

Mysql Veritabanı Dökümanı

Nerelerde Kullanıyoruz?

- Giriş-Çıkış Saatleri
- Arka Trafta Oluşan Data Analizi
- Log Kayıtları
- Yüksek Skor Menüsü
- Kullanıcı Giriş/Çıkış Menüsü

Kurulum:

1. Localhost Oluşturuyoruz.
 - Xampp,
 - WampServer vb.

Kurulum Sonrası Hatalar!

- Apache Server Çalışmayabilir(80. Port http Portudur.)
- Xampp Kapatırken Hata
- Dosya Adlarıyla .php oluşmuyor

Veritabanı İşlemleri

CRUD aşağıdaki gösterilen temel veritabanı işlemlerinin baş harflerinin bir araya gelmesinden oluşan bir ifadedir.

- C- Create(Oluşturma)
- R- Read(Okuma)
- U- Update(Güncelleme)
- D- Delete(Silme)

Create Deyimi

- **Veritabanı Oluşturmak**

Create Database veritabani ismi

Create Database veritabani ismi

```
DEFAULT CHARACTER SET utf8  
DEFAULT COLLATE utf8_general_ci;
```

Eklerin Anlamı:

1. **ci**: Case-Insensitive (Büyük – küçük harfe duyarsız)
2. **cs**: Case-Sensitive (Büyük – küçük harfe duyarlı. Sadece bazı diller için geçerli, türkçede kullanılmaz)
3. **bin**: Binary (Karakteri tanımlayan byte değerine göre karşılaştırmak demek. Pratikte büyük – küçük harfe duyarlılık istendiğinde kullanılır.

Örnek: Java büyük-küçük harf duyarlı olduğundan **sayi, Sayi** değişkenleri farklı olarak algılanır.

- **Tablo Oluşturmak**

Create Table tablo ismi

```
(  
    id int AUTO_INCREMENT,  
    kullanıcı_Adi varchar(25) NOT NULL,  
    şifre varchar(15),  
    PRIMARY KEY(id)  
)
```

- **Create View(Görünüm)** :Fiziksel olarak olmayan fakat çeşitli sorgular sonucunda elde edilen verilerin sanal bir tablo olarak gösterilmesini sağlayan yapılar. Bu yapıların avantajı; veriye erişimi daha rahat kılmak ve kullanıcının asıl tablolarda değişiklik yapmasını engellemek.
- **Create Index** : Tabloda bulunan belirli sütunlardaki değerlere hızlı erişim için kullanılan sıralama tekniği .Ayrıca indexin tekil (unique) özelliği verilerek sütundaki verinin tekrarlanmasını önler. Örneğin birincil anahtarlar (primary key) da aslında bir indexlerdir. Tanımlama işlemi yapılırken otomatik olarak indexlenirler.

Insert into Deyimi

Tablolarımıza ekleme yapmak için kullanılır

Temel prototip:

Insert into *tablo_adi* (Alanlar) **VALUES** (değerler)

Auto_increment set etme:

ALTER TABLE table_name AUTO_INCREMENT = start_value;

Select Deyimi

Select SQL dilinde bilgi çekmek/okumak için kullanılır.

Temel prototip:

SELECT * FROM *tablo_adi* (Açıklama:Tablo_adi adlı tablonun tüm sütunlarını döndürür.)

* işareti tüm sütunları seçmek için kullanılır.

SELECT * FROM *tablo_adi* **Where** *koşullar*

And,or

Update Deyimi

Veritabanında bulunan verileri değiştirmek-güncellemek amacıyla kullanılan bir deyimdir.

Temel prototip:

UPDATE [*tablo_adi*]

SET *sutun1= deger1, sutun2 = deger2, ...*

WHERE [*şartlar*]

Not : Update komutunu çalıştırırken dikkatli olmak gerekir. Kullanılan WHERE ifadesi hangi kayıtların güncelleneceğini belirler. WHERE ifadesi çıkarılırsa tablodaki tüm kayıtlar güncellenecektir.

Delete Deyimi

SQL Veritabanında kayıt silmek için **DELETE** ifadesi kullanılır.

Temel prototip:

Delete From [*tablo_adi*] **Where** [*şartlar*]

Yanlış Kullanım:~~Delete * from ogrenci where ogrno = 50~~

Where ile Farklı Sorgu Örnekleri

SELECT ograd,ogrsoyad,sinif

FROM ogrenci

WHERE cinsiyet='E' and sinif='11A'

And Örneği:Cinsiyeti Erkek ve Sınıfı 11A olan kişileri listeler.

SELECT * FROM ARACLAR

WHERE ARAC_MARKA = 'Mercedes' OR ARAC_MARKA = 'BMW'

OR Örneği:Arac markası Mercedes veya Bmw olan araçları listeler.

SELECT kitapad, sayfasayisi FROM kitap

WHERE sayfasayisi BETWEEN 100 AND 200 AND turno=2

Between Örneği:Sayfa sayısı 100 ile 200(sınırlar dahil) arasında olan ve tür numarası 2 olan kitapları listeler.

SELECT Name FROM MUSTERI

WHERE Name LIKE 'M%'

Like Örneği1:İsminin baş harfi M olan kişileri listeler.

SELECT *

FROM Personel

WHERE Bolum LIKE '%Yönetici%'

(*NOT LIKE Olsaydı geçmeyenler olurdu.)

Like Örneği2:Bolum alanının herhangi bir yerinde Yönetici geçenleri listeler.

Select ISIM, YAS FROM UYE

WHERE YAS IN(11,12,13)

IN Örneği1:Yaşı 18 ve 19 olan kişilerin isim ve yaş bilgilerini listeler.

SELECT * FROM employees

WHERE department_id IN

IN Örneği2:Alt sorguda adı ALEXANDER olan kişilerin departman idleri üst sorguya döner ve bu idlerin müşteriler tablosundaki bilgilerini listeler.

(SELECT department_id FROM employees WHERE first_name='Alexander');

SELECT * FROM personel

WHERE diplomanotu Is Null

(Is Not Null olsaydı null olmayan kayıtları)

Is Null Örneği:Personel tablosunda diploma notu null olan bütün kayıtları listeler.

Veritabanı Anahtar Kavramı ve Türleri Nelerdir?

- Veritabanında bir kayıt içerisinde farklılıkları ve nitelikleri gösteren belirleyicilere Anahtar denir.
- Anahtarlar tabloda bulunan her bir kaydın diğerinden farklı olmasını garanti altına alan bir alandır.

Anahtar Türleri:

- Primary Key (Birincil Anahtar-PK)
- Unique Key (Tekil Anahtar-UK)
- Foreign Key (Yabancı Anahtar-FK)
- Composite Key (Birleşik Anahtar-CK)

1.Primary Key

- Aynı değeri 2 kez içermeyecek olan sütun birincil anahtar olarak belirlenir.
- Bir tabloda yalnızca bir tane birincil anahtar içerebilir.
- Null değer içeremezler.
- Birden fazla kolonun birleşimiyle oluşabilirler.

2.Unique Key

- Bir tabloda tanımlanmış birden çok benzersiz anahtar olabilir.
- Bir veya daha fazla sütun benzersiz bir anahtar oluşturur.
- Sütun NULL olabilir, ancak sütun başına bir NULL değerine izin verilir.
- Birincil Anahtarlar aynı zamanda tekil anahtar sayılabilir fakat tekil anahtarlar birincil anahtar değildirler.

3.Foreign Key

- Tablodaki bir veriyi başka bir tablodaki veri ile ilişkilendirir.
- İlişkilendirilecek olan tablonun Primary key alanı ile diğer tablonun Foreign key alanı birbiri ile bağlanır.

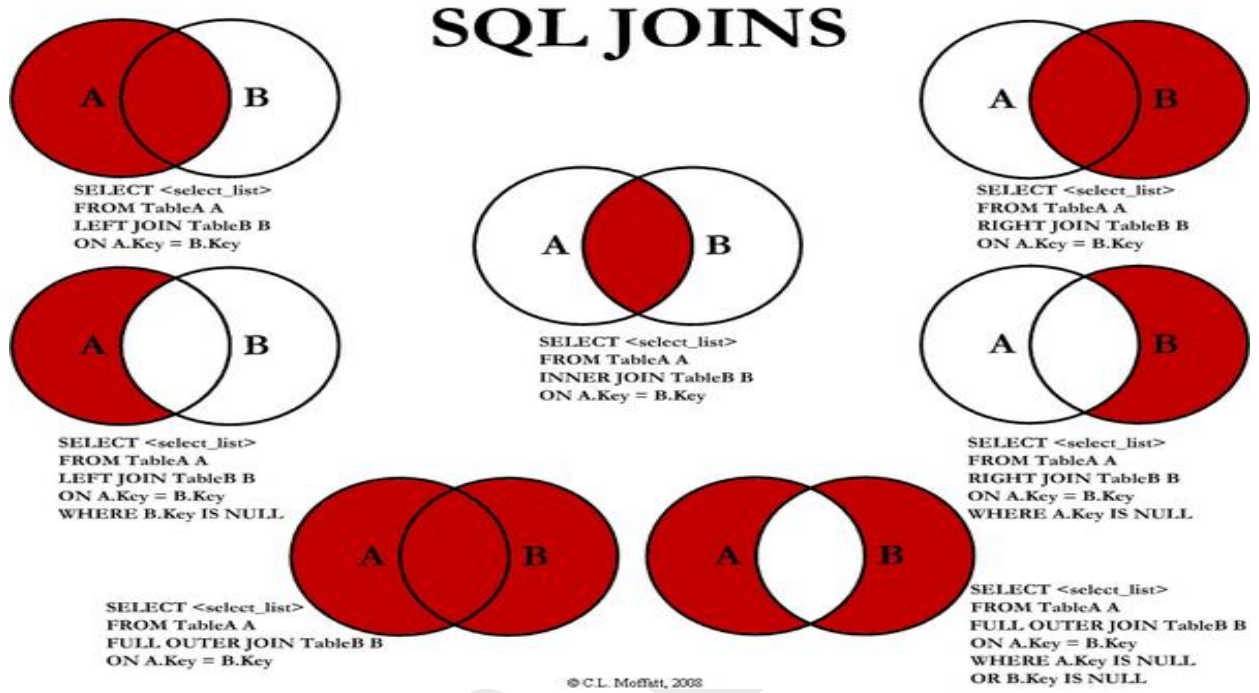
4.Composite Key

- İki yada daha fazla olan birincil anahtarlar tek başına bir birincil anahtar gibi davranır.
- Bazı durumlarda sadece bir Primary key sorunu çözmeye yetmediği durumlarda kullanılır.

JOIN Nedir? Türleri Nelerdir?

JOIN iki veya daha fazla tabloyu birleştirmek için kullanılır.

Türleri:






























- **Inner Join:**
İki tablodaki eşleşen kayıtlar için kullanılır.
- **Left Join:**
İki tablodaki eşleşen kayıtlar ve eşleşmeyen sol kayıtlar için kullanılır.
- **Right Join:**
İki tablodaki eşleşen kayıtlar ve eşleşmeyen sağ kayıtlar için kullanılır.
- **Full (Outer) Join:**
İki tablodaki eşleşen kayıtlar ve eşleşmeyen sol ve sağ kayıtlar için kullanılır. LEFT ve RIGHT JOIN birleşimidir.

Temel prototip:

```
SELECT tablo_adi.sutun_adi, ...  
FROM tablo_A  
{INNER JOIN | LEFT JOIN | RIGHT JOIN} tablo_B  
ON tablo_A.sutun_adi = tablo_B.sutun_adi;
```

+ Seçenekler

		kullanici_ID	nick	elmasMiktar	altinMiktar	toplamOlmeSayisi	toplamOldurmeSayisi	sonGirisTarihi
<input type="checkbox"/>	  	1	shai	125	50	123	254	2020-07-15
<input type="checkbox"/>	  	2	pokemon	75	456	178	562	2020-06-17
<input type="checkbox"/>	  	3	baki	486	455	750	832	2020-06-09
<input type="checkbox"/>	  	4	onePiece	411	253	147	632	2020-07-02
<input type="checkbox"/>	  	5	deathNote	522	530	98	452	2020-07-09
<input type="checkbox"/>	  	6	yujiro	974	846	333	999	2020-07-11
<input type="checkbox"/>	  	7	takao	446	521	145	745	2020-07-11
<input type="checkbox"/>	  	8	takao2	446	521	145	745	2020-07-11
<input type="checkbox"/>	  	9	takao2	0	521	145	745	2020-07-11



```
SELECT k.*,i.*,o.* from kullanicilar  
k  
INNER JOIN istatistikler i On  
k.kullanici_ID=i.kullanici_ID  
Inner JOIN oyunparalari o On  
k.kullanici_ID=o.kullanici_ID
```

```
SELECT k.nick,i.kafadanVurmaYuzdesi  
from kullanicilar as k  
INNER JOIN istatistikler as i On  
k.kullanici_ID=i.kullanici_ID  
Inner JOIN oyunparalari O on  
O.kullanici_ID=k.kullanici_ID
```

Veritabanı Fonksiyonlar

- LENGTH(Sütun adi): String değerli sütunlara uygulanır.
Sütundaki değerlerin **uzunluklarını** döndürür.
- Max(Sütun adi):
Sütunun **en büyük** değerini döndürür.
MAX(LENGTH(nick)): String ifadelerin **en uzun** olanını döndürür.
- Min(Sütun adi):
Sütunun **en küçük** değerini döndürür.
- Count(Sütun adi):
O sütundaki **null** olmayan kayıt sayısı.
Count(*):Tüm Satır Sayısını döndürür **null** olanları saymaz.
- Sum(Sütun adi):
Sütundaki değerler toplamı
- Avg(Sütun adi):
Sütundaki değerlerin ortalaması
- UCASE(Sütun adi):
Sütundaki değerleri büyük harfe ile gösterir.
- LCASE(Sütun adi):
Sütundaki değerleri küçük harf ile gösterir.

Ekstra***

Order By:

Sonuç kümesini **sıralamak** için, ORDER BY yan tümcesini kullanılır. Where ifadesinden sonra kullanılır.

ORDER BY yan tümcesi bize aşağıdaki olanakları sunar.

- Tek bir sütuna veya birden çok sütuna göre ayarlanmış bir sonucu sıralama.
- Bir sonucu farklı sütunlara göre artan veya azalan düzende sıralama.

ASC küçükten büyüğe(artan), **DESC büyükten küçüğe(düşen)** anlamına gelir. Varsayılan **ASC**

Temel prototip:

```
SELECT sutun1, sutun2,... FROM  
tablo_adi ORDER BY sutun1 [ASC|DESC], sutun2 [ASC|DESC],...
```

Limit:

Mysql de verilere sorgular uyguladığımızda bize geri gelen verileri LIMIT komutu ile kontrol edebilir. **İlk 5 kayıt**, **son 5 kayıt**, **50. kayıttan sonra 10 kayıt** gibi işlemleri uygulaya biliyoruz.

Prototip:

Sorgu sonuna **Limit 5** veya **Limit 5,5**

Limit 5 sorgu sonucuna göre 5 kayıt. Limit 5,5 sorgu sonucunun 5. Kayıdından itibaren 5 kayıt döndürür.

Group By: Bu ifade sutun içerisinde bulunan ifadeleri gruplamak için kullanılır. **örnek:** şehirlerden kaçar kişinin oynadığını görmek istiyoruz.

- **Having:** Sadece Group by ile kullanılır. Having ifadesi ile belirtilen kriter ise group by uygulandıktan sonra ortaya çıkan verileri filtrelemek için kullanılır. Ayrıca Where ifadesinden sonra sum, avg gibi fonksiyonlar kullanılamazken, Having ile kullanılabilir.