BÖLÜM 3

PHP'DE MYSQL İLE VERİTABANI İŞLEMLERİ

ANAHTAR KAVRAMLAR

Veri tabanı, MySQL, Tablo, Alan, Kayıt

ÖĞRENME HEDEFLERİ

1. MySQL veri tabanı hakkında bilgi sahibi olma
2. Veri tabanı bağlantısı oluşturma
3. Veri tabanında tablo oluşturma
4. Veri tabanından istenilen verileri getirme
5. Verileri sıralı bir şekilde getirme
6. Veri ekleme, silme, güncelleme

GİRİŞ

Veri tabanı, organize edilmiş bilgi koleksiyonu olarak tanımlanmaktadır. Bu bilgiler genellikle belirli bir konuyla ya da bir mimari ile ilgilidir. Söz konusu veriler bir sistem tarafından erişilip yönetilir. Veri tabanları, verileri saklamak, düzenlemek, güncellemek ve sorgulamak için kullanılmaktadır. Büyük veya küçük ölçekte birçok sektör için önemli olan veri tabanlarının uygulama alanları ile ilgili bankalar, e-ticaret siteleri, müşteri ilişkileri yönetimi (CRM) yazılımları gibi birçok örnek verilebilir. Veri tabanı sistemleri, bilgilerin organize edilmesi, saklanması, erişilmesi, güncellenmesi ve yönetilmesi için günümüzün vazgeçilmez mimarileridir. Şekil 1 veri tabanı yönetim sistemlerinin önemi, özellikleri ve kullanılmasının temel nedenleri başlıklar halinde sunmaktadır.

Şekil 1. Veri tabanı yönetim sistemlerinin özellikleri

Veri yönetimi bakımından incelemek gerekirse, veri tabanları yapılandırılmış bir şekilde verilerin saklanmasını sağlamaktadır. Dolayısıyla verilerin tutarlılığı ve bütünlüğü korunmaktadır. Tablolar, ilişkiler ve kısıtlamalar aracılığıyla veriler düzenlenir ve bu da verilerin daha kolay yönetilmesini olanak tanımaktadır. Diğer bir durum ise söz konusu sitemler sayesinde aynı anda birden fazla kullanıcının verilere erişmesi ve güncellemesi mümkün olmaktadır. Bahsi geçen durum ise iş süreçlerinin verimliliğini artırmaktadır. Güvenlik ise veri tabanı sistemlerinin kullanımını zorunluluk haline getiren hayati konulardan biridir. Veri tabanlarında bulunan yetkilendirme, kimlik doğrulama ve şifreleme gibi önemli güvenlik özelliklerinin bulunması yetkisiz erişimlerin engellenmesini sağlamaktadır ve verilerin gizliliği korumaktadır. İlaveten, veri tabanları verilerin yedeklenmesini ve felaket durumlarında kurtarılmasını sağlayan mekanizmalar bulundurmaktadırlar. İlgili durum veri kaybını önler ve iş sürekliliğini sağlar. Veri analizi ise incelenmesi gereken önemli konulardan bir tanesidir. Veri tabanları sorgu dilleri aracılığıyla verilerin analiz edilmesine ve raporların oluşturulmasına olanak tanımaktadır. Böylece iş karar ve süreçlerinin veriye dayalı olarak uygulanmasını sağlamaktadır. Farklı kaynaklardan gelen verilerin birleştirilmesi ve entegrasyonu da veri tabanlarının önemli bir kullanım alanıdır; örneğin, bir şirketin farklı departmanlarından gelen verilerin bir araya getirilmesi ve ortak bir platformda kullanılması mümkündür. Bu nedenlerle, veri tabanları işletmelerde, web sitelerinde, mobil uygulamalarda, bilimsel araştırmalarda ve birçok başka alanda yaygın olarak kullanılmaktadır, veri yönetimi ve iş süreçlerinin etkin bir şekilde yürütülmesine yardımcı olmaktadır.

Farklı gereksinimlere ve kullanım senaryolarına yönelik çeşitli veri tabanı yönetim sistemleri bulunmaktadır. Hangi veri tabanı yönetim sisteminin sizin için en uygun olduğunu belirlemek, ihtiyaçlarınıza ve projenizin gereksinimlerine bağlı olarak değişkenlik gösterecektir. Veri

tabanı yönetim sistemleri için MySql, Oracle Database, Microsoft SQL Server, PostgreSQL, MongoDB, SQLite gibi popüler örnekler verilebilir. Bu bölümde veri tabanı bağlantısı oluşturma, veri tabanı ve tablo oluşturma, veri ekleme, silme seçme, güncelleme gibi işlemler anlatılacak ve örnekler verilecektir. Ancak öncelikle kısaca MySql veri tabanı yönetim sistemi ile ilgili kısaca bilgi verelim.

MySQL Veri Tabanı

MySql, web geliştirme alanında önemli bir rol oynayan popüler bir ilişkisel veritabanı yönetim sistemidir (RDBMS). MySql'in, ücretsiz ve açık kaynaklı olması küçük işletmelerden büyük kuruluşlara kadar birçok kullanıcı için onu tercih edilebilir kılmaktadır. Açık kaynaklı doğası, geniş bir topluluğun katkıda bulunmasını sağlamaktadır ve hızla gelişmesine olanak tanımaktadır. Hızlı ve verimli bir veritabanı yönetim sistemi olarak bilinen MySql, optimize edilmiş sorgu performansı ve düşük sistem kaynakları gereksinimi sayesinde web uygulamaları için ideal bir seçenek olarak öne çıkmaktadır. MySql, web uygulamaları, içerik yönetim sistemleri (CMS) ve e-ticaret platformları gibi çeşitli web tabanlı projelerde yaygın olarak kullanılmaktadır. Ayrıca, MySql Linux, Windows, macOS ve diğer birçok işletim sistemi üzerinde çalışabilir, bu da farklı platformlarda çalışan web uygulamaları için geniş bir uyumluluk sağlamaktadır. MySql, güvenlik duvarları, şifreleme ve yetkilendirme gibi bir dizi güvenlik özelliği sunmaktadır, böylece veri güvenliği sağlanır. Ölçeklenebilirlik açısından, MySql küçük projelerden büyük ölçekli kurumsal uygulamalara kadar geniş bir ölçeklenebilirlik aralığını destekler. Yüksek trafikli web siteleri ve uygulamaları için bile uygun performansı ve ölçeklenebilirliği sağlar. MySql'in geniş bir dokümantasyon ve aktif bir topluluğu vardır, kullanıcılar belgelerden öğrenme kaynaklarına kadar çeşitli kaynaklardan yararlanabilir ve sorunlarını çözmek için topluluğun desteğinden faydalanabilirler. Bu nedenlerle, MySql web geliştirme alanında yaygın olarak kullanılan bir veri tabanı yönetim sistemidir, hem yeni başlayanlar hem de deneyimli geliştiriciler tarafından tercih edilir ve çeşitli web tabanlı projelerde güvenilir bir veri tabanı çözümü olarak kabul edilmektedir. Bu bölümde aşağıdaki başlıklarda incelenen konuların tümü için MySql veri tabanı yönetim sistemi kullanılmıştır.

Veri Tabanı Bağlantısı

PHP ile veritabanına bağlanmak için iki yaygın yöntem vardır: MySqli (MySql Improved) ve PDO (PHP Data Objects). MySqli, PHP'nin MySql veritabanlarıyla etkileşim kurmak için sunulan geliştirilmiş bir API'dir. MySqli, PHP 5.0'dan itibaren kullanılabilir hale gelmiştir. MySqli'nin özellikleri arasında prosedürel ve nesne yönelimli programlama (OOP) tarzında kullanım, hazırlanmış ifadeler (prepared statements), işlem (transactions) ve çıktı parametreleri gibi özellikler bulunur. Kitabın ilerleyen bölümlerindeki örnekler sırasıyla Örnek 1:Mysqli nesne yönelimli, Örnek 2: Mysqli prosedürel ve Örnek 3: PDO kullanımlarını içermektedir. Aşağıda verilen örnek bir MySqli bağlantısını göstermektedir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// MySQLi bağlantısı oluşturma
$conn = new mysqli($servername, $username, $password);
// Bağlantıyı kontrol etme
if ($conn->connect_error) {
    die("Bağlantı hatası: " . $conn->connect_error);
} else {
    echo "Bağlantı başarılı";
}
?>
```

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// MySQLi bağlantısı oluşturma
$conn = mysqli_connect($servername, $username, $password);
// Bağlantıyı kontrol etme
if (!$conn) {
    die("Bağlantı hatası: " . mysqli_connect_error());
} else {
    echo "Bağlantı başarılı";
}
?>
```

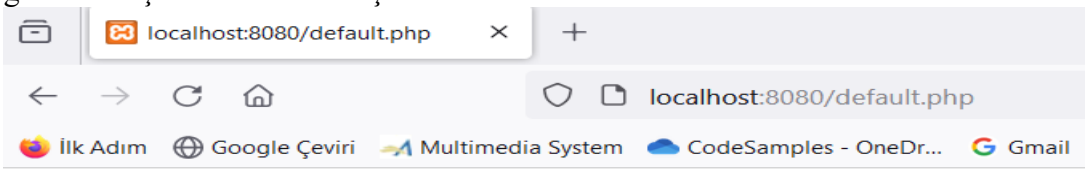
Bu kod, MySql veritabanına MySqli (MySql Improved Extension) kullanarak bağlanmayı denemektedir. İlk olarak, bağlantı parametreleri ($servername, $username, $password,) tanımlanır. Daha sonra, bu parametreler kullanılarak new mysqli() ile bir MySqli bağlantısı oluşturulur.

Bağlantının başarılı olup olmadığı if koşulu ile kontrol edilir. Eğer bağlantı hatası varsa ($conn->connect_error true ise), die() fonksiyonu ile birlikte bir hata mesajı gösterilir ve kod çalışması durdurulur. Aksi takdirde (else bloğu), "Bağlantı başarılı" mesajı ekrana yazdırılır.
if bloğunun içinde die() fonksiyonu kullanılır. Bu fonksiyon, belirtilen bir mesajı ekrana yazdırır ve işlemin çalışmasını durdurur. Yani, eğer bağlantı hatası varsa, kullanıcıya "Bağlantı hatası: [hata_mesajı]" şeklinde bir mesaj gösterilir ve kod çalışmayı durdurur. Eğer bağlantı hatası yoksa, else bloğu çalıştırılır. Bu durumda, echo komutuyla ekrana "Bağlantı başarılı" mesajı yazdırılır. Yani, eğer bağlantı başarılıysa, kullanıcıya bu bilgi verilir. Bu yapı, MySql veritabanıyla bağlantı kurarken oluşabilecek hataları kontrol ederek kullanıcıya uygun bir geri bildirim sağlar. PDO ise farklı veri tabanı türleriyle etkileşim kurmak için kullanılan bir erişim katmanıdır. PDO, çeşitli veritabanlarına (MySql, PostgreSQL, SQLite vb.) bağlanmak için kullanılabilir. Örnek bir PDO bağlantısı aşağıda verilmiştir.

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
try {
    // PDO bağlantısı oluşturma
    $conn = new PDO("mysql:host=$servername;dbname=$database", $username,
$password);
    // Hata modunu ayarlama
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Bağlantı başarılı";
} catch(PDOException $e) {
```

PDO yöntemi kullanılarak yapılan bağlantının bazı farkları bulunmaktadır. `$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION)` satırı PDO'nun hata işleme modunu ayarlar. setAttribute() yöntemi kullanılarak PDO'nun belirli bir özelliği ayarlanır. İlk parametre olarak PDO::ATTR_ERRMODE sabiti seçilir, bu sabit PDO'nun hata işleme modunu ayarlamak için kullanılır. İkinci parametre olarak PDO::ERRMODE_EXCEPTION sabiti kullanılır; bu sabit, PDO'nun hata işleme modunun istisna fırlatma modu olarak ayarlandığını belirtir. Yani, eğer bir PDO işlemi başarısız olursa (örneğin, bir sorgu hatası), bir istisna fırlatılır ve bu istisna, kodun işleyişi normalden farklı bir şekilde devam etmesini sağlar. Bu kod parçası, PDO'nun hata işleme modunu ayarlayarak, PDO'nun hata durumlarını ele almasını ve kodun daha güvenli ve kontrol edilebilir olmasını sağlar. Özellikle geliştirme aşamasında, bu modu kullanmak hata ayıklamayı kolaylaştırabilir ve uygulamanın güvenilirliğini artırabilir. Verilen kodlarda görüldüğü üzere her iki bağlantı çeşidi de sunucu adı (servername), kulanıcı adı (username), şifre (password) ve veritabanı (database) olmak üzere 4 adet parametre almaktadır. Varsayılan olarak sunucuadı=”localhost”, username=”root” değerleri almıştır Yukarıda verilen her iki kodun çalıştırılması sonucu elde edilen sayfa görüntüsü Şekil 2’de verilmiştir.



Şekil 2. MySqli ve PDO örneklerinin ekran görüntüsü

Ek Bilgi: PDO ve MySQLi arasındaki tercih, projenin ihtiyaçlarına ve kişisel tercihlere bağlıdır. PDO, farklı veritabanı motorlarını desteklediği için genellikle daha esnek ve taşınabilir bir seçenektir. Ancak, MySQLi'nin MySQL'e özgü optimizasyonlar ve özellikler sağladığı da unutulmamalıdır.

Veri Tabanı Oluşturma

Bildiğiniz üzere veri tabanı birçok tablodan meydana gelir. Bahsi geçen tabloları ilişkisel olarak tutmak ve tablolara veri ekleyebilmek için öncelikle bu tabloların içerisinde olduğu bir

veri tabanı gerekmektedir. MySql veri tabanı yönetim sisteminde veri tabanı yaratmak için ***create*** deyimi kullanılmaktadır. Örneğin Ankuzef isimli bir veri tabanı oluşturmak için;

```
CREATE DATABASE Ankuzef;
```

şeklinde kod yazmak gerekmektedir. Söz konusu veri tabanı oluşturma komutu MySql platformunda yazıldığı gibi bir php kodu ile de yazılıp sunucuda bir veri tabanı oluşturulması sağlanabilir. MySql veritabanı oluşturmak için hem MySqli hem de PDO kullanılabilmektedir. Sırasıyla her iki yöntemle de MySql veritabanı oluşturmanın kodları aşağıda verilmiştir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// bağlantı oluşturmak
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
// veri tabanı oluşturmak
$sql = "CREATE DATABASE Ankuzef";
if ($conn->query($sql) === TRUE) {
  echo "Veri tabanı Başarılı Bir Şekilde Oluşturuldu!!";
} else {
  echo "Veri Tabanı Oluşturulamadı: " . $conn->error;
}
$conn->close();
?>
```

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// bağlantı oluşturmak
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
// veri tabanı oluşturmak
$sql = "CREATE DATABASE Ankuzef";
if (mysqli_query($conn,$sql)) {
  echo "Veri tabanı Başarılı Bir Şekilde Oluşturuldu!!";
} else {
  echo "Veri Tabanı Oluşturulamadı: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
try {
  $conn = new PDO("mysql:host=$servername", $username, $password);
  // PDO hata modunu istisna olarak ayarlayın
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "CREATE DATABASE Ankuzef";
  // hiçbir sonuç döndürülmediği için exec() kullanın
  $conn->exec($sql);
  echo "Veri Tabanı Başarılı Bir Şekilde Oluşturuldu<br>";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}
$conn = null;
?>
```

Tablo Oluşturma

Tablo oluşturmak, ilişkisel veritabanlarında veri saklamak için temel bir işlemdir. MySql gibi ilişkisel veritabanı yönetim sistemlerinde bir tablo, belirli bir veri tipine sahip bir dizi sütun ve bu sütunlarla ilişkilendirilmiş veri kayıtlarından oluşur. Bir tablo oluşturmak için bazı bileşenlere dikkat etmek gerekmektedir. Her tablo, veri tabanında benzersiz bir ad ile tanımlanması gerekmektedir. Bir tablonun her bir sütunu, sakladığı veri türünü belirtir. Örneğin, bir kullanıcı tablosu oluştururken, ad, soyad, e-posta gibi bilgileri saklayan sütunlar tanımlanabilir. Her sütunun belirli bir veri tipi olmalıdır.

Tablo oluşturmak için MySql'de CREATE TABLE ifadesi kullanılır. kullanicilar isimli bir tablo oluşturmak için SQL kodu aşağıdaki gibidir.

```sql
CREATE TABLE Kullanicilar (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    ad VARCHAR(30) NOT NULL,
    soyad VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    kayit_tarihi TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

Hem MySqli hem de PDO kullanarak MySql tablosu oluşturmak oldukça basittir. İlk olarak, her iki yöntemle de bağlantı kurmamız gerekmektedir. Ardından, SQL sorgularını kullanarak tabloyu oluşturabiliriz. Bahsi geçen örnekler aşağıda verilmiştir.

Örnek 1:

```php
<?php
// MySQL bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = " Ankuzef";
$conn = new mysqli($servername, $username, $password, $dbname);
// Bağlantı kontrolü
if ($conn->connect error) {
```

Örnek 2:

```php
<?php
// MySQL bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = " Ankuzef";

$conn = mysqli_connect($servername, $username, $password, $dbname);
// Bağlantı kontrolü
if (!$conn) {
    die("Bağlantı hatası: " . mysqli_connect_error());
}
// Tablo oluşturma sorgusu
$sql = "CREATE TABLE kullanicilar (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    ad VARCHAR(30) NOT NULL,
    soyad VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    kayit_tarihi TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";
// Sorguyu çalıştırma
if (mysqli_query($conn,$sql)) {
    echo "Tablo başarıyla oluşturuldu.";
} else {
    echo "Tablo oluşturma hatası: " . mysqli_error($conn);
}
// Bağlantıyı kapatma
mysqli_close($conn);
?>
```

Örnek 3.

```php
<?php
// PDO bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // Hata modunu ayarlama: PDOException fırlatma
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // Tablo oluşturma sorgusu
    $sql = "CREATE TABLE kullanicilar (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        ad VARCHAR(30) NOT NULL,
        soyad VARCHAR(30) NOT NULL,
        email VARCHAR(50),
        kayit_tarihi TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
    )";
    // Sorguyu çalıştırma
    $conn->exec($sql);
    echo "Tablo başarıyla oluşturuldu.";
} catch(PDOException $e) {
    echo "Tablo oluşturma hatası: " . $e->getMessage();
}
// Bağlantıyı kapatma
$conn = null;
?>
```

Veri Ekleme

MySql'de INSERT ifadesi, bir tabloya yeni bir kayıt eklemek için kullanılır. INSERT INTO ifadesi, veri eklenmek istenen tabloyu belirtir ve VALUES ifadesi, eklenen verilerin değerlerini belirtir. kullanicilar isimli tabloya herhangi bir kayıt eklemek için gerekli sorgu aşağıda verilmiştir.

```
INSERT INTO kullanicilar (ad, soyad, email) VALUES ('mehmet', 'nallıhan',
'nallihan@example.com');
```

Yukarıda verilen kullanicilar tablosuna veri eklemek için gerekli olan MySqli ve PDO kodları ise aşağıda verilmiştir.

Örnek 1:

```php
<?php
// MySQL bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
$conn = new mysqli($servername, $username, $password, $dbname);
// Bağlantı kontrolü
if ($conn->connect_error) {
    die("Bağlantı hatası: " . $conn->connect_error);
}
// Veri ekleme sorgusu
$sql = "INSERT INTO kullanicilar (ad, soyad, email) VALUES ('mehmet',
'nallihan', 'nallihan@example.com')";
// Sorguyu çalıştırma
if ($conn->query($sql) === TRUE) {
    echo "Yeni kayıt başarıyla eklendi.";
} else {
    echo "Veri ekleme hatası: " . $conn->error;
}
// Bağlantıyı kapatma
$conn->close();
?>
```

Örnek 2:

```php
<?php
// MySQL bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Bağlantı kontrolü
if (!$conn) {
    die("Bağlantı hatası: " . mysqli_connect_error());
}
// Veri ekleme sorgusu
$sql = "INSERT INTO kullanicilar (ad, soyad, email) VALUES ('mehmet',
'nallihan', 'nallihan@example.com')";
// Sorguyu çalıştırma
if (mysqli_query($conn,$sql)) {
    echo "Yeni kayıt başarıyla eklendi.";
} else {
    echo "Veri ekleme hatası: " . mysqli_error($conn);
}
// Bağlantıyı kapatma
mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
// PDO bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // Hata modunu ayarlama
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Veri ekleme sorgusu
    $sql = "INSERT INTO kullanicilar (ad, soyad, email) VALUES ('mehmet',
'nallihan', 'nallihan@example.com')";

    // Sorguyu hazırlama ve çalıştırma
    $stmt = $conn->prepare($sql);
    $stmt->execute();

    echo "Yeni kayıt başarıyla eklendi.";
} catch(PDOException $e) {
    echo "Veri ekleme hatası: " . $e->getMessage();
}
// Bağlantıyı kapatma
$conn = null;
```

Çoklu Kayıt Ekleme

Birden fazla SQL ifadesi mysqli_multi_query() fonksiyonu ile yürütülmelidir. Aşağıdaki örnekler kullanicilar isimli tabloya 3 yeni kayıt eklemektedir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
// Bağlantı Oluştur
$conn = new mysqli($servername, $username, $password, $dbname);
// Bağlantıyı Kontrol Et
if ($conn->connect_error) {
  die("Başarısız Bağlantı: " . $conn->connect_error);
}
$sql = "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('ahmet', 'ahmet', 'ahmet@example.com');";
$sql .= "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('mehmet', 'mehmet', 'mehmet@example.com');";
$sql .= "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('hasan', 'hasan', 'hasan@example.com')";
if ($conn->multi_query($sql) === TRUE) {
  echo "Yeni kayıtlar eklendi";
} else {
  echo "Hata: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>
```

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

// Bağlantı Oluştur
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Bağlantıyı Kontrol Et
if (!$conn) {
  die("Başarısız Bağlantı: " . mysqli_connect_error());
}

$sql = "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('ahmet', 'ahmet', 'ahmet@example.com');";
$sql .= "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('mehmet', 'mehmet', 'mehmet@example.com');";
$sql .= "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('hasan', 'hasan', 'hasan@example.com')";

if (mysqli_multi_query($conn, $sql)) {
  echo "Yeni kayıtlar başarılı bir şekilde eklendi";
} else {
  echo "Hata: " . $sql . "<br>" . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
  // PDO hata modunu istisna olarak ayarlayın
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  // işlemi başlatın
  $conn->beginTransaction();
  // Sql Sorguları
  $conn->exec("INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('ahmet', 'ahmet', 'ahmet@example.com');");
  $conn->exec("INSERT INTO kullanicilar (ad, soyad, email)
  VALUES ('mehmet', 'mehmet', 'mehmet@example.com')");
  $conn->exec("INSERT INTO kullanicilar (ad, soyad, email)
  VALUES ('hasan', 'hasan', 'hasan@example.com')");
  // işlemi gerçekleştirin
  $conn->commit();
  echo "Yeni kayıtlar başarılı bir şekilde eklendi!!!";
} catch(PDOException $e) {
  // İşlemin başarısız olduğu durumda geri alma
  $conn->rollback();
  echo "Hata: " . $e->getMessage();
}
$conn = null;
?>
```

Yukarıda örneği verilen PDO kullanımını açıklamak yerinde olacaktır. PDO kullanarak gerçekleştirilen bir işlemi (transaction) tamamlamak için **commit()** yöntemini kullanır. Bir işlem (transaction) başarıyla tamamlandığında, veritabanındaki tüm değişiklikler kalıcı hale gelir ve veritabanı kilidi kaldırılır. Ardından, kullanıcıya "Yeni kayıtlar başarılı bir şekilde eklendi" mesajı gösterilir. Ancak, **try** bloğundaki kodda bir hata oluştuğunda, bu kod **catch** bloğuna girer. Bu durumda, **rollback()** yöntemi çağrılarak işlem (transaction) geri alınır ve yapılan tüm değişiklikler geri alınır, böylece veritabanı önceki durumuna döner. Son olarak, kullanıcıya bir hata mesajı gösterilir. Bu mesaj, **PDOException** sınıfından türetilen hatanın **getMessage()** yöntemi aracılığıyla alınır ve kullanıcıya gösterilir.

Hazırlanan İfadeler ve Bağlı Parametreler

Hazırlanan ifadeler (prepared statements) ve bağlı parametreler, veritabanı sorgularını güvenli ve etkili bir şekilde çalıştırmak için kullanılan bir tekniktir. Bu teknik, SQL enjeksiyon saldırılarına karşı koruma sağlar ve sorguların daha verimli bir şekilde çalışmasına olanak tanır.

Hazırlanan İfadeler (Prepared Statements):

Hazırlanan ifadeler, bir veritabanı sorgusunun önceden derlenmiş bir kalıba (template) sahip olduğu ve bu kalıbın veritabanına gönderilmeden önce doldurulduğu sorgu türüdür. Bu sayede,

sorgunun güvenliği artırılır ve aynı sorgu farklı parametrelerle tekrar tekrar kullanılabilir. Hazırlanan ifadeler genellikle veritabanı sunucusunda sorgunun ön işlemesi (pre-processing) yapılır, böylece sorgular daha hızlı çalışabilir. Hazırlanan ifadeler genellikle iki kısımdan oluşur:

1. **Sorgu kalıbı**: Sorgunun sabit kısmı, değişmeyen kısmıdır. Genellikle parametre yerine bir placeholder (yer tutucu) olarak kullanılır.
2. **Parametreler**: Sorgunun değişken kısmı, her sorguda değişen kısımdır. Bu parametreler sorgu gönderilmeden önce sorgu kalıbına yerleştirilir.

Bağlı Parametreler (Bound Parameters):

Bağlı parametreler, hazırlanan ifadelerin parametrelerine değer atanmasında kullanılan değerlerdir. Bağlı parametreler, sorgu gönderilmeden önce sorgu kalıbına yerleştirilir ve sorgunun çalıştırılması sırasında bu parametreler sorguya bağlanır. Bağlı parametreler, veritabanı sunucusuna değerlerin güvenli bir şekilde aktarılmasını sağlar ve SQL enjeksiyon saldırılarını önler.

Bağlı parametreler genellikle kullanılan programlama dili ve veritabanı sorgu arayüzüne göre farklılık gösterebilir. Örneğin, PHP ile PDO veya MySQLi kullanırken bağlı parametreler "?" veya ":name" gibi bir placeholder olarak ifade edilir ve sorgu çalıştırılırken parametreler bu placeholder'lara bağlanır. Hazırlanan ifadeler ve bağlı parametreler, veritabanı sorgularını güvenli hale getirirken aynı zamanda performansı da artırır. Bu teknikler, modern web uygulamalarında yaygın olarak kullanılan güvenlik önlemlerindendir. Hazırlanmış ve bağlı ifadelere ait örnekler sırasıyla MySqli ve PDO şeklinde aşağıda verilmiştir.

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
// Bağlantı yarat
$conn = new mysqli($servername, $username, $password, $dbname);
// Bağlantıyı kontrol et
if ($conn->connect_error) {
  die("Başarısız Bağlantı: " . $conn->connect_error);
}
// hazırlayın ve bağlayın
$stmt = $conn->prepare("INSERT INTO kullanicilar (ad, soyad, email) VALUES
(?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);
// parametreleri ayarlayın ve çalıştırın
$firstname = "ahmet";
$lastname = "ahmet";
$email = "ahmet@example.com";
$stmt->execute();

$firstname = "mehmet";
$lastname = "mehmet";
$email = "mehmet@example.com";
$stmt->execute();

$firstname = "hasan";
$lastname = "hasan";
$email = "hasan@example.com";
$stmt->execute();

echo "yeni kayıtlar başarılı bir şekilde eklendi.";
$stmt->close();
$conn->close();
?>
```

$stmt->bind_param("sss", $firstname, $lastname, $email); bu kod satırı parametleri SQL sorgusuna bağlar ve veritabanına parametrelerin ne olduğunu söyler.”sss” argümanı, parametrelerin olduğu veri türlerini listeler. S karakteri MySql’e parametrenin bir string olduğunu ifade eder. Argüman dört türden biri olabilir:

- i-integer
- d-double
- s-string
- b-BLOB

Aşağıdaki örnekte PDO'da hazır deyimler ve bağlı parametreler kullanılmaktadır:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
  // PDO hata modunu istisna olarak ayarlayın
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  // PDO hata modunu istisna olarak ayarlayın
  $stmt = $conn->prepare("INSERT INTO kullanicilar (ad, soyad, email)
  VALUES (:ad, :soyad, :email)");
  $stmt->bindParam(':ad', $firstname);
  $stmt->bindParam(':soyad', $lastname);
  $stmt->bindParam(':email', $email);

  // satır ekle
  $firstname = "mehmet";
  $lastname = "mehmet";
  $email = "mehmet@example.com";
  $stmt->execute();

  // satır ekle
  $firstname = "ahmet";
  $lastname = "ahmet";
  $email = "ahmet@example.com";
  $stmt->execute();

  // satır ekle
  $firstname = "hasan";
  $lastname = "hasan";
  $email = "hasan@example.com";
  $stmt->execute();

  echo "Yeni kayıtlar başarılı bir şekilde eklendi!!!!!!";
} catch(PDOException $e) {
  echo "Hata: " . $e->getMessage();
}
$conn = null;
?>
```

Veri Seçme (Select), Şart İfadesi (Where),Veri Sıralama (Order By)

"**SELECT**" ifadesi, bir veya daha fazla sütunu seçmek veya belirli bir koşulu karşılayan satırları seçmek için kullanılır. Temel kullanım;

SELECT sütun1,sütun2, …. FROM tablo_adı

şeklindedir. Bu ifade, tablodaki belirtilen sütunları seçer. Eğer tüm sütunları seçmek istiyorsak, "*" karakteri kullanılır. **SELECT** ifadesi ayrıca **WHERE** koşulu ile birlikte kullanılarak belirli bir koşulu karşılayan satırları seçmek için kullanılabilir. Örnek bir where sorgusu aşağıda verilmiştir.

SELECT sütun1,sütun2, …. FROM tablo_adı WHERE koşul ifadesi

Burada "**where**", seçilen satırları filtrelemek için kullanılır. Koşul, bir veya daha fazla sütunun belirli bir değere eşit olması, belirli bir aralıkta olması, bir örüntü ile eşleşmesi veya başka bir mantıksal ifade olabilir. Seçilen verileri belirli bir sütuna göre sıralamak için ise **ORDER BY** ifadesi kullanılır. Örneğin:

SELECT sütun1, sütun2, ... FROM tablo_adı ORDER BY sütun3 ASC;

Burada "sütun3" sütununa göre sıralama yapılır ve "**ASC**" parametresi küçükten büyüğe (artan) sıralama yapılmasını belirtir. "**DESC**" parametresi ise büyükten küçüğe (azalan) sıralama yapılmasını sağlar.

SELECT id, soyad, yaş FROM kullanicilar WHERE soyad ='yıldırım'ORDER BY yaş DESC örneğini inceleyelim. Bu örnekte, "**kullanicilar**" tablosundan "**ad**", "**soyad**" ve "**id**" sütunları seçilir. Ancak, sadece "yaş" sütunu 30'dan büyük olan kullanıcılar seçilir. Son olarak, "yaş" sütununa göre büyükten küçüğe sıralama yapılır. Bahsi geçen sorgu ile ilgili örnekler aşağıda verilmiştir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
// bağlantı oluştur
$conn = new mysqli($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if ($conn->connect_error) {
  die("başarısız bağlantı: " . $conn->connect_error);
}
$sql = "SELECT id, ad, soyad FROM kullanicilar WHERE soyad='yıldırım' ORDER
BY id DESC";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // her satırın çıktı verisi
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Ad: " . $row["ad"]. " " . $row["soyad"].
"<br>";
  }
} else {
  echo "0 results";
}
$conn->close();
?>
```

Yukarıda verilen örneği açıklayalım. Öncelikle kullanicilar tablosundan id, ad ve soyad sütunlarını seçen bir SQL sorgusu oluşturuyoruz. SQL sorgusu ilgili sütunları seçerken soyadı “yıldırım” olan kayıtları getiriyor ve id sütununa göre azalan sıralama yapmaktadır. Sonraki

kod satırı sorguyu çalıştırır ve elde edilen verileri **$result** adlı bir değişkene koyar. Daha sonra işlev, **num_rows()** ile döndürülen sıfırdan fazla satır olup olmadığını kontrol eder. Döndürülen sıfırdan fazla satır varsa, işlev **fetch_assoc()** tüm sonuçları döngü yapabileceğimiz ilişkisel bir diziye yerleştirir. Yanı sıra **fetch_both()** ve **fetch_num()** ile de sonuçlar döndürülebilir. fetch_both(), sorgudan dönen verileri hem bir ilişkisel dizi (assoc array) hem de bir sayısal dizi (numeric array) olarak alır. fetch_num() ise sorgudan dönen verileri bir sayısal dizi olarak alır. Her bir satır, sütunların sıra numaralarıyla ilişkilendirilir. Yani, sorgudan alınan her bir satır bir sayısal dizi olarak döner. **while()** dögüsü, sonuç kümesi boyunca döngü yapar ve id, ad ve soyadı sütunlarındaki verileri getirir.

Aşağıdaki örnek, MySQLi yordamsal yöntemiyle yukarıdaki örneğin aynısını göstermektedir:

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
// bağlantı yarat
$conn = mysqli_connect($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if (!$conn) {
  die("başarısız bağlantı: " . mysqli_connect_error());
}
$sql = " SELECT id, ad, soyad FROM kullanicilar WHERE soyad='yıldırım' ORDER
BY id DESC ";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // her satırın çıktı verisi
  while($row = mysqli_fetch_assoc($result)) {
    echo "id: " . $row["id"]. " - Ad: " . $row["ad"]. " " . $row["soyad"].
"<br>";
  }
} else {
  echo "0 results";
}

mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

// bağlantı oluştur
$conn = new mysqli($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if ($conn->connect_error) {
  die("Başarısız bağlantı: " . $conn->connect_error);
}

$sql = " SELECT id, ad, soyad FROM kullanicilar WHERE soyad='yıldırım' ORDER
BY id ASC ";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  echo "<table><tr><th>ID</th><th>Name</th></tr>";
  // her satırın çıktı verisi
  while($row = $result->fetch_assoc()) {
    echo "<tr><td>".$row["id"]."</td><td>".$row["ad"]."
".$row["soyad"]."</td></tr>";
  }
  echo "</table>";
} else {
  echo "0 results";
}
$conn->close();
?>
```

Veri Silme (Delete)

Herhangi bir kaydı silmek için delete ifadesi kullanılmaktadır. Delete ifadesinin genel kullanımı:

DELETE from tablo_adı where koşul ifadesi

şeklindedir.

Dikkat Edelim: DELETE söz dizimindeki WHERE yan tümcesine dikkat edin: WHERE yan tümcesi, hangi kaydın veya kayıtların silinmesi gerektiğini belirtir. WHERE deyimini atlarsanız tüm kayıtlar silinecektir!

Delete ifadesi ile ilgili örnekler aşağıda verilmiştir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

// bağlantı oluştur
$conn = new mysqli($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if ($conn->connect_error) {
  die("başarısız bağlantı: " . $conn->connect_error);
}
// sql to delete a record
$sql = "DELETE FROM kullanicilar WHERE id=3";

if ($conn->query($sql) === TRUE) {
  echo "Kayıt başarılı bir şekilde silindi";
} else {
  echo "kayıt silinemedi: " . $conn->error;
}
$conn->close();
?>
```

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Bağlantı Hatası: " . mysqli_connect_error());
}
// sql to delete a record
$sql = "DELETE FROM kullanicilar WHERE id=3";

if (mysqli_query($conn, $sql)) {
  echo "kayıt başarılı bir şekilde silindi";
} else {
  echo "hata: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
  // PDO hata modunu istisna olarak ayarlayın
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  // kayıt silmek için sql sorgusu
  $sql = "DELETE FROM kullanicilar WHERE id=3";

  // kayıt silmek için sql
  $conn->exec($sql);
  echo "Kayıt başarılı bir şekilde silindi";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

Veri Güncelleme (Update)

Herhangi bir kaydı güncellemek için update ifadesi kullanılmaktadır. Update ifadesinin genel kullanımı:

UPDATE tablo_adı SET sütun1=değer, sütun2=değer … where koşul ifadesi şeklindedir.

Dikkat Edelim: UPDATE söz dizimindeki WHERE yan tümcesine dikkat edin: WHERE yan tümcesi, hangi kaydın veya kayıtların güncellenmesi gerektiğini belirtir. WHERE deyimini atlarsanız tüm kayıtlar güncelenecektir!

Update ifadesi ile ilgili örnekler aşağıda verilmiştir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "mahmut";

// bağlantı oluştur
$conn = new mysqli($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if ($conn->connect_error) {
  die("başarısız bağlantı: " . $conn->connect_error);
}

$sql = "UPDATE kullanicilar SET soyad='kılıçaslan' WHERE id=2";

if ($conn->query($sql) === TRUE) {
  echo "kayıt başarılı bir şekilde güncellendi";
} else {
  echo "hata: " . $conn->error;
}

$conn->close();
?>
```

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

// bağlantı oluştur
$conn = mysqli_connect($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if (!$conn) {
  die("başarısız bağlantı: " . mysqli_connect_error());
}

$sql = "UPDATE kullanicilar SET soyad='kılıçaslan' WHERE id=2";

if (mysqli_query($conn, $sql)) {
  echo "Kayıt başarılı bir şekilde güncellendi!";
} else {
  echo "hata: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
  // PDO hata modunu istisna olarak ayarlayın
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "UPDATE kullanicilar SET soyad='kılıçaslan' WHERE id=2";
  //
  $stmt = $conn->prepare($sql);

  //
  $stmt->execute();
  //
  echo $stmt->rowCount() . " Kayıt başarılı bir şekilde güncellendi";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}
$conn = null;
?>
```

Sınırlı Veri Seçme (Limit Data)

MySQL, döndürülecek kayıt sayısını belirtmek için kullanılan bir LIMIT cümleciği sağlar. LIMIT ifadesi, çok sayfalı sonuçları veya sayfalandırmayı SQL ile kodlamayı kolaylaştırır ve büyük tablolarda çok kullanışlıdır. Çok sayıda kaydın döndürülmesi performansı etkileyebilir.

"kullanicilar" adlı bir tablodan 1'den 30'a kadar (dahil) tüm kayıtları seçmek istediğimizi varsayalım. SQL sorgusu daha sonra şöyle görünecektir:

$sql = "SELECT * FROM kullanicilar LIMIT 30";
Yukarıdaki SQL sorgusu çalıştırıldığında ilk 30 kaydı döndürecektir. Ancak 16 - 25 (dahil) arasındaki kayıtları seçmek istersek Mysql'de bu durum için OFFSET kullanılmaktadır. Aşağıdaki SQL sorgusu 16. kayıttan başlayarak yalnızca 10 kayıt döndür.

$sql = "SELECT * FROM kullanicilar LIMIT 10 OFFSET 15";
Aynı sonucu elde etmek için daha kısa bir sözdizimi de kullanabilirsiniz:

$sql = "SELECT * FROM kullanicilar LIMIT 15, 10;

PHP MYSQL KULLANIMI İÇİN ÖRNEK BİR UYGULAMA

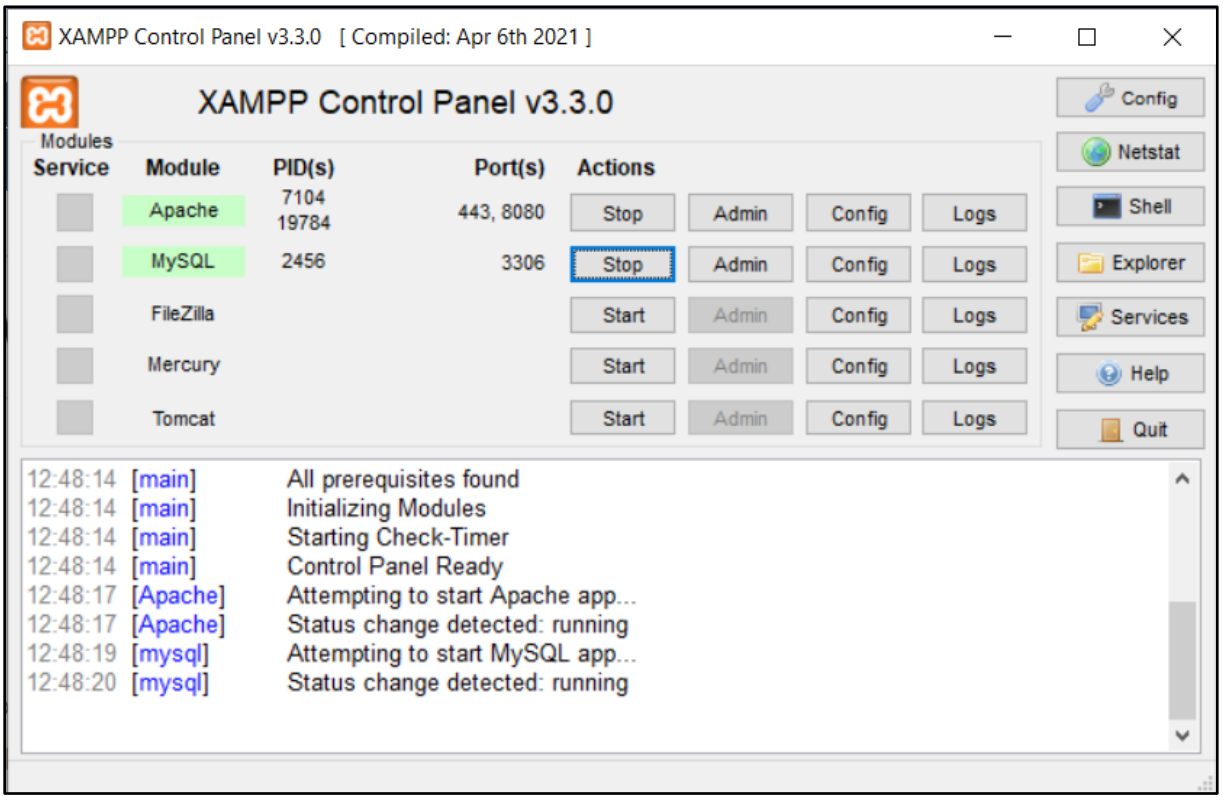
Bölümümüzün bu kısmında ekle/sil/güncelle ve veri çekme üzerine basit bir web uygulaması geliştireceğiz. Kayıt olarak da öğrenciye ait numara (ogrno), isim (ograd, ogrsoyad), kadın/erkek cinsiyet durumu (cinsiyet), doğum tarihi (dtarih), okuduğu sınıf (sinif) ve aldığı puan (puan) bilgilerini tutulmaktadır. Projemizin **bağlan.php, index.php, ekle.php, güncelle.php ve detay.php** olmak üzere beş sayfası bulunmaktadır. Proje için gerekli olan sql

ve php dosyaları sistem üzerinden indirilebilir. Aşağıda tablomuz için oluşturulan alanların isimleri, veri türleri, indeks bilgisi verilmektedir:

```
CREATE TABLE `ogrenci` (
  `ogrno` int(11) NOT NULL AUTO_INCREMENT,
  `ograd` varchar(30) NOT NULL,
  `ogrsoyad` varchar(30) NOT NULL,
  `cinsiyet` varchar(30) NOT NULL,
  `dtarih` date NOT NULL,
  `sinif` varchar(30) NOT NULL,
  `puan` int(11) NOT NULL
)
```

Görüldüğü üzere ogrno alanı indeks için kullanılmaktadır. Ayrıca otomatik artış değeri belirlendiği için bu alana veri girişi yapılmayacaktır. Her kayıt oluştuğunda ilgili yere 1'den başlayıp bir artarak öğrenci numarası verilmektedir.

Projenin çalıştırılması için sisteminize XAMPP paket programını yüklemeniz gerekmektedir (https://www.apachefriends.org/tr/). Program kurulduktan sonra ise XAMPP kontrol panelinden Apache ve MySQL modüllerinin başlatılması gerekir (Şekil 1).

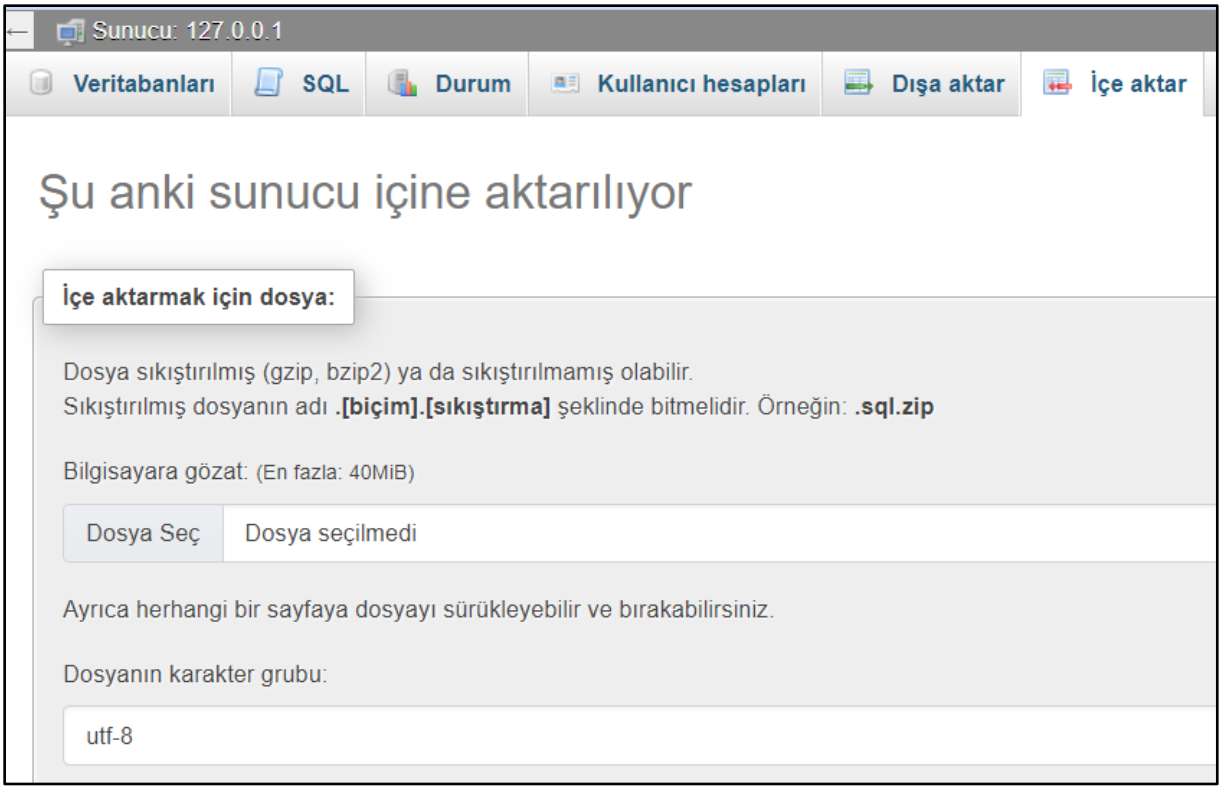


Şekil 1. XAMPP Kontrol Paneli

Bu işlemlerden sonra herhangi bir tarayıcı penceresinde localhost yazıp XAMPP'ın modüllerine bağlantı sağlayan pencereye (dashboard) geçilmelidir. Ardından menüden phpMyAdmin bağlantısı kullanılacaktır. Aşağıda phpMyAdmin'den kısmi bir ekran görüntüsü paylaşılmaktadır. İlgili yerde "İçe Aktar" ile ders sisteminizde yüklü bulunan "kutuphane.sql" dosyasını içe aktarmak gerekmektedir (Şekil 2). Proje için gerekli olan php dosyaları ise,

C:\xampp\htdocs

yolunda belirleyeceğiniz bir klasöre aktarılmalıdır. Burada oluşturulması gereken klasörü "proje" olarak kabul edelim.



Şekil 2. phpMyAdmin Ekran Görüntüsü

Projenizi başlatmak için tarayıcı penceresine "lolcahost/proje" yazmanız yeterlidir. Ana sayfanın (index.php) ekran görüntüsü aşağıdaki gibidir (Şekil 3):

Şekil 3. Ana Sayfa Ekran Görüntüsü

Ana sayfadaki Öğrenci Ekle düğmesi ile buraya daha önceden iki tane örnek kayıt getirilmiştir. Sizlerde bu kısım öğrenci eklenene kadar boş gözükecektir. Aşağıda öğrenci ekle işleminin (ekle.php) ekran görüntüsü verilmektedir (Şekil 4):

Şekil 4. Öğrenci Ekle İşlemi Ekran Görüntüsü

Ekle işlemi için bölümümüzün veri ekleme kısmında anlatılandan farklı olarak eklenecek veriler form öğesi içinde bulunan input öğelerinden alınmaktadır. Form öğesinden gönderilen veriler POST metodu ile aktarıldığından aşağıdaki gibi bir dizi değişken oluşturulmaktadır:

```
$dizi =[
        $_POST["ad"],
        $_POST["sad"],
        $_POST["cins"],
        $_POST["dtarih"],
        $_POST["sinif"]
    ];
```

Kayıt silme işlemi index.php dosyasında yer almaktadır:

```
<a href="?sil=<?=$satir['ogrno']?>" onclick="return confirm('Silinsin mi?')" class="btn btn-danger">Kaldır</a>
```

Burada silme işlemi sorgulu URL kodu ile a öğesinden gerçekleştirilmektedir. Kayıt silme (Şekil 5), listeleme (detay.php) (Şekil 6), güncelleme (guncelle.php) (Şekil 7) işlemlerinin ekran görüntüsü aşağıda sırasıyla verilmektedir:

Şekil 5. Öğrenci Silme İşlemi Ekran Görüntüsü

Şekil 6. Öğrenci Listeleme İşlemi Ekran Görüntüsü

Şekil 7. Öğrenci Güncelleme İşlemi Ekran Görüntüsü

Projedeki form öğelerinin daha etkili gösterilmesi için bootstrap.css dosyasından yararlanılmaktadır:

> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="anonymous">

İlgili satır form öğelerinin çağrıldığı sayfalarda <head></head> bloğu içerisinde yazılmaktadır. Bahsi geçen CSS dosyasındaki kodlar ise öğelerde class özniteliğinde değer atanarak kullanılmaktadır. Bağlantının oluşturulması/kapatılması ve tüm sorguların çalıştırılmasında PDO söz dizimi benimsenmektedir.

BÖLÜM ÖZETİ

My eskuÊL, ilişkisel veritabanı yönetim sistemleri arasında en popüler olanlardan biridir ve geniş kullanım alanlarına sahiptir. Veritabanı yönetimi, web uygulamaları, e-ticaret platformları, mobil uygulamalar ve veri analizi gibi çeşitli alanlarda yaygın olarak kullanılır. My eskuÊL, web sitelerinin ve uygulamalarının arkasında veri tabanı olarak kullanılarak, kullanıcı verilerinin saklanması, işlenmesi ve geri alınması için önemli bir rol oynar. PHP ise sunucu tarafı web geliştirme için özel olarak tasarlanmış güçlü bir programlama dilidir. Web sitelerinin dinamik ve etkileşimli özelliklerini geliştirmek için yaygın olarak kullanılır. Web tabanlı uygulamaların geliştirilmesi, veri tabanı işlemleri, form işlemleri, kullanıcı oturumları yönetimi ve sunucu tarafı programlama gibi birçok alanda PHP tercih edilir. Ayrıca, PHP'nin içerik yönetim sistemlerinde ve e-ticaret platformlarında kullanımı da oldukça yaygındır. My eskuÊL, verilerin saklanması, düzenlenmesi ve erişilmesi için kullanılır. eskuÊL dilini kullanarak veritabanı işlemleri gerçekleştirilir. Veri tabanı bağlantısı oluşturma, PHP gibi programlama dilleri kullanılarak My eskuÊL veritabanına bağlanmak için My eskuÊLi veya PDO gibi uzantılar kullanılır. Bağlantı bilgileri, (sunucu adı, kullanıcı adı, şifre vb.) belirtilerek bağlantı oluşturulur. Veri tabanında tablo oluşturma, veritabanında verilerin organize edildiği yapılardır. “CREATE TABLE” ifadesi kullanılarak yeni bir tablo oluşturulur. Tablonun adı ve sütunların adları ve veri türleri belirtilerek tablo oluşturulur. Veri tabanından istenilen verileri getirme, “SELECT” ifadesi kullanılarak belirli sütunlar veya tüm sütunlar seçilir. “WHERE” koşulu ile belirli bir koşulu karşılayan satırlar seçilebilir. Verileri sıralı bir şekilde getirme, “SELECT” ifadesinin sonuna “ORDER BY” ifadesi eklenerek veriler belirli bir sütuna göre sıralanabilir. “ASC” (artan) veya “DESC” (azalan) parametreleri kullanılarak sıralama yöntemi belirlenir. Veri ekleme, silme, güncelleme, “insert into” ifadesi kullanılarak yeni veri eklenir. “DELETE FROM” ifadesi kullanılarak belirli bir koşulu karşılayan veriler silinir. “UPDATE” ifadesi kullanılarak belirli bir koşulu karşılayan veriler güncellenir. Bu işlemler, veritabanındaki verilerin yönetilmesi ve güncellenmesi için kullanılır.

GÖZDEN GEÇİRELİM

1. PHP'de MySql söz dizim yöntemleri nelerdir?

2. Bağlantı cümlesinin conn isimli değişkende tutulduğu kabul edilsin. Bu durumda PDO söz diziminde ilgili bağlantı nasıl kapatılır?

3. bind_param() fonksiyonu aldığı parametreleri sql sorgusuna bağlar ve veritabanına parametrelerin ne olduğunu söyler. Söz konusu parametreler nelerdir?

4.

```
$stmt = $conn->prepare("INSERT INTO uye (ad, soyad, yas) VALUES (?, ?, ?)");
$stmt->bind_param("___", $firstname, $lastname, $email);
```

Yukarıdaki kodda insert deyimiyle hazırlanmış ifade oluşturulmuştur. Söz konusu uye tablosunun 3 adet string alanı bulunmaktadır. Bu durumda bind_param() fonksiyonunun alması gereken ilk parametre ne olmalıdır?

5. Bağlantı hatasını döndüren MySqli prosedürel söz dizimi nedir?

Cevaplar:

1. MySqli Nesne Yönelimli, MySqli Prosedürel, PDO

2. `$conn = null`

3. i-integer, d-double, s-string, b-BLOB

4. ssi

5.

```
$baglanti = mysqli_connect($servername, $username, $password, $dbname);
mysqli_error($baglanti)
```

KAYNAKÇA

[1] PHP ve MySQL, Pusula Yayıncılık, Erkan Balaban, İstanbul, 2010

[2] PHP ve AJAX, Seçkin Yayıncılık, Haydar Tuna, Ankara, 2010

FAYDALI KAYNAKLAR VE INTERNET KAYNAKLARI

https://www.w3schools.com/php/default.asp
https://www.php.net/manual/tr/book.mysql.php