1 PHP ile Veritabanı İşlemleri

Yaptığımız web sitelerinin daha kullanışlı olması için veritabanı sistemleri ile bağlantı kurup ihtiyaca göre verileri okuyup yazmasını isteriz.

1.1 Veritabanı Nedir?

Veritabanı düzenli bilgiler topluluğudur. Bilgisayar terminolojisinde, sistematik erişim imkânı olan, yönetilebilir, güncellenebilir, taşınabilir, birbirleri arasında tanımlı ilişkiler bulunabilen bilgiler kümesidir. Bir başka tanımı da, bir bilgisayarda sistematik şekilde saklanmış, programlarca işlenebilecek veri yığınıdır.

Bir veritabanı adından da anlaşılacağı gibi bilgilerin depolandığı hiyerarşik bir yapıdır. Biz web üzerinde kullanıcı adı, şifresi, e-posta adresi gibi bilgileri bu veri tabanlarında saklarız. Aynı bilgileri bir dosyaya yazıp gerektiğinde dosyayı açıp bilgileri okutmak da bir çözüm gibi görünse de bu işlem hem daha karmaşık ve zahmetli hem de daha yavaştır. Birçok veritabanı yönetim sistemi mevcuttur (**MySQL**, **PostgreeSQL**, **Oracle**, **DB2**, **SQL Server**). Biz uygulamalarımızda **MySQL** veritabanı sistemini kullanacağız.

1.2 Neden MySQL?

MySQL, altı milyondan fazla sistemde yüklü bulunan çoklu iş parçacıklı (multi-threaded), çok kullanıcılı (multi-user), hızlı ve sağlam (robust) bir veritabanı yönetim sistemidir. **UNIX**, **OS/2** ve Windows platformları için ücretsiz dağıtılmakla birlikte ticari lisans kullanmak isteyenler için de ücretli bir lisans seçeneği de mevcuttur. Linux altında daha hızlı bir performans sergilemektedir. Kaynak kodu açık olan **MySQL**'in pek çok platform için çalıştırılabilir ikilik kod halindeki indirilebilir sürümleri de mevcuttur. Ayrıca **ODBC** sürücüleri de **bulunduğu** için birçok geliştirme platformunda rahatlıkla kullanılabilir.

1.3 MySQL Avantajları

MySQL, tuttuğu tablolarla çok kullanıcılı sistemlerde söz konusu olan erişim hakları sorununu başarılı bir şekilde çözmektedir. **MySQL**'in **4.0** sürümü ile birlikte "**transaction**" desteği, **4.1** sürümüyle birlikte de alt sorgu desteği eklenmiştir.

PHP ile sorunsuz çalıştığı için **PHP** tabanlı web sitelerinde **MySQL** tercih edilir. Herhangi bir veri tabanı sistemini kullanmış birisi için **MySQL** kullanmak zor olmayacaktır. Standart **SQL** deyimleri bütün veritabanı sistemlerinde aynıdır.

1.4 PDO: PHP Data Object

PHP'nin eski versiyonlarında veritabanı bağlantısı için **mysql_conenct()** kullanılmakta idi. Fakat PHP 5.1'den sonra **PDO (PHP Data Objects)** sınıfları kullanılmaktadır. PDO, veritabanları için daha fazla esneklik sağlayan nesneye dayalı programlama imkânları sunmaktadır.

PHP'nin **PDO**'yu kullanabilmesi için **php.ini** dosyasında **php_pdo_mysql.dll** ve **php_pdo_sqlite.dll** extension'larının önündeki **#** işaretlerinin kaldırılmış olması gerekmektedir.

```
880 ;extension=php_pdo_firebird.dll
881 extension=php_pdo_mysql.dll
882 ;extension=php_pdo_oci.dll
883 ;extension=php_pdo_odbc.dll
884 ;extension=php_pdo_pgsql.dll
885 extension=php_pdo_sqlite.dll
```

Şekils. php.ini dosyası

1.5 PDO'nun Desteklediği Veritabanı Sistemleri

PDO, birçok farklı veritabanı yönetim sistemine bağlanmak için tek düzenli bir arabirim sunmaktadır. Böylece aynı PHP kodları ile sadece bağlantı arabirimini değiştirerek birçok veritabanı sistemi ile çalışma imkânı ortaya çıkmıştır. Böylece programcı her veritabanı için ayrı ayrı kod yazma zahmetinden kurtulmuş olur.

Aşağıdaki veritabanı sürücüleri **PDO** tarafından desteklenmektedir.

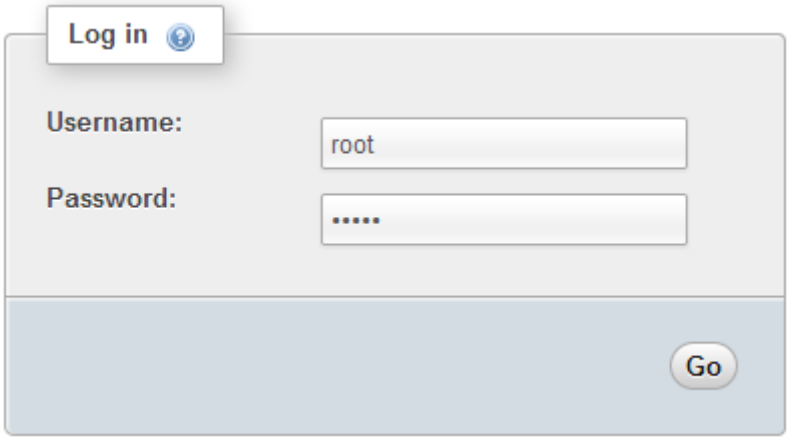
Driver name	Supported databases
PDO_CUBRID	Cubrid
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird/Interbase 6
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC and win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 and SQLite 2
PDO_SQLSRV	Microsoft SQL Server / SQL Azure
PDO_4D	4D

2 PHP'ile Örnek Veritabanı Uygulaması

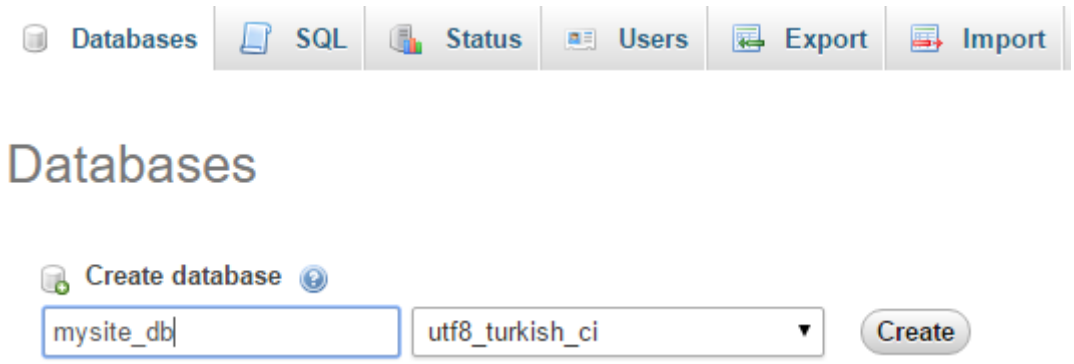
Örneğimizde bir ziyaretçi defteri uygulaması geliştireceğiz. Uygulamada deftere yazma, mesajları listeleme ve arama işlemleri gerçekleştirilecektir.

PHP uygulamasına geçmeden önce **PHPMyAdmin**'i kullanarak veritabanını ve tabloyu oluşturalım.

Yerel bilgisayardan **PHPMyAdmin**'i açmak için http://localhost/phpmyadmin adresine girlir. Gelen **Oturum Açma** ekranda **Kullanıcı Adı : root** ve **Parola : mysql** girilir. BU kulanıcı adı ve parola bilgileri sunucuya göre değişebilmektedir.

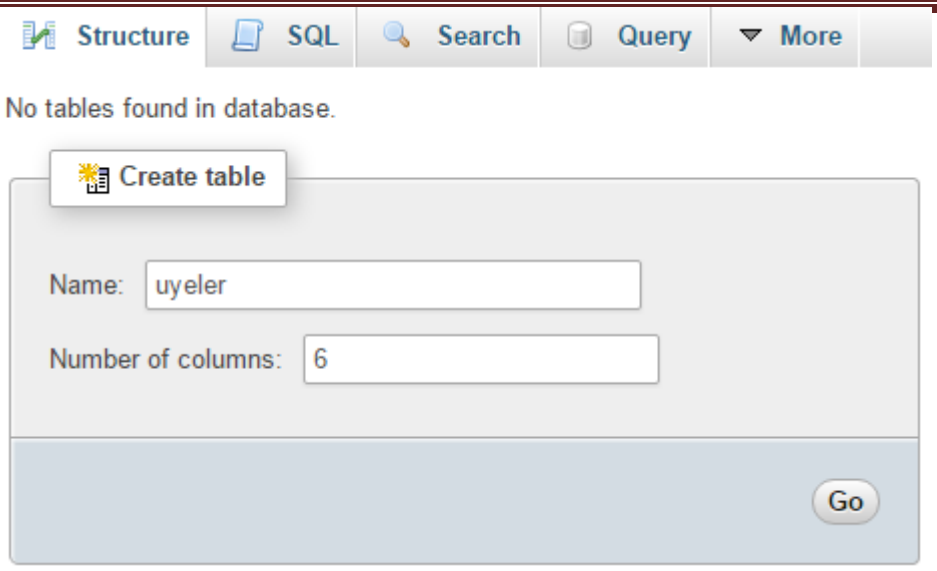


Databases sekmesinde **mysite_db** adlı yeni bir **database** (**veritabanı**) oluşturulur.



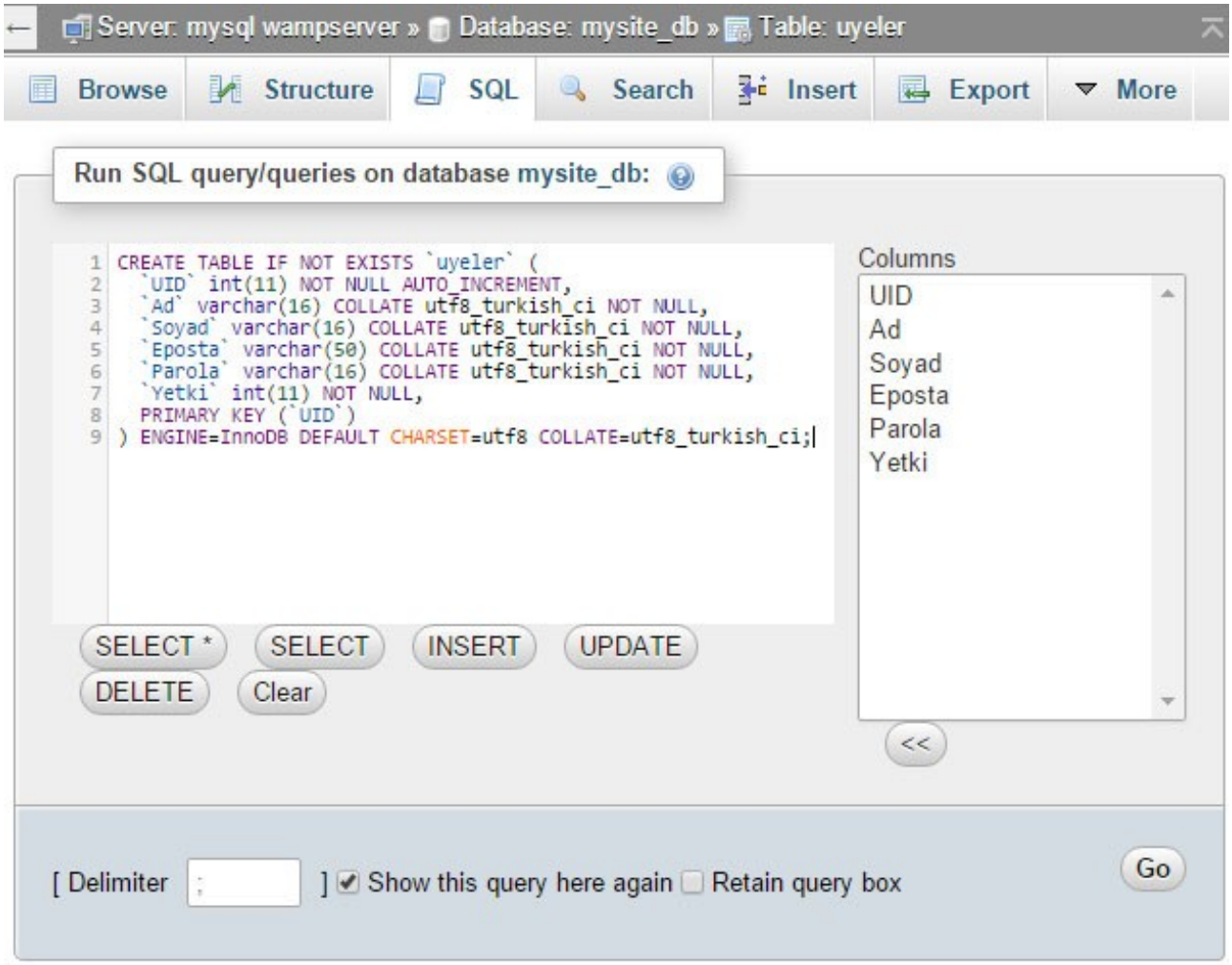
Bu aşamada veri tabanındaki tablolar oluşturulacak. Bunun için iki yöntem mevcuttur.

Birinci yöntem: **Structure** sekmesinde **Create Table** bölümü kullanılır.



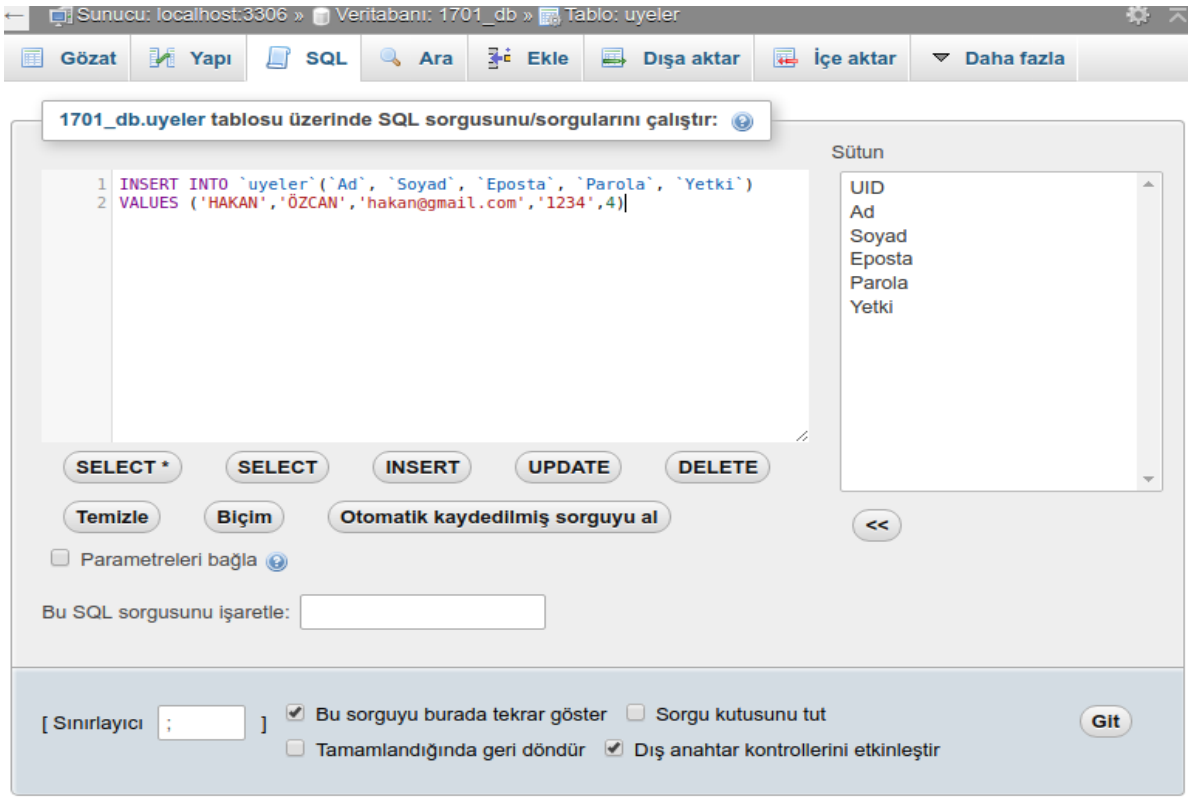
Go butonuna tıkladıktan sonra açılacak formda alan adları ve özellikleri girilir.

İkinci yöntemde **phpMyAdmin** de **SQL** sekmesine aşağıdaki kodları yazarak **uyeler tablosu** oluşturulur.



Yukarıdaki **SQL** ifadeleri ile **mysite_db** veritabanı içerisinde **üyeler** adlı tablo oluşturulur.

Şimdilik PHPMyAdmin in SQL penceresinden tabloya örnek birkaç kayıt girelim. Daha sonra kayıt ekleme işlemi Web formları ile yapılacak.



Bu aşamadan sonra PHP kodları ile MySQL veritabanına bağlanıp uygulama geliştirilebilecektir.

Veritabanı işlemleri yapılacak her sayfada **MySQL** Veritabanı Yönetcisi'ne ve oluşturulan veritabanı şemasına bir bağlantı kurulması gerekmektedir.
Bu amaçla **dbconn.php** adlı bir sayfa oluşturup bunu her sayfamızın başına include edelim.

Bir PDO sınıfı tanımlanırken bir **DSN "Data Source Name"** ile belirtilir. Hangi veritabanı sürücüsüne bağlanılacağı ve hangi veritabanının kullanılacağı DSN ile ifade edilmektedir. Ayrıca veritabanı **kullanıcı adı** ve **şifre** bilgilerinin de girilmesi gerekmektedir.

dbconn.php | listele.php

```php
<?php
//PHP sayfalarında Türkçe karakterlerin görüntülenmesi için
@header("Content-Type: text/html; charset=utf-8");
//Data Source Name bilgileri tanımlanıyor.
$dsn = 'mysql:host=localhost;dbname=mysite_db';
//Kullanıcı bilgileri. Bu örnekte root kulalnıcısı.
$dbuser = 'root';
$dbpass = 'mysql';
//Bağlantı kuruluyor.
try {
    $pdo = new PDO($dsn, $dbuser, $dbpass);
    $pdo->exec("SET NAMES 'utf8'; SET CHARSET 'utf8'");
    //echo "Bağlantı kuruldu";
} catch (PDOException $e) {
    //Eğer bağlantıda bir sorun olursa hata mesajı yazılacak.
    echo 'Bağlantı hatası: ' . $e->getMessage();
}
?>
```

2.1 Kayıt Listeleme

Bir tablodaki kayıtları listelemek için aşağıdaki PHP kodları kullanılır.

dbconn.php | listele.php | kaydet.php

```php
<?php
    include "dbconn.php";
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>PDO ile Kayıt Listeleme</title>
</head>
<body>
<h2>Kayıtlar Listesi</h2>
<?php
    $sql = "SELECT * FROM uyeler WHERE 1";
    $query = $pdo->prepare($sql);
    $query->execute();
    //Bir döngü ile bütün akyırlar listeleniyor.
    foreach($query as $row)
    {
        echo "<br>".$row["Ad"]." ".$row["Soyad"];
    }
    $pdo = NULL;     //Bağlantı sonlandırılıyor.
?>
</body>
</html>
```