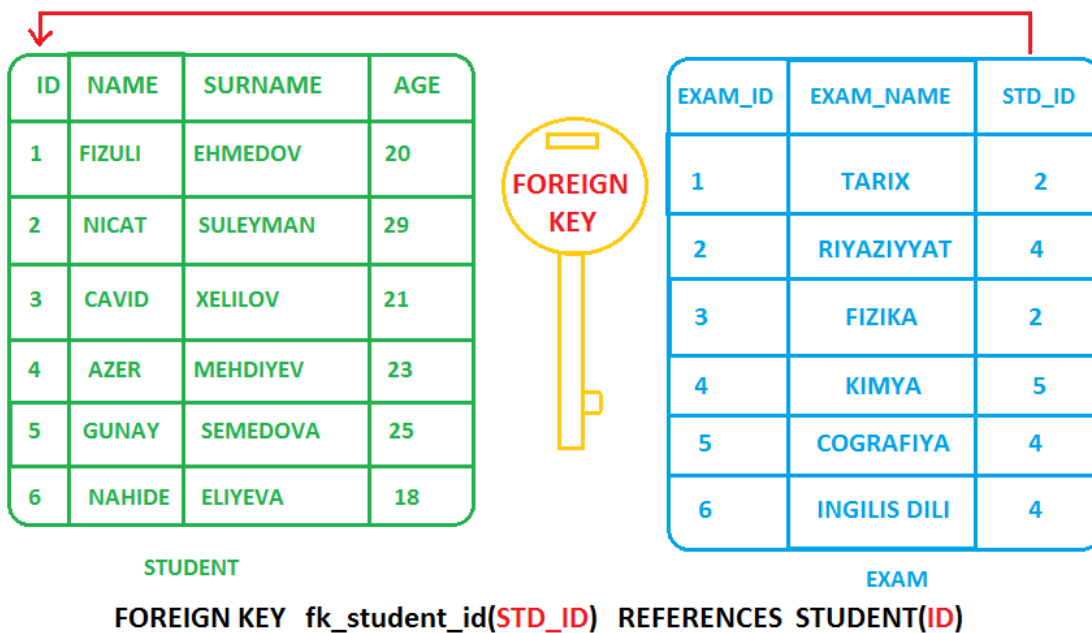
sadə **indeks** yaratmaq üçün sadəcə **unique** sözünü ötürüb **index** yazmaq kifayətdir. Məsələn yuxarıdakı nümunədə

sadəcə **index(cardNumber)** kimi də sadə indeks yaratmaq olardı.

*İndekslər MySQL cədvəllərində performansını yüksəltmək üçün istifadə olunan bir konsepsiyadır lakin biz əgər cədvəlimizdə daimi olaraq daxil etmə və yeniləmə əməliyyatları aparmağa məcburuqsa onda indeks istifadə etmək o qədər də məsləhət olunan deyildir.*

## FOREIGN KEY.

Bir cədvəldə **foreign key** xarici açarı başqa cədvəlin **primary key** daxili açarını göstərir.



Tutaq ki bizdə yuxarıdakı kimi **student** və **exam** cədvəlləri var. Exam cədvəlinin **STD\_ID** sütunu **student** cədvəlinin **ID** sütunundan asılıdır və yalnız oradan olan **id**-ləri saxlaya bilər. **Exam** cədvəli **student** cədvəlinin bir hissəsi kimidir və bu cədvəl özündə xarici bir cədvəlin (**student** cədvəlinin)

sütununun dəyərlərini saxlayır. Ona görə **STD\_ID** -yə xarici açar yəni **foreign key** deyilir.

Yuxarıdakı şəkilli izahı real bir nümunə ilə göstərək. Yuxarıdakı kimi student cədvəlindən özündə xarici açar saxlayan Exam cədvəli yaradaq.

```
create table if not exists exam(  
exam_id int not null auto_increment primary key,  
exam_name varchar(20) default null,  
STD_ID int not null,  
FOREIGN KEY fk_student_id(STD_ID) references  
student(id));
```

```
mysql> create table if not exists exam(  
-> exam_id int not null auto_increment primary key,  
-> exam_name varchar(20) default null,  
-> STD_ID int not null,  
-> FOREIGN KEY fk_student_id(STD_ID) references student(id));  
Query OK, 0 rows affected (0.30 sec)  
mysql> show create table exam;  
+-----+  
+-----+  
| Table | Create Table  
+-----+  
+-----+  
+-----+  
| exam | CREATE TABLE `exam` (  
  `exam_id` int(11) NOT NULL AUTO_INCREMENT,  
  `exam_name` varchar(20) DEFAULT NULL,  
  `STD_ID` int(11) NOT NULL,  
  PRIMARY KEY (`exam_id`),  
  KEY `fk_student_id` (`STD_ID`),  
  CONSTRAINT `exam_ibfk_1` FOREIGN KEY (`STD_ID`) REFERENCES `student` (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

foreign key başqa cədvəlin(ana cədvəlin) müvafiq sütunundan asılı olduğuna görə ana cədvəldə bu sütunun elementlərində dəyişiklik oluna bilər(yəni vacib deyil ki ana cədvəlin sütunu primary key olsun). Bu halda gərək asılı



olan cədvəldəki **foreign key** də avtomatik olaraq dəyişilmiş olsun. Bunun üçün mysql-də əlavə olunmuş konstruksiya vardır. Bu **on delete cascade** və **on update cascade** konstruksiyasıdır. Əgər biz **exam** cədvəlini silib bu dediyimiz normalizasiya qaydalarına uyğun olaraq bir daha yaratsaq gərək sorğunu aşağıdakı kimi yazaq:

```
create table if not exists exam(  
exam_id int not null auto_increment primary key,  
exam_name varchar(20) default null,  
STD_ID int not null,  
FOREIGN KEY fk_student_id(STD_ID) references  
student(id) on delete cascade on update cascade);
```

```
mysql> drop table exam;  
Query OK, 0 rows affected (0.09 sec)  
  
mysql> create table if not exists exam(  
-> exam_id int not null auto_increment primary key,  
-> exam_name varchar(20) default null,  
-> STD_ID int not null,  
-> FOREIGN KEY fk_student_id(STD_ID) references student(id) on delete cascade on update cascade);  
Query OK, 0 rows affected (0.23 sec)  
  
mysql> show create table exam;  
+-----+  
| Table | Create Table  
+-----+  
| exam | CREATE TABLE `exam` (  
  `exam_id` int(11) NOT NULL AUTO_INCREMENT,  
  `exam_name` varchar(20) DEFAULT NULL,  
  `STD_ID` int(11) NOT NULL,  
  PRIMARY KEY (`exam_id`),  
  KEY `fk_student_id` (`STD_ID`),  
  CONSTRAINT `exam_ibfk_1` FOREIGN KEY (`STD_ID`) REFERENCES `student` (`id`) ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

\*\*\*

## 6-cı darsın sonu

Növbəti darsın mövzusu :

*Atomarlıq prinsipləri.*

*DDL, DML və DQL anlayışları*

**Diqqətiniz üçün təşəkkürlər**



Mərhəmətli və Rəhimli Allahın adı ilə



## MySQL Development Training

### 7-ci dər

*Cədvəllərdə atomarlıq prinsipi.*

*DDL, DML və DQL anlayışları.*



*Təlimçi : Etibar Vəzirov*

*Java Developer*

## ACID prinsiplər. Atomarlıq.



**ACID** prinsiplər **VBİS (verilənlər bazasını idarəetmə sistemləri)** üçün əsas işləm xüsusiyyətləridir və onlar olmadan verilənlər bazasının tamlığı (bütövlüyü) təmin edilə bilməz. Aşağıdakı ACID prinsiplər vardır:

***Atomicity (Atomarlıq)*** - verilənlər bazasında aparılan əməliyyatların vahidliyini göstərir. Bazada əməliyyatlar seriyası ya hamısı baş verir yada heç biri baş vermir deməkdir.

***Consistency (Ardıcılıq)*** - tranzaksiyaların (eyni vaxtda yerinə yetirilən əməliyyatların) yarımçıq deyil ardıcıl olaraq yerinə yetirilməsini təmin edir. Bu prinsip həmçinin bazada əməliyyatların səhvsiz və problemsiz olaraq irəliləməsi üçün vacibdir.

***Isolation (Ayrıcalıq)*** - tranzaksiyaların onlar bitənə qədər bir birindən ayrı şəkildə icra olunmasını təmin edir. Məsələn bir tranzaksiya hələ tamamlanmamış digər tranzaksiyadan məlumat oxuya bilməz. Əgər iki tranzaksiya eyni zamanda baş verirsə onlar ayrı ayrılıqda sərbəst şəkildə yerinə yetiriləcəklər. Və əgər biri digərində yazılmış məlumatı oxumalıdırsa o zaman onun bitməsini gözləyəcək.

***Durability (Davamlılıq)*** - davamlılıq o deməkdir ki əgər bir tranzaksiya sona yetibsə onun səbəb olduğu dəyişikliklər heç zaman itməyəcək və yadda saxlanacaq. Hətta sistemdə xətlər və hər hansı problem baş versə belə.

**Bu dərisdə biz əsasən atomarlıq üzərində dayanacağıq. Atomarlığa aid bir real nümunə gətirə bilərik. Məsələn aviabilet sifariş etmək iki hərəkəti tələb edir: ödəmə və yer bron etmək. Potensial müştəri ya eyni zamanda **ödəmə edib yer bron edəcək** ya da **ödəmə etməyib** heç bir **yer məşğul etməyəcək**.**

Atomarlıq prinspinə riayət etmək verilənlər bazasının düzgün **proyektləşdirilməsi** üçün çox önəmlidir. **Proyektləşdirmə** dedikdə cədvəllərin bir biri ilə düzgün əlaqələndirilməsi, cədvəllərdə sütunlar üçün tiplərin düzgün seçilməsi, bu tiplərə uyğun düzgün informasiyaların saxlanması və s. nəzərdə tutulur.

Proyektləşdirməyə atomarlıq prinsipi cəhətdən baxaq.

Tutaq ki bizdə **cellphone** cədvəli var və cədvəldə telefonların adları, hansı ölkələrdən sifariş olunduqları və onlar haqqında informasiyalar əks olunub. Məlumdur ki bir telefon bir deyil bir neçə ölkədən gətirilə bilər.

*Sütunlara daxil ediləcək məlumatlar ən kiçik detallara qədər ayrılmalıdır. Cədvəldə eyni xanada bənzər ikili informasiyalar saxlanıla bilməz, yəni eyni tipli elementlər birgə yazıla bilməz. Belə olmadıqda atomarlıq prinsipi pozula bilər.*

Aşağıdakı izahlı nümunədə bunu açıq aşqar görmək mümkündür.

**CellPhone** ❌

ID	NAME	COUNTRY
1	Samsung Galaxy S7	South Korea, Switzerland
2	iPhone 6	USA , Germany
.	...	...
.	...	...

NOT GOOD IDEA



ATOMARLIQ POZULUR

**CellPhone** ✅

ID	NAME	Description
1	Samsung S7	...
2	iPhone 6	...
3	Nokia 300	...
4	Sony xp Z5	...

**COUNTRY**

ID	CNT_NAME	CellPhone_ID
1	USA	2
2	Germany	2
3	Switzerland	1
4	South KOREA	1

GOOD IDEA!!



ATOMARLIQ POZULMUR

Lakin atomarlıq prinsipi **aspektədən asılı olaraq** dəyişilə bilər. Məsələn tutaq ki Əli proqramçıdır, lakin müəyyən bir proqramlaşdırmaya aidiyyəti olmayan bir şirkətə CV göndərüb orada çalışmaq istəyir. O öz CV-sində **kompyuter bilikləri** sahəsində **MySQL, Java SE/EE, T-SQL, PHP** şəklində proqramlama biliklərini yazarsa bu zaman **atomarlıq pozulmuş hesab olunmaz** çünki müraciət olunan şirkətə proqramçı deyil normal kompyuter bilikləri olan işçi lazımdır. Lakin əgər bu şirkət hər hansı **IT şirkəti** olarsa bu zaman birgə yazılan bu **məlumatlar atomar sayılmaz**. Bu zaman məlumatları bu şəkildə ayırmaq lazımdır:

**Database lang** : MySQL, T-SQL

**Programming lang** : Java SE/EE , PHP

Atomarlıq cədvəldəki verilənlərə nəzarəti artırmaq baxımından çox faydalıdır.

Cədvəlləri bir biri ilə elə əlaqələndirmək lazımdır ki onlar vahid bir obyektə təsvir etmiş olsun. Yuxarıda şəkillə göstərdiyimiz nümunəni SQL kodla ifadə edək.

```
mysql> create table if not exists cellPhone(  
-> id int not null auto_increment primary key,  
-> name varchar(40) not null,  
-> date_of_entry timestamp default current_timestamp);  
Query OK, 0 rows affected (0.22 sec)
```

```
mysql> create table if not exists country(  
-> id int not null auto_increment primary key,  
-> name varchar(50) not null,  
-> cellPhone_id int,  
-> foreign key fk_phone_id(cellPhone_id) references  
-> cellPhone(id) on update cascade on delete cascade);  
Query OK, 0 rows affected (0.28 sec)
```

burada **country** cədvəlində **cellPhone\_id** sütunu **cellPhone** cədvəlindən xarici açar saxlayır. Buna görə foreign key olaraq təyin olunmuşdur:

**foreign key fk\_phone\_id(cellPhone\_id) references cellPhone(id) on update cascade on delete cascade**

Və sonda hər bir cədvəli describe etməklə qurduğumuz sturuktura baxa bilərik:

```
mysql> describe cellPhone;  
+-----+-----+-----+-----+-----+-----+  
| Field      | Type          | Null | Key | Default          | Extra          |  
+-----+-----+-----+-----+-----+-----+  
| id         | int(11)       | NO   | PRI | NULL             | auto_increment |  
| name       | varchar(40)   | NO   |     | NULL             |                |  
| date_of_entry | timestamp     | NO   |     | CURRENT_TIMESTAMP |                |  
+-----+-----+-----+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> describe country;  
+-----+-----+-----+-----+-----+-----+  
| Field      | Type          | Null | Key | Default          | Extra          |  
+-----+-----+-----+-----+-----+-----+  
| id         | int(11)       | NO   | PRI | NULL             | auto_increment |  
| name       | varchar(50)   | NO   |     | NULL             |                |  
| cellPhone_id | int(11)       | YES  | MUL | NULL             |                |  
+-----+-----+-----+-----+-----+-----+  
3 rows in set (0.00 sec)
```

## DDL, DML və DQL anlayışları.



MySQL -də sorğular müəyyən kateqoriyalara ayrılırlar:

- 1. Cədvəlin ümumi sturukturuna təsir edən sorğular qrupu;(DDL)**
- 2. Cədvəldəki məlumatlara manipulyasiya edən sorğular qrupu;(DML)**
- 3.Cədvəlin özünə müəyyən sorğular göndərən sorğular qrupu; (DQL)**

Bunlardan əlavə də sorğu qrupları vardır lakin əsas önəmli olanlar bunlardır.İndi isə ayrı ayrılıqda bu sorğu kateqoriyalarına nəzər salaq.

**DDL - Data Defination Language (Verilənlərin Təyin olunması Dili )**

**create** - database və table yaratmaq üçün,

**alter** - cədvəllərin sturukturunu dəyişmək üçün,

**drop** - cədvəlləri silmək üçün

təyin olunmuş sorğulardır ki DDL qrupuna aid edilirlər.



## **DML - Data Manipulation Language (Verilənlərə Manipulyasiya etmə Dili)**

**insert** - məlumatları cədvələ əlavə etmək üçün,

**update** - məlumatları yeniləmək üçün,

**delete** - məlumatları silmək üçün

nəzərdə tutulan sorğular qrupudur. Əsasən bu sorğular verilənlər bazasına manipulyasiya etmək üçün geniş istifadə olunadırlar.

## **DQL - Data Query Language (Verilənlərə Sorğu vermək Dili)**

**select** - informasiyanın seçilməsi üçün,

**show** - mövcud informasiyaların göstərilməsi üçün,

**help** - MySQL bələdçisindən online axtarış edərək kömək almaq üçün

istifadə olunan DQL komandalar toplusudur.

## **MySQL-də script faylların run olunması. (mysql batch mode)**

SQL-də script fayl **.sql** uzantılı fayllara deyilir. Hansı ki bu fayllar öz daxilində toplu şəkildə sql sorğular saxlayır.

MySQL-də script fayl yaratmaq üçün bir text faylı açıb daxilinə lazımı sorğularımızı yazırıq və sonra həmin faylı

**.sql** uzantısı ilə adlandırırıq.Məsən bir text faylı yaradıb daxilinə aşağıdakı sorğuları kopyalayaq:

```
#my sql commands bundle
```

```
# işarəsi commentdə açıqlamalar yazmaq üçün istifadə olunur
```

```
# author Etibar Vazirov
```

```
drop database if exists smth_error;
```

```
create database if not exists my_new_db;
```

```
use my_new_db;
```

```
create table if not exists person(
```

```
id int not null primary key auto_increment,
```

```
pr_name varchar(22),
```

```
pr_surname varchar(33));
```

```
insert into person(pr_name,pr_surname)  
values('Etibar','Vazirov'),('Fizik','Ehmedov');
```

```
alter table person add column age int not null;
```

```
update person set age =27 where id=1;
```

```
update person set age =20 where id=2;
```

Daha sonra isə bu faylı **library.sql** adıyla yaddaşa verərək kompyuter ekrarında saxlayaq.Bu sql script faylını həm **cmd-dən** həm də **mysql workbench**-dən run etmək mümkündür.

**cmd** əmrlər sətrindən run etmək üçün

**C:\Program Files\MySQL\MySQL Server 5.7\bin**

yoluyla bin qovluğuna daxil olduqdan sonra

**mysql -u root -p <** yazıb daha sonra library.sql faylının yolunu kopyalamaq lazımdır: (məndə library.sql faylının yolu budur : **C:\Users\Admin\Desktop\ library.sql** )

```
C:\Program Files\MySQL\MySQL Server 5.7\bin>mysql -u root -p < C:\Users\Admin\Desktop\library.sql
Enter password: ****
C:\Program Files\MySQL\MySQL Server 5.7\bin>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.7.13-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

library.sql scriptinin yaratdığı dəyişiklikləri görmək üçün **show databases;**

```
mysql> select * from person;
+----+-----+-----+-----+
| id | pr_name | pr_surname | age |
+----+-----+-----+-----+
| 1  | Etibar  | Uzirov     | 27  |
| 2  | Fizik   | Ehmedov    | 20  |
+----+-----+-----+-----+
2 rows in set (0.01 sec)
```

**use my\_new\_db;**

**show tables;**

**select \* from person;**

sorğularını bir bir icra etmək kifayətdir.

Eyni əməliyyatı **mysql workbench** ilə daha sadə yolla etmək olar. Belə ki workbench -də yuxarıda File menyusundan

**'Open SQL Script'** pəncərəsini açırıq. Açılan pəncərədə **library.sql** faylının yerləşdiyi yerə gedib həmin faylı seçirik və open düyməsinə klik edirik. Bundan sonra faylda olan

**bütün sql sorgular(hamısı düz olacağı təqdirdə) run olacaqdır.**

**\*\*\***

**7-ci dərsin sonu**

**Növbəti dərsin mövzusu :**

*Alter table komandası.*

*Cədvəllərarası əlaqələr(joins).*

**Diqqətiniz üçün təşəkkürlər**



Mərhəmətli və Rəhimli Allahın adı ilə



## MySQL Development Training

### 8-ci dər

*MySQL -də ALTER TABLE komandası.*

*Cədvəllərarası əlaqələr(joins).*



*Təlimçi : Etibar Vəzirov*

*Java Developer*

## ALTER table komandası və onunla birlikdə işlənən operatorlar.

**Alter table** vasitəsilə biz cədvəlin adını dəyişə, ona sütun əlavə edə, sütunun adını və ya tipini dəyişə, sütunu silə , foreign key, primary key, unikal indeks əlavə edə və s. bir sıra proseslər yerinə yetirə bilərik.

**Alter table** ilə birlikdə bir sıra açar sözlər işlədilir. Bunlardan **change, modify, add** və **drop** sadalaya bilərik. Bunların hər biri ilə praktiki olaraq tanış olacağıq.

**Alter table** komandası ilə bazada mövcud cədvəlin adını dəyişə bilərik. Məsələn **mysql\_training** bazasında bizdə **test\_student** cədvəli var, onun adını **learner** olaraq dəyişmək istəyirik. Bunun üçün sorğunu bu şəkildə yazmalıyıq:

**alter table test\_student  
rename to learner;**

```
mysql> show tables;
+-----+
| Tables_in_mysql_training |
+-----+
| cellphone                |
| country                  |
| employee                 |
| exam                     |
| student                  |
| test                     |
| test_student             |
+-----+
7 rows in set (0.00 sec)

mysql> alter table test_student rename to learner;
Query OK, 0 rows affected (0.16 sec)

mysql> show tables;
+-----+
| Tables_in_mysql_training |
+-----+
| cellphone                |
| country                  |
| employee                 |
| exam                     |
| learner                  |
| student                  |
| test                     |
+-----+
7 rows in set (0.00 sec)
```

Şəkildən də göründüyü kimi atırıq bazadakı cədvəllər sırasında **test\_student** deyil **learner** cədvəli vardır.

**Change** ilə cədvəldə sütunların həm **adlarını** həm də **tiplərini** dəyişə bilərik.

Sütunların yerlərini dəyişmək üçün **modify** komandası istifadə edilir. **Modify** ilə həmçinin **change** kimi **sütunların tiplərini** də dəyişmək olur.

**Add** komandası ilə cədvələ **yeni sütun əlavə etmək** mümkündür.

**Drop** əmri ilə də cədvəldən **istədiyimiz sütunu silə bilərik**.

Change komandası ilə bazadakı exam cədvəlinin **exam\_name** sütun adını **name\_of\_exam** olaraq dəyişək. Bu sütunun həmçinin tipində də dəyişiklik edə bilərik.

**alter table exam**

**change column exam\_name name\_of\_exam**

**varchar(15) not null;**

```
mysql> describe exam;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| exam_id    | int(11)       | NO   | PRI | NULL    | auto_increment |
| exam_name  | varchar(20)   | YES  |     | NULL    |                |
| STD_ID    | int(11)       | NO   | MUL | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.03 sec)

mysql> alter table exam
-> change column exam_name name_of_exam varchar(15) not null;
Query OK, 3 rows affected (0.60 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> describe exam;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| exam_id    | int(11)       | NO   | PRI | NULL    | auto_increment |
| name_of_exam | varchar(15)   | NO   |     | NULL    |                |
| STD_ID    | int(11)       | NO   | MUL | NULL    |                |
+-----+-----+-----+-----+-----+-----+
```

Əgər bir yox **bir neçə sütunun adını dəyişmək istəsək** bu zaman **change column** yazıb ad dəyişikliyi etdikdən sonra **ardınca vergül qoyub yenidən change column** yazaraq digər sütunlar üçün də ad və tip dəyişikliyi edə bilərik.

Bundan əlavə əgər sütunun adı deyil **sadəcə tipini dəyişmək** lazımdırsa o zaman **change column yazıb sütunun adını iki dəfə yazaraq** ("köhnə ad " " yeni ad " yerinə) sonra tipi dəyişə bilərik. Eyni zamanda əgər sütun adını dəyişib tipi dəyişmək istəmiriksə bu zaman əvvəl təyin olunan tipi təkrar olaraq yenə yazmalıyıq:

**alter table exam**

**change column name\_of\_exam examName varchar(15);**

Modify komandası ilə sütun tipini dəyişdikdə isə sütun adını iki dəfə təkrarlamağa ehtiyac olmur:

**alter table exam**

**modify column name\_of\_exam varchar(15);**

Əvvəldə qeyd etdiyimiz kimi modify komandası ilə həmçinin sütunların yerlərini dəyişmək mümkündür:

```
mysql> describe exam;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| exam_id    | int(11)       | NO   | PRI | NULL     | auto_increment |
| name_of_exam | varchar(15)   | NO   |     | NULL     |                |
| STD_ID     | int(11)       | NO   | MUL | NULL     |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)

mysql> alter table exam modify column name_of_exam varchar(15) after STD_ID;
Query OK, 0 rows affected (0.56 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe exam;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| exam_id    | int(11)       | NO   | PRI | NULL     | auto_increment |
| STD_ID     | int(11)       | NO   | MUL | NULL     |                |
| name_of_exam | varchar(15)   | YES  |     | NULL     |                |
+-----+-----+-----+-----+-----+-----+
```



`alter table exam modify column name_of_exam varchar(15) after STD_ID;`

sorğusu ilə exam cədvəlində name\_of\_exam və STD\_ID sütunlarının yerini dəyişdik. Bundan əlavə **after** kimi FIRST, SECOND, LAST və s. açar sözlərindən istifadə etməklə sütunların yerini asanlıqla dəyişə bilərik.

İndi isə alter table komandası ilə cədvələ yeni bir sütun əlavə olunmasına baxaq.

```
mysql> describe learner;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)       | NO   | PRI | NULL    | auto_increment |
| name  | varchar(30)   | YES  |     | unknown |                |
| surname | varchar(40)  | YES  |     | unknown |                |
| age   | int(11)       | YES  |     | 0       |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> alter table learner add column email varchar(33) default null;
Query OK, 0 rows affected (0.43 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe learner;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)       | NO   | PRI | NULL    | auto_increment |
| name  | varchar(30)   | YES  |     | unknown |                |
| surname | varchar(40)  | YES  |     | unknown |                |
| age   | int(11)       | YES  |     | 0       |                |
| email | varchar(33)   | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

Şəkildən görüldüyü kimi learner cədvəlinə

`alter table learner add column email varchar(33) default null;`

sorğusu ilə **email** sütunu əlavə etdik. Həmçinin qeyd edək ki yuxarıda sadalanan açar sözlər (first, last, after və s.) vasitəsilə yeni sütunu cədvəldə istədiyimiz yerə əlavə edə bilərik.

**Drop** komandasıyla cədvəldən istədiyimiz sütunu silə bilərik:

```
mysql> alter table learner drop column email;
Query OK, 0 rows affected (0.45 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe learner;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(30)	YES		unknown	
surname	varchar(40)	YES		unknown	
age	int(11)	YES		0	

```
4 rows in set (0.01 sec)
```

`alter table learner drop column email;`

sorgusuyla **learner** cədvəlindən **email** sütununu sildik.

Əvvəlcədən mövcud olan cədvələ alter table konstruksiyası vasitəsilə primary key, unique key və s. əlavə etmək üçün yeni bir sadə **goods** cədvəl yaradaq.

`create table goods(`

`id int not null,`

`gd_name varchar(20),`

`price double(5,2) not null); // (double ədəd 000.00-999.99 )`

```
mysql> create table goods(
-> id int not null,
-> gd_name varchar(20),
-> price double(5,2) not null);
Query OK, 0 rows affected (0.20 sec)
```

Demək bu cədvəldə biz id sütununa primary key və auto\_increment əlavə etməliyik.

`alter table goods change column id id int not null`

`auto_increment, add primary key(id);`

```
mysql> alter table goods change column id id int not null
-> auto_increment, add primary key(id);
Query OK, 0 rows affected (0.56 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe goods;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
gd_name	varchar(20)	YES		NULL	
price	double(3,2)	NO		NULL	

Dəyişikliyi görmək üçün **describe goods** yazmaq kifayətdir.

Başqa bir dəyişiklik də etmək olar, məlumdur ki goods(məhsullar) cədvəlində məhsulların adları təkrarlanan olmamalıdır, buna görə gd\_name sütununa unikal indeksləmə (unique index) təyin edə bilərik.

**alter table goods add unique index(gd\_name);**

```
mysql> alter table goods add unique index(gd_name);
Query OK, 0 rows affected (0.25 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe goods;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)| NO   | PRI | NULL    | auto_increment |
| gd_name | varchar(20)| YES | UNI | NULL    | |
| price | double(3,2)| NO  |    | NULL    | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

**foreign key** əlavə etmək üçün goods cədvəli başqa bir cədvəllə əlaqəli olmalıdır. Məsələn bu məhsulların sifariş olunduqları ölkələr(**country**) cədvəlini quraq.

**create table country(**

**id int not null primary key auto\_increment,**

**name varchar(40) not null);**

```
mysql> create table country(
  -> id int not null primary key auto_increment,
  -> name varchar(40) not null);
Query OK, 0 rows affected (0.33 sec)

mysql> describe country;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)| NO   | PRI | NULL    | auto_increment |
| name  | varchar(40)| NO  |    | NULL    | |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

Bu country cədvəlinə **goods** -dan id saxlayan xarici açar (foreign key) əlavə edək.

```
alter table country add column goods_id int not null ,  
add constraint foreign key(goods_id) references goods(id);
```

```
mysql> alter table country add column goods_id int not null,  
-> add constraint foreign key(goods_id) references goods(id);  
Query OK, 0 rows affected (0.68 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
mysql> describe country;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(40)	NO		NULL	
goods_id	int(11)	NO	MUL	NULL	

Bu sorğuyla iki əməliyyat yerinə yetirmiş olduq: həm country cədvəlinə **goods\_id** sütununu əlavə etdik həm də bu sütuna xarici açar(**foreign key**) məhdudiyyəti təyin etdik.

## Cədvəllərarası əlaqələr(**JOIN**).

Verilənlər bazasında ən önəmli aspektlərdən biri də cədvəllərarası əlaqənin təmin ediləsidir. Cədvəllər arasında əlaqə join açar sözü ilə qurulur. Əsasən aşağıdakı əlaqə növləri mövcuddur: **inner join**, **left join**, **right join**, **full outer join**.

Tutaq ki bizdə bazada **developer** və **PL** (programming language) kimi iki cədvəlimiz var və **PL** cədvəli **developer** cədvəlindən **foreign key** saxlayır.

Öncə bu cədvəlləri yaratmaq üçün sorğular yazaq və sonra onlara test üçün informasiyalar daxil edək. Bundan sonra onların **join** olunması məsələsinə baxa bilərik.

```
mysql> create table if not exists developer(  
-> id int not null auto_increment primary key,  
-> name varchar(25) not null);  
Query OK, 0 rows affected (1.54 sec)  
  
mysql> create table if not exists PL(  
-> id int not null auto_increment primary key,  
-> pl_name varchar(30) not null,  
-> dev_id int,  
-> foreign key fk_dev_id(dev_id) references developer(id));  
Query OK, 0 rows affected (0.25 sec)
```

```
create table if not exists developer(  
id int not null auto_increment primary key,  
name varchar(25) not null);
```

---

```
create table if not exists PL(  
id int not null auto_increment primary key,  
pl_name varchar(30) not null,  
dev_id int,  
foreign key fk_dev_id(dev_id) references developer(id));
```

Bu cədvəllərə müəyyən informasiyalar daxil edək.

```
insert into developer(name) values('Vuqar'),('Sahil'),  
('Elekber'),('Leman'),('Nino'),('Aysel');  
insert into PL(pl_name,dev_id) values('Delphi',2),('C++',3),  
('C#',3),('Java',1),('Python',4),('PHP',1);
```

Yuxarıdakı kimi insert əməliyyatlarından sonra belə bir (sağda şəkildəki kimi) informasiyaya malik cədvəllərimiz olacaqdır.

**select \* from developer;**

**select \* from PL;**

İndi isə bu cədvəllər arasında **join** əməliyyatı aparılmasına

baxaq. **Developer** və **PL** cədvəllərini ümumi bir açara görə birləşdirə bilərik ki bu açar **developer** cədvəlində **id**, **PL** cədvəlində isə **dev\_id** -dir.

**select developer.name, PL.pl\_name from developer**

**inner join PL on developer.id = PL.dev\_id;**

```
mysql> select developer.name, PL.pl_name from developer
-> inner join PL on developer.id = PL.dev_id;
+-----+-----+
| name   | pl_name |
+-----+-----+
| Uuqar  | Java    |
| Uuqar  | PHP     |
| Sahil  | Delphi  |
| Elekber| C++     |
| Elekber| C#      |
| Lemana | Python  |
+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> select * from developer;
+----+-----+
| id | name  |
+----+-----+
| 1  | Uuqar |
| 2  | Sahil |
| 3  | Elekber |
| 4  | Lemana |
| 5  | Nino  |
| 6  | Aysel |
+----+-----+
6 rows in set (0.00 sec)

mysql> select * from pl;
+----+-----+-----+
| id | pl_name | dev_id |
+----+-----+-----+
| 1  | Delphi  | 2      |
| 2  | C++     | 3      |
| 3  | C#      | 3      |
| 4  | Java    | 1      |
| 5  | Python  | 4      |
| 6  | PHP     | 1      |
+----+-----+-----+
6 rows in set (0.00 sec)
```

Belə əlaqəliliyə **inner join** (daxili birləşmə) deyilir.

Əgər yuxarıdakı sorğuda **inner** sözü əvəzinə **left** yazsaq, bu **left join** (soldan birləşmə) olacaq.

```
select
developer.name,
PL.pl_name from
developer
```

```
left join PL on
developer.id = PL.dev_id;
```

```
mysql> select developer.name, PL.pl_name from developer
-> left join PL on developer.id = PL.dev_id;
```

name	pl_name
Uuqar	Java
Uuqar	PHP
Sahil	Delphi
Elekber	C++
Elekber	C#
Leman	Python
Nino	NULL
Aysel	NULL

```
8 rows in set (0.00 sec)
```

Sorğuda left join sözündən solda yazılan **developer** cədvəli **PL** cədvəlinə **soldan birləşir**, buna görə də sorğunun nəticəsi olaraq soldakı cədvəldə olan **bütün** məlumatlar, sağdakı cədvəldə isə **yalnız kəsişmədə olan** məlumatlar görünəcəkdir. Şəkildən də göründüyü kimi **PL** cədvəlində **Aysel** və **Nino** adlı developerlərə uyğun **id**-lər olmadığına görə onlar üçün null dəyərləri çıxmışdır.

Analoji qaydada deyə bilərik ki sonuncu sorğuda **left** sözü əvəzinə **right** yazsaq belə birləşmə **right join** (sağdan birləşmə) olacaq. Buna görə də sorğunun nəticəsi olaraq soldakı cədvəldə **yalnız kəsişmədə olan** məlumatlar, sağdakı cədvəldə isə **bütün** məlumatlar görünməlidir.

```
select developer.name, PL.pl_name from developer
```

```
right join PL on developer.id = PL.dev_id;
```

Lakin bizim sağ cədvəldə - yəni **PL** cədvəlində **dev\_id** **foreign key** olduğuna görə o heç cür **developer** cədvəlinin **id**-sindən başqa bir dəyər götürə bilmir ona görə insert zamanı ona **developer**-in **id**-sindən savayı (**1,2,3,4,5,6 ədədlərindən**

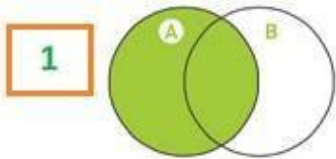
**başqa)** digər tam qiymət insert edə bilməzdik. Bu halda bu cədvəllərin **right join** olması **inner join** kimi görünəcəkdir.

```
mysql> select developer.name, PL.pl_name from developer
-> right join PL on developer.id = PL.dev_id;
+-----+-----+
| name   | pl_name |
+-----+-----+
| Sahil  | Delphi  |
| Elekber| C++     |
| Elekber| C#      |
| Uuqar  | Java    |
| Lenan  | Python  |
| Uuqar  | PHP     |
+-----+-----+
6 rows in set (<0.00 sec)
```

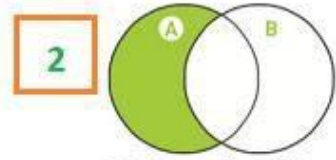
SQL-də cədvəllərin join olunmasını Venn diaqramları vasitəsilə belə göstərmək olar:

## SQL JOINS

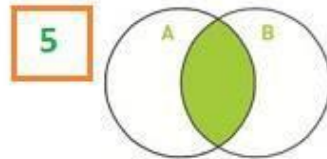
MySQL Development Training



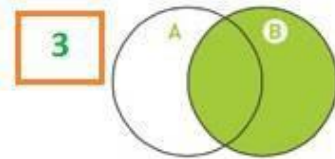
```
1
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



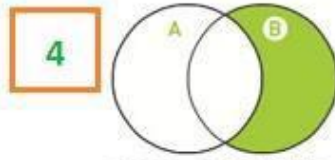
```
2
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



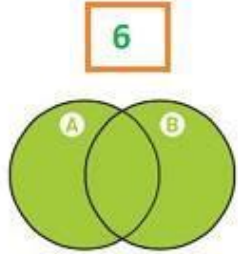
```
5
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



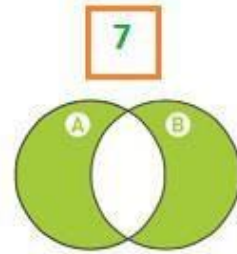
```
3
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
4
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
6
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
7
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```



**1. LEFT JOIN (kəsişmə null deyil)**

select <select\_list> from table A LEFT JOIN table B  
on A.key = B.key;

**2. LEFT JOIN (kəsişmə NULL-dur)**

select <select\_list> from table A LEFT JOIN table B  
on A.key = B.key where B.key is NULL;

**3. RIGHT JOIN (kəsişmə null deyil)**

select <select\_list> from table A RIGHT JOIN table B  
on A.key = B.key;

**4. RIGHT JOIN (kəsişmə NULL-dur)**

select <select\_list> from table A RIGHT JOIN table B  
on A.key = B.key where A.key is NULL;

**5. INNER JOIN**

select <select\_list> from table A INNER JOIN table B  
on A.key = B.key;

**6. FULL OUTER JOIN (kəsişmə null deyil)**

select <select\_list> from table A FULL OUTER JOIN  
table B on A.key = B.key;

**7. FULL OUTER JOIN (kəsişmə NULL-dur)**

select <select\_list> from table A FULL OUTER JOIN  
table B on A.key = B.key where A.key is NULL or B.key is  
NULL

\*\*\*

**8-ci darsin sonu**

**Növbəti darsin mövzusu :**

*MySQL-də tranzaksiyalar və triggerlər*

**Diqqətiniz üçün təşəkkürlər**



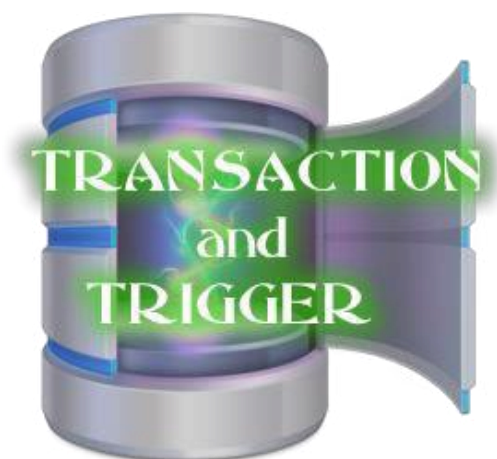
Mərhəmətli və Rəhimli Allahın adı ilə



## MySQL Development Training

### 9-cu dər

### *MySQL-də tranzaksiyalar və triggerlər*

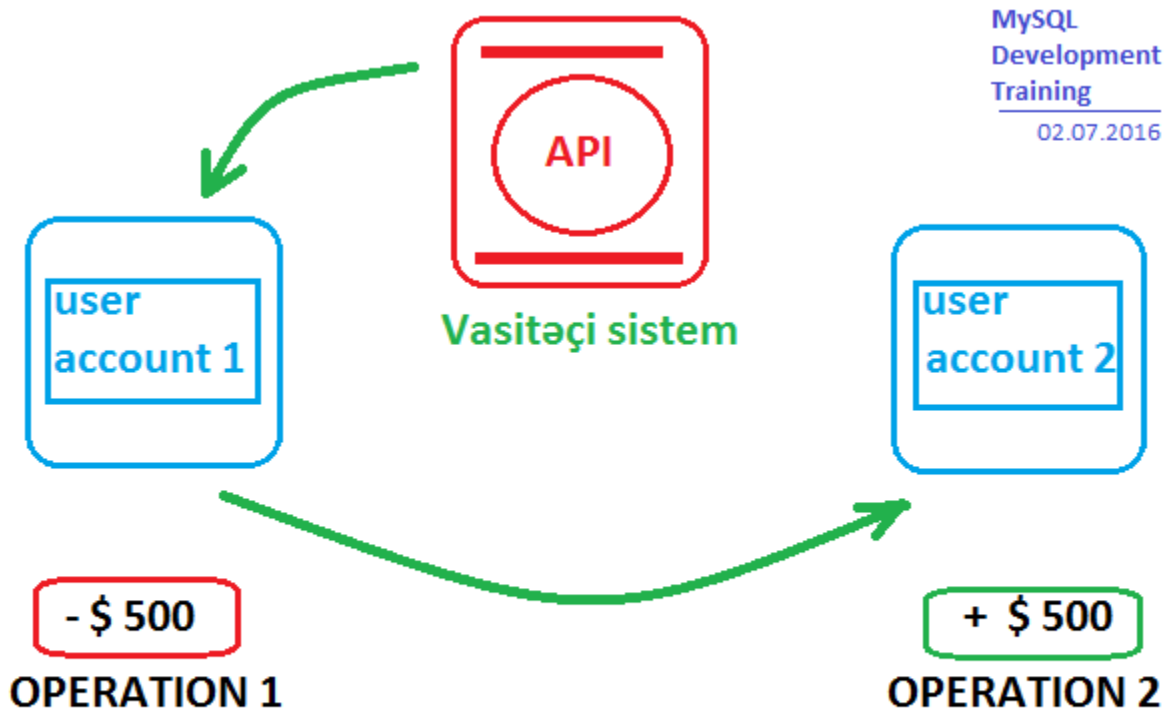


Təlimçi : *Etibar Vəzirov*

*Java Developer*

## MySQL -də Tranzaksiyalar.

Öncəki dərslərdə də qeyd etdiyimiz kimi **tranzaksiya** bir neçə əməliyyatın vahid bir əməliyyat kimi qəbul olunmasıdır. Xüsusilə ödəmə sistemləri ilə bağlı layihələrdə **tranzaksiyalar** geniş istifadə olunur. Bununla əlaqədar bir nümunəyə baxaq.



### API - Application Programming Interface

Tutaq ki bizdə vasitəçi bir ödəmə sistemi var və bu sistem 1-ci istifadəçinin balansından \$500 çıxarıb ikinci istifadəçinin balansına əlavə etməlidir. Bunun üçün iki əməliyyat icra olunmalıdır. Bu iki əməliyyatın vahid bir əməliyyat kimi tam olaraq icra olunması üçün biz **tranzaksiya** başlatmalıyıq.

Burada ödəmə sistemi **API** -nin adı keçir.**API**- lar hər hansı mürəkkəb sistemlə işi asanlaşdırmaq üçün həmin sistemin müvafiq funksiyalar və metodlar kitabxanasıdır və "**Application Programming Interface**" birləşməsinin qısaltmasıdır.

*Əgər hər hansı problem səbəbilə tranzaksiya zamanı 1-ci istifadəçinin balansından çıxarılan məbləğ 2-ci istifadəçinin balansına yazılmazsa bu zaman əməliyyat uğursuz hesab olunur,yəni yarımçıq yerinə yetirilmiş olur.Belə olduqda mütləqdir ki yarımçıq qalmış əməliyyat geri qaytarılsın.*

Tranzaksiyanı başlatmaq üçün **start transaction** əmrindən istifadə edilir və **iki sorğu ; ilə ayrılmaqlarda** arda yazılır.Sonra əməliyyat uğurlu gedərsə onu təsdiqləmək üçün **COMMIT** yazılır.Əgər əməliyyatda problem yaranarsa əvvəlki vəziyyətə qaytarılmaq üçün sonda **ROLLBACK** yazmaq lazımdır.

Bu dediklərimizi real bir nümunədə göstərmək üçün yeni bir **transactions** database yaradıb orada **master\_card** və **web\_money** adlarında iki cədvəl yaradaq.Bu cədvəllərdəki bir userin balansından

**\$500** məbləğ çıxarıb digər userin balansına əlavə edək.Sorğularımız aşağıdakı kimi olacaq:

```
mysql> create database if not exists transactions;
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> use transactions;
Database changed

mysql> create table if not exists master_Card(
  -> id int not null auto_increment primary key,
  -> name varchar(20) not null,
  -> account1 double(8,2));
Query OK, 0 rows affected (0.27 sec)

mysql> create table if not exists web_Money(
  -> id int not null auto_increment primary key,
  -> name varchar(20) not null,
  -> account2 double(8,2));
Query OK, 0 rows affected (0.21 sec)

mysql> show tables;
+-----+
| Tables_in_transactions |
+-----+
| master_card             |
| web_money               |
+-----+
```

create database if not exists **transactions**;

---

create table if not exists **master\_card**(  
id int not null primary key auto\_increment(  
name varchar(20) not null,  
**account1** double(8,2));

---

create table if not exists **web\_money**(  
id int not null primary key auto\_increment(  
name varchar(20) not null,  
**account2** double(8,2));

Sonrabu cədvəllərə müvafiq informasiyalar daxil edək:

```
mysql> insert into master_card(name,account1) values('Nicat',1700.77);  
Query OK, 1 row affected (0.07 sec)  
  
mysql> select * from master_card;  
+----+-----+-----+  
| id | name  | account1 |  
+----+-----+-----+  
| 1  | Nicat | 1700.77  |  
+----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql> insert into web_money(name,account2) values('Fizuli',800.24);  
Query OK, 1 row affected (0.07 sec)  
  
mysql> select * from web_money;  
+----+-----+-----+  
| id | name  | account2 |  
+----+-----+-----+  
| 1  | Fizuli | 800.24   |  
+----+-----+-----+  
1 row in set (0.00 sec)
```

insert into **master\_card**(name,account1) values('Nicat',1700.77);

insert into **web\_money**(name,account2) values('Fizuli',800.24);

İndi isə tranzaksiyamızı başlada bilərik:

```
mysql> start transaction; update master_card set account1= account1 - 500
Query OK, 0 rows affected (0.00 sec)
-> where id=1; update web_money set account2= account2 + 500 where id=1; COMMIT;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
Query OK, 0 rows affected (0.07 sec)
mysql> select * from master_card;
+----+-----+-----+
| id | name  | account1 |
+----+-----+-----+
| 1  | Nicat | 1200.77  |
+----+-----+-----+
1 row in set (0.00 sec)
mysql> select * from web_money;
+----+-----+-----+
| id | name  | account2 |
+----+-----+-----+
| 1  | Fizuli | 1300.24  |
+----+-----+-----+
1 row in set (0.00 sec)
```

start transaction; update **master\_card** set account1= **account1-500**  
where id=1;update **web\_money** set account2= **account2 +500** where  
id=1;**COMMIT**;

Bu sorğunu icra etdikdən sonra

**select \* from master\_card;** və **select \* from**  
**web\_money;**yazmaqla müvafiq dəyişikliyi görə bilərik.

Əgər hər hansı bir yalnızlıq ucbatından bazada dəyişiklik  
olmasını istəmiriksə bu zaman sonda **ROLLBACK** yazırıq:

start transaction; update **master\_card** set account1= **account1-500**  
where id=1;update **web\_money** set account2= **account2 +500** where  
id=1;**ROLLBACK**;

```
mysql> start transaction; update master_card set account1= account1 - 500
Query OK, 0 rows affected (0.00 sec)
-> where id=1; update web_money set account2= account2 + 500 where id=1; ROLLBACK;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Query OK, 0 rows affected (0.05 sec)

mysql> select * from web_money;
+----+-----+-----+
| id | name  | account2 |
+----+-----+-----+
| 1  | Fizuli | 1300.24  |
+----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from master_card;
+----+-----+-----+
| id | name  | account1 |
+----+-----+-----+
| 1  | Nicat | 1200.77  |
+----+-----+-----+
```

## Triggerlər

SQL-də **trigger** cədvəllə əlaqədar database obyektinə verilən addır və cədvəldə müəyyən əməliyyatlar baş verərkən **avtomatik** işə

düşür. **Triggerlər** bir çox məqsədlər üçün istifadə oluna bilər, məsələn cədvələ daxil edilən müəyyən məlumatları göstərmək yaxud

cədvəllərdə olan dəyişikliyi, yeniləmələri izləmək və s.

Verilənlər bazası ilə iş üçün **Stored Procedures** adlı prosedurlar vardır ki onlar müəyyən standart funksiyalardır. Bu funksiyaları biz özümüz işə sala bilərik. Necə ki obyekt yönümlü proqramlaşdırma dillərində funksiyalar yaradılır eynilə MySQL-də də elə funksiyalar yaratmaq olur ki onlar **Stored Procedures** adlanır. Lakin





**trigger**lərin **Stored Procedure** -lardan fərqi ondan ibarətdir ki hər hansı bir cədvələ müəyyən bir **dinləyici (listener)** qurlaşdırılır. Bu **dinləyici** cədvəldə müəyyən proseslər gedən zaman həmin **prosesə qoşulur** və **avtomatik olaraq işə düşür**lər. Əsasən cədvəllərdə gedən **3 əməliyyat** zamanı (**insert, update, delete**) **trigger**lərdən istifadə olunur.

**Trigger**lərin **AFTER** və **BEFORE** adlı xüsusi konstruksiyaları vardır. Hansı ki onların köməyi ilə hər hansı cədvəldə müəyyən bir prosesin yerinə yetirilməsindən **əvvəl** və ya **sonra** **trigger**in işə düşməsinə təmin etmək olar.

**Trigger** yaratmaq üçün **create trigger**, silmək üçün isə **drop trigger** komandaları istifadə olunur.

Birlikdə bir **trigger** yaradılması nümunəsinə baxaq. Keçən dərslərimizdə yaratdığımız **developer** cədvəlində cədvələ gedən **insert** prosesi üçün **trigger** yaradaq.

Qeyd edək ki **trigger** yaratmazdan əvvəl onun, dinləməni yazacağı cədvəli yaratmalıyıq. Bu cədvəl **developer** cədvəlindən dinləmə yazacağına görə onu uyğun olaraq **log\_dev** adlandırma bilərik.

```
mysql> select * from developer;
+----+-----+
| id | name  |
+----+-----+
| 1  | Uğur |
| 2  | Sahil |
| 3  | Elekber |
| 4  | Leman |
| 5  | Nino  |
| 6  | Aysel |
+----+-----+
6 rows in set (0.05 sec)

mysql> create table if not exists log_dev(
  -> id int not null primary key auto_increment,
  -> description varchar(50),
  -> dev_id int);
Query OK, 0 rows affected (0.25 sec)
```

```
create table if not exists log_dev(  
id int not null primary key auto_increment,  
description varchar(50),  
dev_id int);
```

Burada **description** sütünü **developer** cədvəlindəki **name** sütununa yazılacaq məlumatı , **dev\_id** sütunu isə **developer** cədvəlində **id** sütununda generasiya olunacaq **id**-ni əks etdirəcəkdir.

İndi isə triggerimizi yarada bilərik:

```
create trigger my_trg AFTER insert on developer FOR  
EACH ROW insert into log_dev(description,dev_id)  
values(NEW.name,NEW.id);
```

```
mysql> create trigger my_trg AFTER insert on developer FOR EACH ROW  
-> insert into log_dev(description,dev_id) values(NEW.name,NEW.id);  
Query OK, 0 rows affected (0.11 sec)
```

triggerin qoşulmasını yoxlamaq üçün developer cədvəlimizə məlumat daxil (insert) edək:

```
mysql> insert into developer(name) values('Etibar');  
Query OK, 1 row affected (0.05 sec)
```

```
insert into developer(name)  
values('Etibar');
```

sorğusu ilə cədvələ insert etdikdən sonra

```
mysql> select * from developer;  
+----+-----+  
| id | name  |  
+----+-----+  
| 1  | Uqar  |  
| 2  | Sahil |  
| 3  | Elekber |  
| 4  | Leman |  
| 5  | Nino  |  
| 6  | Aysel |  
| 7  | Etibar |  
+----+-----+  
7 rows in set (0.00 sec)
```

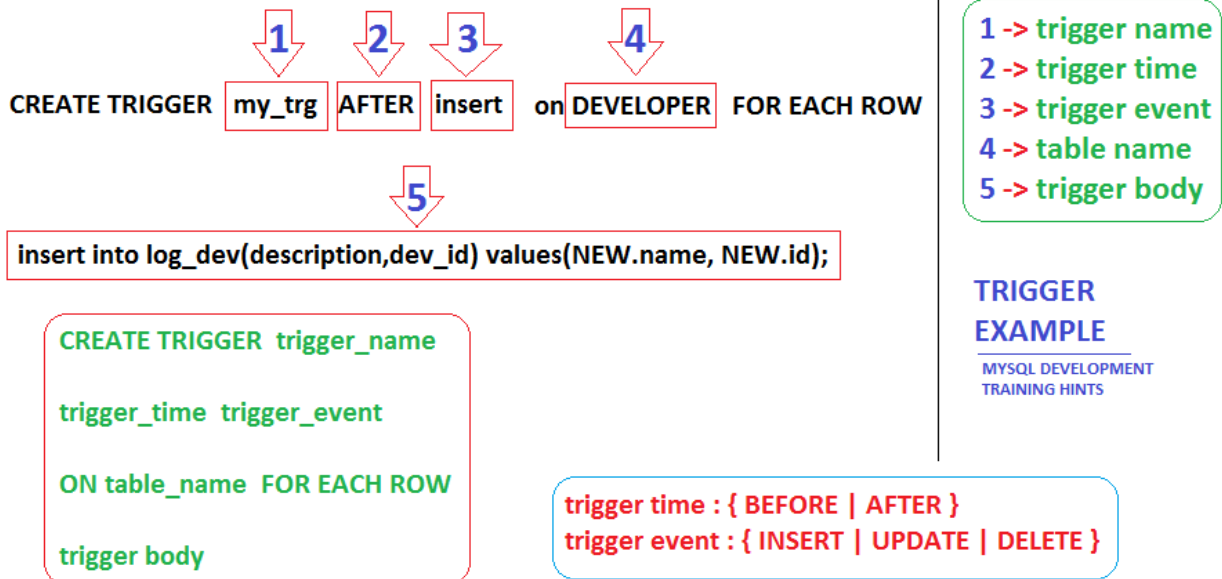
**select \* from developer;**yazaraq 7ci sətrin əlavə olunduğunu görürük.

Triggerimizin **log\_dev**cədvəlinə nə yazdığına baxaq:

**select \* from log\_dev;**komandası ilə triggerimizin insert prosesini dinlədiyini görürük:

```
mysql> select * from log_dev;
+----+-----+-----+
| id | description | dev_id |
+----+-----+-----+
| 1  | Etibar     | 7      |
+----+-----+-----+
1 row in set (0.00 sec)
```

Yuxarıda yazdığımız **trigger** komandasının açıklamasını daha dərindən qavramaq üçün aşağıdakı şəkilə nəzər yetirək:



**trigger\_body** ( yəni **5** hissəsində) hissəsində yazılan **NEW** açar sözü ilə developer cədvəlinin sütunlarına müraciət edirik.Yəni burada **NEW** sözü developer cədvəlindən link saxlayır.Aşağı sol hissədə trigger yaradılmasının sintaksisi əks olunmuşdur.

Yuxarıdakı trigger yaratmasorğusunda **trigger time**-ı və **trigger event** -i dəyişərək digər komandaları da özünüzyoxlayabilərsiniz.

Yaratdığımız triggeri **sil**mək üçün yazacağımız komanda **drop trigger my\_trg;**şəklində olacaq.

\*\*\*

**9-cu dərsin sonu**

**Növbəti dərsin mövzusu :**

***MySQL-in PHP ilə  
əlaqələndirilməsi. SQL injection***

**Diqqətiniz üçün təşəkkürlər**



Mərhəmətli və Rəhimli Allahın adı ilə



## MySQL Development Training

### 10-cu dərş (FINAL)

*MySQL-in PHP ilə əlaqələndirilməsi.SQL injection*



*Təlimçi : Etibar Vəzirov*  
*Java Developer*

## PHP və Wampserver

**PHP** server tərəfdə işləyən bir **script** (hərfi mənada ssenari , təlimatlar seriyası,plan deməkdir) dilidir.**Scriptlər** sadə bir mətn editoru (məsələn notepad) ilə yazıla bilən mətn fayllarıdır. Hansı ki bu mətn fayllarında müəyyən script dilində təlimatlar yazılır.Script dilləri **compile** (byte koda yəni maşın dilinə çevirmə) **mərhələsi tələb etməyən** proqramlama dilləridir.Məsələn normalda **Javascript** -lə yazılmış proqramı run etməzdən əvvəl compile etməyə ehtiyac yoxdur.Lakin **C** dilində yazılan proqram run olunmazdan əvvəl compile olunmalıdır.

**PHP** server tərəfdə **Javascript** isə **client** tərəfdə işləyən **script** dilləridir.Burada server tərəf dedikdə **web server**, client tərəf dedikdə isə **istifadəçi brauzeri** nəzərdə tutulur.



PHP HTML ilə birlikdə istifadə olunmaq üçün hazırlanmışdır və kodları HTML daxilində yazıla bilir.PHP fayllar serverdə **.php** uzantısı şəklində saxlanır.

Brauzer serverdəki **.php** uzantılı fayla sorğu göndərdikdə web server bu faylı **php çeviriciyə** (interpreter) göndərir.Php çevirici faylda olan **php kodlarını html kodlara** çevirir və server

üzərindən brauzerə göndərir. Brauzer ona çatan fayl daxilindəki html kodlarını görür lakin php kodlarını görmür.



PHP dilinin sintaksisi Java və Perl dillərinin sintaksisinə çox oxşardır.

Kompyuterimizdə php proqramlar yazmaq üçün bizə aşağıdakılar lazımdır:

1. Web Server
2. PHP
3. Verilənlər bazası
4. Mətn editoru
5. Brauzer (web səyyah)

Mətn editoru olaraq windows -da **notepad**, Mac -da isə **TextMate** istifadə edə bilərik. Lakin rahatlıq və

funksionallıq baxımından **Notepad ++** proqramını yükləyib onu istifadə etmək daha məsləhətlidir.

(**Notepad ++** yükləmə linki burada

<https://notepad-plus-plus.org/download/v6.9.2.html>)

Brauzer olaraq chrome və ya firefox istifadə etmək çox əlverişlidir. Web server olaraq isə **Apache http server** istifadə edəcəyik.

Və nəhayət verilənlər bazası olaraq MySQL istifadə edəcəyik.

Bütün bu proqramları bir bir yükləmək əvəzinə onların hamısını özündə cəmləşdirən proqramlardan istifadə etmək daha əlverişlidir.

**Wampserver** dediyimiz bu tip proqramlardandır.

(daxilində **Apache, PHP, MySQL, phpMyAdmin** vardır)

Eynilə Linux ƏS üçün **LAMP**, Mac ƏS üçün də **MAMP** proqramlarını istifadə etmək mümkündür.

**Wampserver** -i yükləmək üçün aşağıdakı linkə daxil olub **download** bölməsindən əməliyyat sisteminizə uyğun wampserver proqramını yükəyin :

<http://www.wampserver.com/en/>

**Wampserver** -i yüklədikdən sonra bəzi konfiqurasiya əməliyyatlarını yerinə yetirmək lazımdır. Üzərinə iki dəfə klik etdikdən sonra wampserver ikonu ekranda saat simgəsinin yanında görünəcəkdir. Wampserver-də iki servis olduğu üçün onlardan hər ikisi start olarsa





proqram ikonu yaşıl rəngdə,

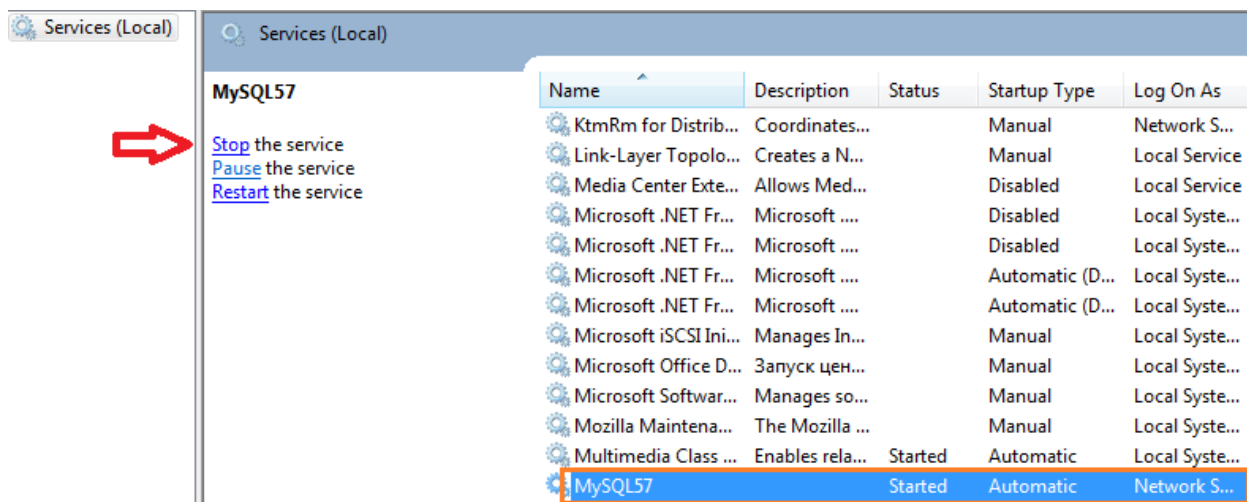
ikisindən biri (MySQL və ya Apache) start olarsa,



proqram ikonu narıncı rəngdə,

heç biri start olunmazsa isə sadəcə qırmızı rəngdə görünəcəkdir. Biz əməliyyat zamanı hər iki servisin işləməsi vacib olduğuna görə ikonun yaşıl rəngdə görünməsi artıq mühitin hazır olduğuna işarədir.

Əgər servislərdən hər hansı biri məsələn MySQL start olursa bu sistemdə başqa MySQL serverin start vəziyyətində olduğunu göstərir. Problemi həll etmək üçün sadəcə start menyusundan **services (windows 8 -də services.msc şəklində axtarıq)** yazıb servislərə daxil olaraq avtomatik işə düşmüş MySQL servisi söndürmək lazımdır:



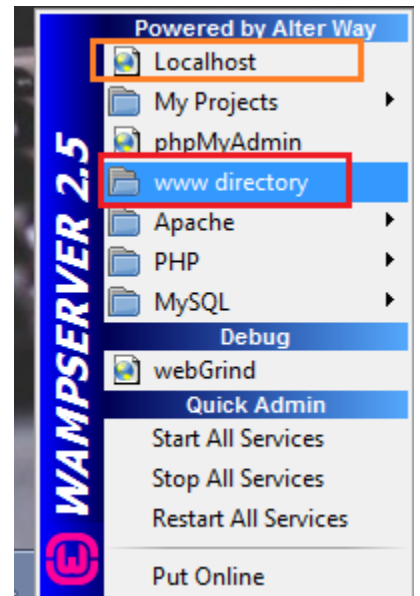
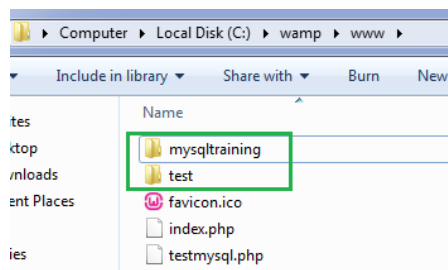
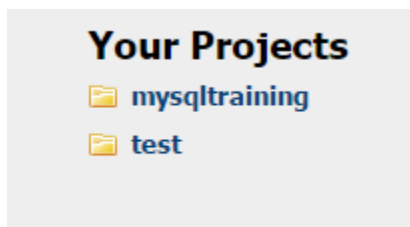
Bundan sonra wampserver simgəsinə klik edib bütün **servisləri stop və yenidən start** etməklə işi tamamlayırıq.

Apache açılan menyusundakı **httpd.conf** faylı Apache serveri üçün konfiqurasiya faylıdır. Eləcə də PHP açılan menyusunda **php.ini** və MySQL menyusunda **my.ini** faylları uyğun proqramlar üçün **konfiqurasiya fayllarıdır**.



**phpMyAdmin** proqramı PHP ilə yazılmış MySQL verilənlər bazasındakı əməliyyatları vizual olaraq yerinə yetirməyə imkan verən proqramdır. Menyudakı phpMyAdmin yazısına klik etdikdə brauzerdə phpMyAdmin açılacaq və bazalarımızı göstərəcəkdir.

Wampserver menyusundakı **localhost** yazısına klik etdikdə brauzerdə wampserver ana sahifəsi açılacaq və bu sahifədə **www** root qovluğu daxilindəki proyektlər görüntülənəcəkdir.

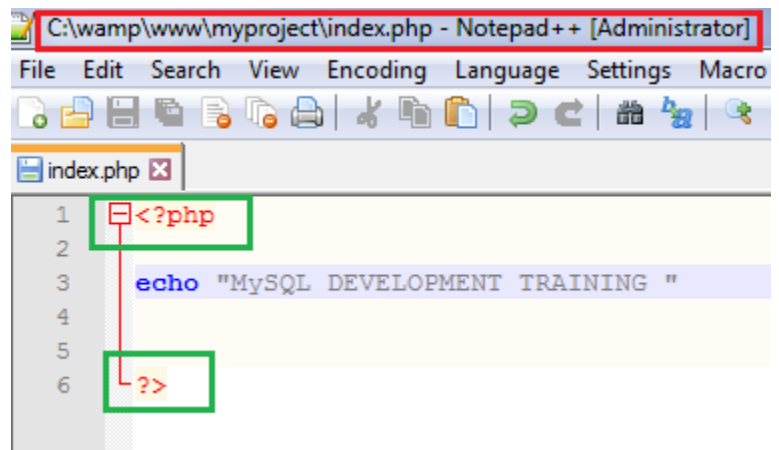


Yuxarıdakı şəkillərdən görüldüyü kimi **www directory** root qovluğu daxilində (**c:\wamp\www**) **mysqltraining** və **test** qovluqları olduğu üçün brauzerdə açılan localhost səhifəsində Your Projects başlığı altında bu qovluqların adları görünür.

İndi isə PHP-də ilk səhifəmizi yarada və ilk kodumuzu yaza bilərik.

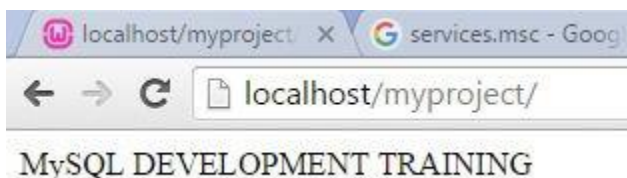
**c:\wamp\www** daxilində **myproject** qovluğu yaradıb bu qovluğun içində bir text faylı yaradaq və onu notepad++ editoru ilə açaq. Açılan pəncərədə file menyusundan save as seçərək bu faylı **index.php** adı ilə yadda saxlayaq.

Şəkildən görünür ki php kodlarını `<?php ?>` tag-ları arasında yazmaq lazımdır. PHP-də ekranda yazı yazdırmaq üçün `echo` komandasından istifadə olunur.



```
1 <?php
2
3 echo "MySQL DEVELOPMENT TRAINING "
4
5
6 ?>
```

Bu yazdığımız kodu işlətmək üçün brauzerə gəlib **localhost/myproject** və ya **localhost/myproject/index.php** yazıb enter-ə vururuq:



Və artıq ilk php kodumuzu wampserver-lə işə saldıq. İndi isə MySQL və PHP-nin əlaqələndirilməsinə keçə bilərik.

PHP-nin 5.5-ci versiyasına qədər ən çox istifadə olunan API interfeyslərdən biri **mysql\_\*** idi. Burada

**mysql\_connect()** - bazaya qoşulmaq üçün

**mysql\_query()** - sorğudan nəticə almaq üçün

**mysql\_fetch\_assoc()** - bazadan assosiativ şəkildə massiv çəkmək üçün

**mysql\_close()** - bazayla əlaqəni kəsmək üçün

funksiyaları istifadə olunurdu.

5.5 versiyasından sonra **mysqli\_\*** interfeysi istifadə edilir. Hansı ki yeni

**mysqli\_connect()**

**mysqli\_query()**

**mysqli\_fetch\_assoc()**

**mysqli\_close()**

funksiyalarına sahibdir və bu funksiyalar da yuxarıdakılarla eyni işi görürlər. **mysqli** interfeysini həmçinin obyekt yönümlü php-də də istifadə etmək mümkündür. Obyekt yönümlü üçün onun

**\$mysqli = new mysqli();** şəklində obyekt yaradılır.

Bu obyektə linki **\$mysqli** dəyişəni saxlayır.



**mysql** -dan başqa böyük layihələrlə işləmək üçün ən çox istifadə olunan API **PDO** adlanır.**PDO** obyekt yönümlü cəhətdən bir neçə verilənlər bazası modelləri ilə işləməyə imkan verir.

Verilənlər bazası ilə işləmək üçün yuxarıda sadaladığımız funksiyalar toplusunun PHP -nin konfigurasiyasında necə yer aldığına baxaq.Öncə qeyd etdiyimiz kimi PHP-nin konfigurasiya faylı olan **php.ini** faylı içində məhz **mysql** ilə işləmək üçün **dll extension kitabxanalar toplusu** vardır.Məsələn sırf mysql interfeysi üzərindən işləmək üçün burada **extension = php\_mysql.dll** faylı vardır.

İndi isə yuxarıda sadaladığımız funksiyalardan istifadə etməklə real bir nümunədə PHP və MySQL-in necə inteqrasiya olunmasına baxaq.

Öncə **create database new\_db;** sorğusu ilə **new\_db** adında yeni database yaradırıq.Xatırladaq ki bunu həm phpMyAdmin programından, həm MySQL console -dan həm də mysql workbench-dən etmək mümkündür.

Sonra bu database-də **employee** adında bir cədvəl yaradaq:

```
create table employee(  
id int not null primary key auto_increment,  
name varchar(20),  
surname varchar(30),  
age tinyint unsigned,  
mail varchar(20)) engine=InnoDB charset=utf8;
```

İndi isə **mysqli\_\*** kitabxanasının funksiyalarından istifadə etməklə **new\_db** bazasına qoşulub onun daxilindəki cədvələ sorğular göndərək.

Öncə **www** root qovluğu daxilində **mysqltraining** adlı qovluq yaradıb onun daxilində də **form.html** faylı yaradaq. Bu faylı **Notepad+** ilə açıb daxilində aşağıdakı kodları yazmaq:

```
.php x form.html x
<html>
<head>
<title>Qeydiyyat formu</title>
<meta charset="utf-8"/>
</head>
<body>
<form action="show.php" method="POST">
AD : <br/>
<input type="text" name="name" placeholder='your name'> <br/>
SOYAD : <br/>
<input type="text" name="surname" placeholder='your surname'> <br/>
YAŞ : <br/>
<input type="text" name="age" placeholder='your age'> <br/>
MƏİL : <br/>
<input type="text" name="mail" placeholder='your mail'> <br/><br/>
<input type="submit" value="TƏSDİQ ET">
</form>
</body>
</html>
```

Yuxarıdakı kodu işlətmək üçün brauzerdə adres çubuğuna **localhost/mysqltraining/form.html** yazmaq lazımdır:

← → ↻ localhost/mysqltraining/form.html

AD :

SOYAD :

YAŞ :

MƏİL :

Şəkildən də göründüyü kimi bu formu təsdiq etdikdə o **show.php** səhifəsinə yönələcək. Buna görə də indi **show.php** səhifəsini yaradıb php kodlarımızı və sql sorğularımızı yazmalıyıq.

**form.html** səhifəsini yaratdığımız kimi **show.php** səhifəsini də **mysqltraining** qovluğu daxilində yaradıb daha sonra notepad ++ programı ilə açırıq. PHP səhifəsində kodlar bildiyimiz kimi **<?php ?>** tagları daxilində yazılır.

Formdan gələcək məlumatlarda simvolların düzgün şəkildə oxunabilməsi üçün kodlaşma sistemini set edirik:

```
header('content-type: text/html; charset=utf-8');
```

Daha sonra `mysqli_connect()` funksiyasına `host`, `username`, `password` və baza adı parametrlərini verməklə bazaya qoşuluruq. (burada `host : localhost`, `usrname : root`, `password: ''`, `baza : new_db`)

```
// connect to mysql
```

```
$conn = mysqli_connect("localhost","root","","new_db");
```

Baza ilə əlaqəyə `$conn` adı verdik. Və şərt yazırıq ki əgər bazaya qoşularkən səhv baş verərsə `mysqli_connect_error()` metodu vasitəsilə səhv məlum olsun:

```
//connection error
```

```
if($conn === false){
```

```
die("Bazaya qosularken sehv bash verdi".mysqli_connect_error());}
```

Formdan gələcək məlumatların təhlükəsizliyini təmin etmək üçün `mysqli_real_escape_string` metodunu istifadə edirik:

(sql injection -a qarşı işlədilən konstruksiya)

```
//escape user inputs for security
```

```
$name = mysqli_real_escape_string($conn,$_POST['name']);
```

```
$surname = mysqli_real_escape_string($conn,$_POST['surname']);
```

```
$age = mysqli_real_escape_string($conn,$_POST['age']);
```

```
$mail = mysqli_real_escape_string($conn,$_POST['mail']);
```

İndi isə bazaya sql sorğularımızı yazma bilərik:

```
//set values to sql query
```

```
$sql1 = "insert into employee(name,surname,age,mail) values('$name','$surname','$age','$mail')";
```

```
$sql2 = "select id,name,surname,age,mail from employee";
```

Birinci sorğu employee cədvəlinə məlumatlar daxil etmək, ikinci isə həmin cədvəldən məlumatları oxumaq üçün yazılmışdır.

`mysqli_query()`; funksiyası ilə yazdığımız sorğuların nəticələrini geri almış oluruq:

```
//get queries' results
```

```
$result1 = mysqli_query($conn,$sql1);
```

```
$result2 = mysqli_query($conn,$sql2);
```



Sorğunun uğurlu olub olmamağını aşağıdakı kodlarla öyrənə bilərik:

```
//check your result
```

```
if($result1){
```

```
    echo "<h3 style='color:green'>Insert query successfull!</h3>.<br/>";
```

```
}else{
```

```
    echo "Sehv bash verdi";
```

```
}
```

Bundan sonra **mysqli\_fetch\_assoc()**; funksiyası ilə ikinci sorğuyla seçilən informasiyaları assosiativ massiv şəklində bazadan görüntüləmək mümkündür:

```
//show output of each row
```

```
while($row = mysqli_fetch_assoc($result2)){
```

```
    echo $row_id = $row['id']."<br>";
```

```
    echo $row_name= $row['name']."<br>";
```

```
    echo $row_surname= $row['surname']."<br>";
```

```
    echo $row_age= $row['age']."<br>";
```

```
    echo $row_mail= $row['mail']."<br><hr/>";
```

```
}
```

Sonda baza ilə işimiz bitdikdən sonra əlaqəni kəsmək lazımdır.Bunun üçün **mysqli\_close()**; metodunu işə salırıq:

```
//close connection
```

```
mysqli_close($conn);
```

**show.php** səhifəsindən hər dəfə yenidən ana səhifəyə qayıtmaq üçün php tagları bağlandıqdan sonra

**<a href="form.html">Go back form</a>**

linkini yerləşdirmək olar.

Son olaraq yazdığımız kodların düzgün işləməsini yoxlamaq üçün forma bir neçə məlumat daxil edə bilərik:

localhost/mysqltraining/form.html

AD :  
Etibar  
SOYAD :  
Vəzirov  
YAŞ :  
27  
MƏİL :  
etibar.vazirov@gmail.com

TƏSDİQ ET

localhost/mysqltraining/show.php

Insert query successfull!

1  
Etibar  
Vəzirov  
27  
etibar.vazirov@gmail.com

Go back form

localhost/mysqltraining/form.html

AD :  
Nicat  
SOYAD :  
Süleyman  
YAŞ :  
30  
MƏİL :  
nicat.suleyman@gmail.com

TƏSDİQ ET

localhost/mysqltraining/show.php

Insert query successfull!

1  
Etibar  
Vəzirov  
27  
etibar.vazirov@gmail.com

2  
Nicat  
Süleyman  
30  
nicat.suleyman@gmail.com

Go back form

1 • SELECT \* FROM new db.employee;

id	name	surname	age	mail
1	Etibar	Vəzirov	27	etibar.vazirov@gmail.com
2	Nicat	Süleyman	30	nicat.suleyman@gmail.com
NULL	NULL	NULL	NULL	NULL

## SQL injection.

**SQL injection web** projelərdəki ən ciddi çatışmazlıqlardan biridir. Belə ki web projelərdə istifadəçidən alınan informasiya ilə dinamik sorğu yaradılır. SQL sorğusu yaradılarkən arada sıxışdırılan hər hansı **meta simvol** SQL injection-a səbəb ola bilər. Meta simvollar bir program üçün xüsusi məna kəsb edən simvollara verilən addır. Meta simvollar misal olaraq **Javascript və PHP-də tərs slash ( \ )**, **SQL-də isə tək dırnaq ( ' ) və nöqtəli vergül ( ; )** göstərilə bilər. İki tək dırnaq arası **string** olaraq qəbul olunur, eləcə də nöqtəli vergül sql sorğuda bir **sətrin bitdiyini və yeni sətir** başladığını göstərir.

Sql injection problemini daha yaxşı qavramaq üçün wamp serverdə **www** root qovluğu daxilində **sql\_injection** qovluğu açaq və bu qovluqda **form.html** və **welcome\_page.php** səhifələri yaradaq. Belə ki form səhifəsində user üçün sadə daxilolma paneli olsun.

Bu kodu run etmək üçün brauzeri açıb

**localhost/sql\_injection/form.html**

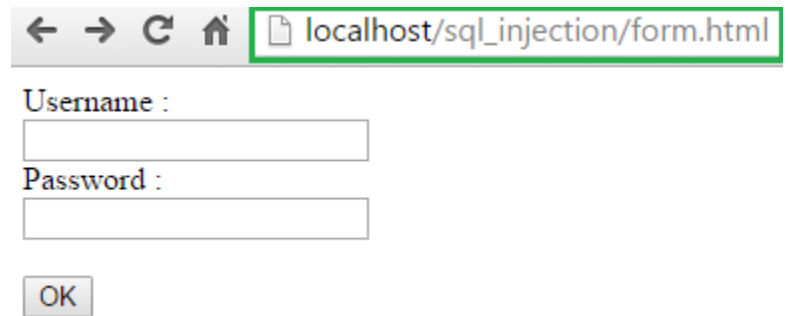
yazaraq enter-ə vurmaq lazımdır. Qarşımıza belə bir görüntü gələcək:

## WEB SECURITY: SQL INJECTION



```
form.html welcome_page.php
1 <html>
2
3 <head>
4 <title>sql_injection</title>
5 </head>
6
7 <form action="welcome_page.php" method='POST'>
8
9 Username : <br/>
10 <input type=text name='username'><br/>
11
12 Password : <br/>
13 <input type=text name='password'><br/><br/>
14
15 <input type="submit" value="OK">
16 </form>
17 </html>
```

İndi isə bu səhifənin  
yönəldiyi  
**welcome\_page.php**  
səhifəsini işləyib hazırlayaq.



← → ↻ 🏠 localhost/sql\_injection/form.html

Username :

Password :

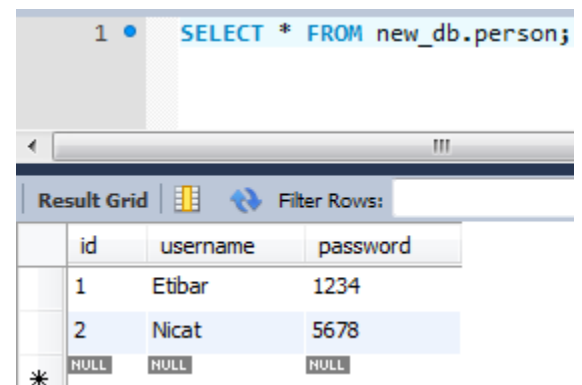
OK

Lakin bundan öncə gərək bir databaza yaradıb daxilində bütün userləri saxlayan cədvəl yaradaq. Öncəki dərstdə biz new\_db bazası yaratmışdıq. Bu bazada person cədvəlini aşağıdakı kimi yaradıb ona bir neçə məlumat insert edək. (bununla fərz edirik ki sanki bizdə userlərin artıq qeydiyyatdan keçdiyi bir cədvəl vardır)

```
create table person(  
id int not null primary key auto_increment,  
username varchar(20),  
password varchar(25)) engine=InnoDB charset=utf8;
```

Bir neçə user insert edək:

```
insert into  
person(username,password)  
values('Etibar','1234'),('Nicat','5678');
```



1 • SELECT \* FROM new\_db.person;

Result Grid | Filter Rows:

id	username	password
1	Etibar	1234
2	Nicat	5678
*	NULL	NULL

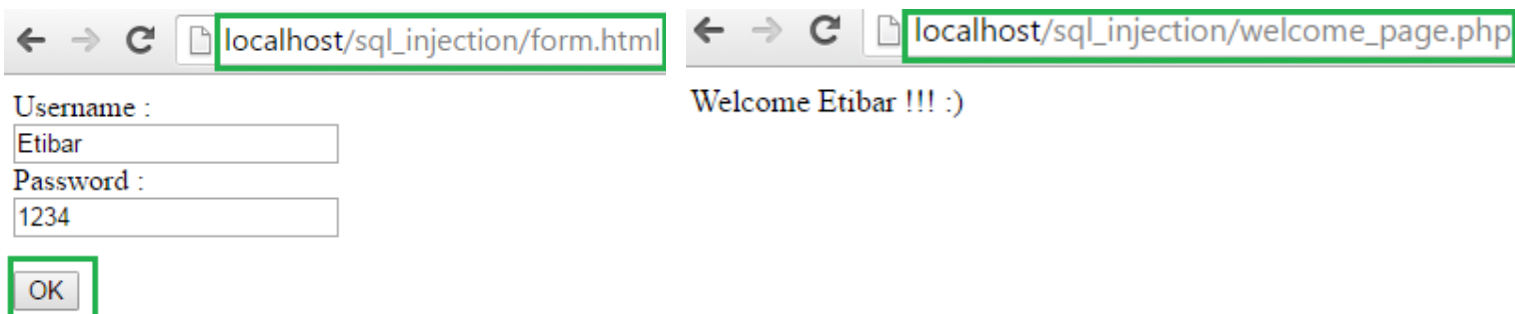
İndi isə **welcome\_page.php** səhifəsində new\_db bazasına qoşulub müvafiq sorğu və kodlarımızı yazmağa bilərik :

```

<?php
header('content-type: text/html; charset=utf-8');
//build connection
$conn = mysqli_connect('localhost','root','','new_db');
//check connection
if($conn === false){
    die("Error ocurred".mysqli_error());
}
//get user input data
$username = $_POST['username'];
$password = $_POST['password'];
//find user form person table
$query = "select * from person
where username='$username' and password='$password'";
//get query result
$result = mysqli_query($conn,$query);
//count of your results will be 1 or 0
$count = mysqli_num_rows($result);
// if count =1 it is OK but if count=0 then don't allow user sign_in
if($count){
    die("Welcome $username !!! :) ");
}else {
    die("Wrong username or password entered!!! Try again please!");
}
//close connection
mysqli_close($conn);
?>

```

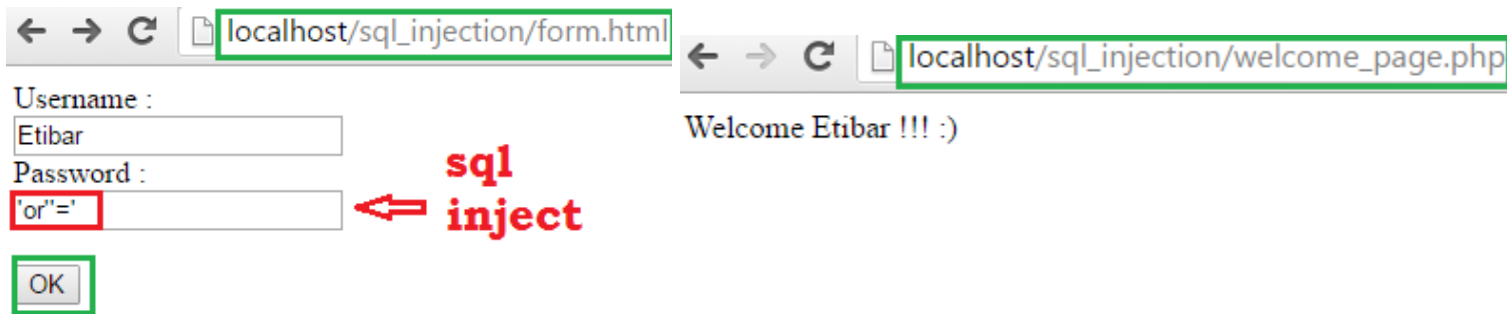
**form.html** səhifəsində ilk user üçün məlumatları düzgün daxil etdikdə **welcome\_page.php** səhifəsində **'Welcome Etibar'** yazısı görürük.



**Bu hissəyə qədər hər şey qaydasındadır deyə bilərdik...Lakin user üçün daxilolma formunda password**

hissəsinə **'or''='** ( və yaxud məsələn **'or'7'='7** şəklində də yazıla bilər) yazıldıqda Etibar userinin hesabına password-a ehtiyac olmadan daxil olmaq olur.

**qeyd:** yuxarıda **'or''='** kodunda cüt deyil yalnız tək dırnaqlar istifadə olunmalıdır, çünki mysql üçün meta simvol tək dırnaqdır.



Buna səbəb aşağıda şəkildə göstərilən prosesin baş verməsidir :

user inputdan gələn inject kodu : **'or''='**

sql sorğusu : **select \* from person  
where username='\$username' and password='\$password'**

**'or''='** kodu hara daxil olur?

**'or''='** **'or''='**  
...where username=**'\$username'** and password=**'\$password'**

\$username ve \$password dəyişənləri yerinə **'or''='** kodu daxil olur

...where username = **'or''='** and password = **'or''='** alınır. Nəticədə

**username= ''or''=' and password= ''or''='**

Demək password olaraq **'or' '='** daxil edildikdə select sorğusu bu şəkilə gəlir:

```
select * from person
```

```
where username="" or "" and password="" or "" ;
```

Bu sorğu isə doğru olduğu üçün icra olunur çünki sorğuda **or** məntiqi operatoru varsa onunla birlikdə yazılan şərtlərdən birinin düz olması kifayət edir ki sorğu icra olunsun. Burada isə or -la **""** (**boş = boş hansı ki hər zaman doğrudur**) şərti doğru olduğuna görə query problemsiz işləyir və bazaya sorğu gedir. Bu sql sorğuya müdaxilə (injection) deməkdir ki heç də arzuolunan hal deyil. Belə halların qarşısının alınması üçün mysql-də xüsusi bir funksiya **mysql\_real\_escape\_string()** vardır. Hansı ki formdan gələn məlumatları onun köməyi ilə sql-ə ötürsək injection-nun qarşısını almış olarıq. Başqa sözlə yuxarıdakı kodda

```
$username = $_POST['username'];
```

```
$password = $_POST['password'];
```

**hissəsini**

```
$username = mysql_real_escape_string($_POST['username']);
```

```
$password = mysql_real_escape_string($_POST['password']);
```

kodları ilə əvəz etmək lazımdır. Və bu dəyişiklik ilə artıq yenidən **'or' '='** yazmaqla injection-a cəhd etsək, user-in hesabına parolsuz daxil olmaq mümkün olmayacaq.

\*\*\*

**10-cu darsin sonu**

**Növbəti təlimlərdə görüşənədək əziz tələbələr...**

**Diqqətiniz üçün təşəkkürlər**





# İÇİNDƏKİLƏR

Verilənlər bazası.Relyasiyalı verilənlər bazası.SQL.....	1
MySQL-in qoşulması.Bazalarla iş.....	11
Cədvəllərdə sütun tipləri.....	24
Cədvəllərin yaradılması və silinməsi.....	25
Cədvələ məlumatların daxil edilməsi.(INSERT).....	29
Verilənləri saxlanılma sistemləri(Storage Engine).....	34
Primary key. (əsas açar).....	36
Update və Delete əməlləri.....	39
Cədvəldən məlumatların seçilməsi.Select sorğusu.....	44
Select sorğusu ilə birlikdə işlənən əməllər.....	47
Aqreqat funksiyalar (Aggregate functions).....	51
Müvəqqəti cədvəllərin yaradılması.....	55
INDEX və FOREIGN KEY.....	58
ACID prinsiplər.Atomarlıq prinsipi.....	66
DDL,DML və DQL anlayışları.....	70
MySQL-də script faylların run olunması.....	71
ALTER TABLE komandası və onunla birlikdə işlənən əməllər.....	76
Cədvəllərarası əlaqələr.(JOINS).....	82
MySQL-də tranzaksiyalar.....	90
MySQL-də triggerlər.....	94
MySQL-in PHP ilə əlaqələndirilməsi.PHP və Wampserver.....	100
SQL injection.....	113

**Əziz oxucular! Təlim vəsaiti ilə bağlı düşüncə və təkliflərinizi [etibar.vazirov@gmail.com](mailto:etibar.vazirov@gmail.com) ünvanına və ya <https://web.facebook.com/etibar.vezirov> facebook səhifəmə yazmağınızı xahiş edirəm.**

## **Müəllif haqqında**



### **Etibar Vəzirxan oğlu Vəzirov**

**29 iyul 1989 tarixində Abşeron rayonu Giləzi qəsəbəsində anadan olmuşdur.**

**2006-cı ildə orta məktəbi fərqlənmə ilə bitirib BDU -nun Mexanika-riyaziyyat fakültəsinə qəbul olunmuşdur. 2010-cu ildə bakalavr, 2013-cü ildə magistratura pilləsini fərqlənmə ilə başa vurmuşdur və həmin ildən bəri İT sektorunda freelancer Java Proqramçı kimi çalışır. **AZERBAIJAN CODERS' INSTITUTE** - un qurucusudur.**

**Hazırda BDU -nun Tətbiqi riyaziyyat və kibernetika fakültəsinin İT və Proqramlaşdırma kafedrasında Java Texnologiyaları üzrə mühazirəçi müəllimdir.**