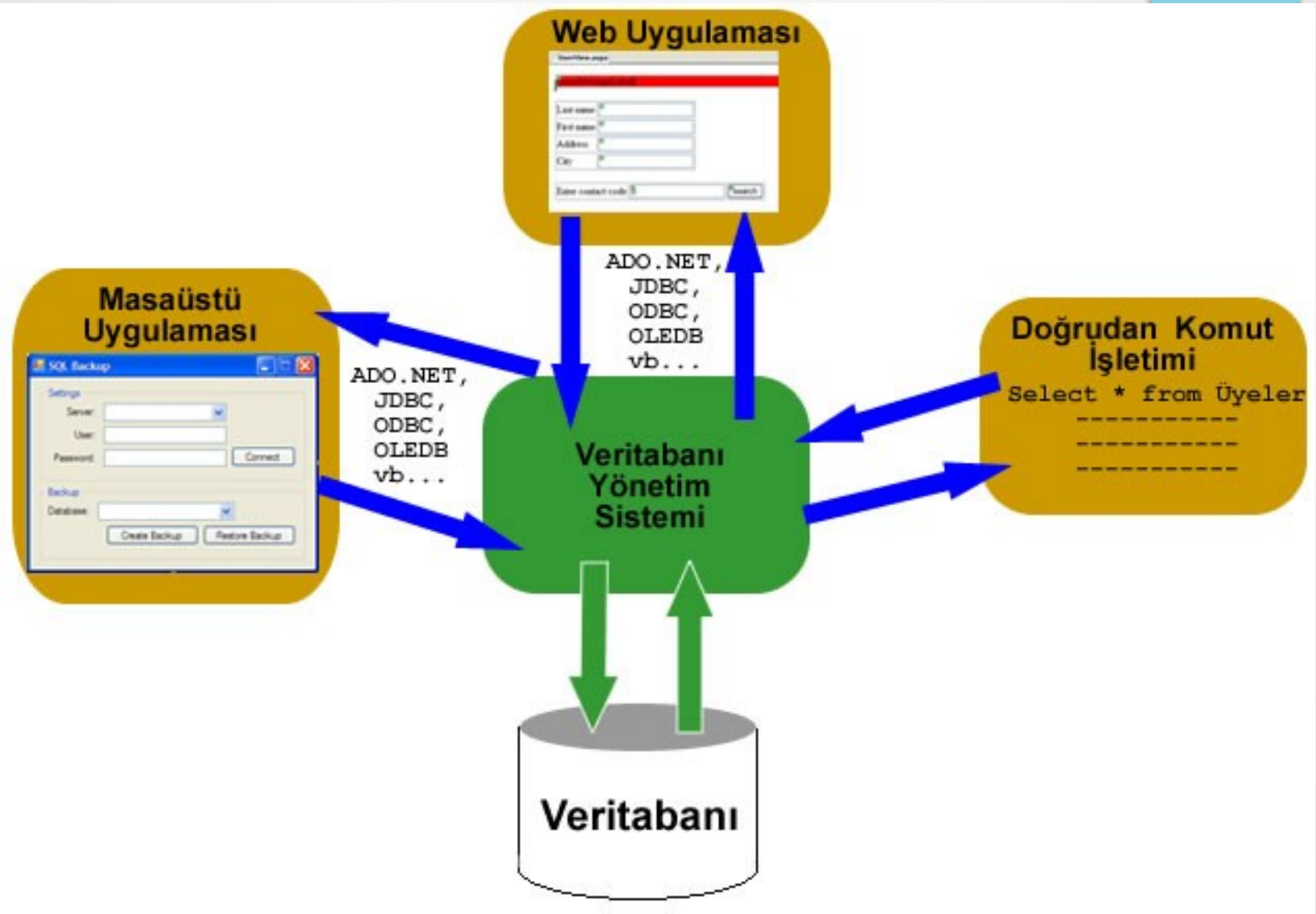
Mysql Veritabanı Komutları

Mysql Veri Tabanı Komutları

Hazırlayan : M. Başar ACAROĞLU

Kaynaklar:
http://www.hrzafer.com/sql-dersleri
http://www.w3schools.com/sql/default.asp

Veri Tabanı Nedir ?



Mysql Veritabanı oluşturma

```
CREATE DATABASE vtismi;
```

Mysql Veritabanı kullanma

```
USE dbAdi;
```

Mysql Veritabanı Tablo Oluşturma

```
CREATE TABLE tablo_adi
(
alan_adi1 veri_tipi(boyut),
alan_adi2 veri_tipi(boyut),
alan_adi3 veri_tipi(boyut),....
);
```

Örnek :

```
CREATE TABLE Kisiler
(
KisiID int,
Soyad varchar(255),
Ad varchar(255),
Adres varchar(255),
Sehir varchar(255)
);
```

Mysql Veri Türleri

Text Veri Tipleri:

- CHAR( ) : Sabit 0 – 255 karakter.(adi char(5)–> diye bir tanımla yapıp bu alana 2 karakterlik bir veri girsenizde 5 byte alan ayrılır)
- VARCHAR( ) : Değişken 0 – 255 karakter.(adi varchar(5)–> diye bir tanımla yapıp bu alana 2 karakterlik bir veri girseniz, 2 byte alan ayrılır)
- TINYTEXT : En Fazla 255 karakter.
- TEXT : En Fazla 65.535 karakter.
- BLOB : En Fazla 65.535 karakter.
- MEDIUMTEXT : En Fazla 16.777.215 karakter.
- MEDIUMBLOB : En Fazla 16.777.215 karakter.
- LONGTEXT : En Fazla 4.294.967.295 karakter.
- LONGBLOB : En Fazla 4.294.967.295 karakter.

Mysql Veri Türleri

Numeric Veri Tipleri:

- TINYINT( ) : -128 ,127 yada 0-255 UNSIGNED.
- SMALLINT( ) : -32.768 ,32.767 yada 0 “ 65.535 UNSIGNED.
- MEDIUMINT( ) : -8.388.608 , 8.388.607 yada 0 “ 16.777.215 UNSIGNED.
- INT( ) : -2.147.483.648 , 2.147.483.647 yada 0 “ 4.294.967.295
-
- BIGINT( ) : -9.223.372.036.854.775.808 , 9.223.372.036.854.775.807 yada 0 “ 18.446.744.073.709.551.615
- FLOAT : KüÇük Noktalı sayı.
- DOUBLE(,) : Büyük Noktalı sayı.
- DECIMAL(,) : DOUBLE tipte string şeklinde saklanır.

Mysql Veri Türleri

Tarih ve Zaman Veri Tipleri:

- DATE : YYYY-MM-DD
- DATETIME : YYYY-MM-DD HH:MM:SS
- TIMESTAMP : YYYYMMDDHHMMSS
- TIME : HH:MM:SS

Diğer Veri Tipleri:

- ENUM ( ) : Kullanıcı tanımlı liste tipi. Ür; ENUM(“E”,”H”)
- SET : Küme Tipi. ENUM benzeri. Aynı anda birden fazla kayıt tutabilir.

Constraint (Kısıtlayıcı)

Veri üzerindeki mantıksal sınırlamalara kısıt adı verilir. Kısıtların genel olması tercih edilen bir durumdur. Kısıtlar, veri modellerinde bütünlük sağlamak için kullanılır. Kısıtlamalar, tabloların tanımlanmasıyla beraber oluşan öğelerdir. Kısıtlamalar ile Rule (kural) ve Default'ların (varsayılan) yapabileceği işler yapılabilir. Constraintler tablo oluştururken yani CREATE TABLE komutuyla tanımlanabilir. Tablo oluşturulmuşsa ALTER TABLE komutuyla bu işlem gerçekleşir. ALTER TABLE komutuyla kullanıldığında sütunlara girilen bilgilerin dikkate alınması gerekir.

Constraint (Kısıtlayıcı)

- NOT NULL – Alana boş değer girilemeyeceğini belirtir
- UNIQUE – Tekil alan kısıtlayıcı anlamındadır. Birincil anahtar olan ve tablodaki diğer alanlar içinde aynı içeriğe sahip verilerin olmaması için Unique Constraint tanımlanır. T.C.Kimlik Nu. primary key ve Okul Nu. Unique şeklinde bir tanımlama Unique Constraint'e bir örnektir.
- PRIMARY KEY - Birincil anahtar kısıtlayıcı anlamındadır. Aynı olmayan değerler girilmesini sağlar. Bu da her kaydın farklı olması demektir. Her tablonun en fazla 1 adet Primary Key Constraint'i olabilir.
- FOREIGN KEY - Yabancıl anahtar kısıtlayıcı anlamındadır. Bir tablodaki bir sütuna ait verilerin başka bir tablonun belirli bir sütunundan gelmesini denetler.
- CHECK - Kontrol kısıtlayıcı anlamındadır. Belirtilen formata göre verilerin girilmesini sağlar. Örneğin, T.C.Kimlik Nu. alanına 11 karakterin girilmesi Check Constraint ile sağlanabilir.
- DEFAULT - Varsayılan kısıtlayıcı anlamındadır. Tablodaki herhangi bir alan için girilmesi gereken bir değerin atanmasıdır. INSERT komutu için geçerlidir. Örneğin, kişi bilgilerinin alındığı bir tabloda kişinin uyruğunun girilmesi işleminde varsayılan değer olarak "T.C." atanabilir.

MYSQL NOT NULL

```
CREATE TABLE KisilerNotNull
(
K_Id int NOT NULL,
Soyad varchar(255) NOT NULL,
Ad varchar(255),
Adres varchar(255),
Sehir varchar(255)
)
```

MYSQL UNIQUE

```
CREATE TABLE Kisiler
(
K_Id int NOT NULL,
Soyad varchar(255) NOT NULL,
Ad varchar(255),
Adres varchar(255),
Sehir varchar(255),
UNIQUE (K_Id)
)
```

MYSQL PRIMARY KEY

```
CREATE TABLE Kisiler
(
K_Id int NOT NULL,
Soyad varchar(255) NOT NULL,
Ad varchar(255),
Adres varchar(255),
Sehir varchar(255),
PRIMARY KEY (K_Id)
)
```

MYSQL FOREIGN KEY

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

```
CREATE TABLE Siparisler
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
K_Id int,
PRIMARY KEY (O_Id),
FOREIGN KEY (K_Id) REFERENCES Kisis(K_Id)
)
```

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	2
4	24562	1

MYSQL CHECK

```
CREATE TABLE Kisiler
(
K_Id int NOT NULL,
Soyad varchar(255) NOT NULL,
Ad varchar(255),
Adres varchar(255),
Sehir varchar(255),
CHECK (K_Id>0)
)
```

```
CREATE TABLE Kisiler
(
K_Id int NOT NULL,
Soyad varchar(255) NOT NULL,
Ad varchar(255),
Adres varchar(255),
Sehir varchar(255),
CONSTRAINT kont_Kisi CHECK (K_Id>0 AND Sehir='Eskişehir')
)
```

MYSQL DEFAULT

```
CREATE TABLE Kisiler
(
K_Id int NOT NULL,
Soyad varchar(255) NOT NULL,
Ad varchar(255),
Adres varchar(255),
Sehir varchar(255) DEFAULT 'İZMİR'
)

CREATE TABLE Siparisler
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
K_Id int,
OrderDate date DEFAULT GETDATE()
)
```

MYSQL AUTO INCREMENT ALAN

```
CREATE TABLE Kisiler
(
ID int NOT NULL AUTO_INCREMENT,
Soyad varchar(255) NOT NULL,
Ad varchar(255),
Adres varchar(255),
Sehir varchar(255),
PRIMARY KEY (ID)
)
```

MYSQL DROP INDEX, DROP TABLE, and DROP DATABASE

DROP TABLE tablo_adi

DROP DATABASE database_Adi

TRUNCATE TABLE tablo_adi

MYSQL ALTER TABLE

ALTER TABLE komutu bir tabloda değişiklik yapmak için kullanılır.

ALTER TABLE eskiad RENAME TO yeniad

ALTER TABLE ogrenciler ADD babaismi varchar(50);

ALTER TABLE Uyeler ADD (Yer VARCHAR(50), Yas INT, Uye_ID Counter, Kayit_Tarihi DATETIME, Profil MEMO);

ALTER TABLE ogrenciler DROP COLUMN kimyanotu ;

ALTER TABLE Uyeler MODIFY Yer VARCHAR(100);

MYSQL INSERT INTO

INSERT INTO tablo_adi VALUES (deger1,deger2,deger3,...);

INSERT INTO tablo_adi (kolon1,kolon2,kolon3,...)
VALUES (deger1,deger2,deger3,...);

MYSQL UPDATE

```
UPDATE tablo_adi
SET kolon1=deger1,kolon2=deger2,...
WHERE kolon=deger;
```

MYSQL DELETE

```
DELETE FROM tablo_adi
WHERE kolon=deger;
```

MYSQL SELECT

```
SELECT kolon_adi,kolon_adi
FROM tablo_adi;
```

```
SELECT * FROM tablo_adi;
```

MYSQL SELECT ALIASES

```
SELECT kolon_adi AS alias_Adi
FROM tablo_adi;


SELECT MusteriAdi AS Musteri, ilgiliAdi AS [ilgili Kisi]
FROM Musteriler;
SELECT MusteriAdi, Adres+', '+Sehir+', '+PostaKodu+',
'+Ulke AS Adres
FROM Musteriler;
```

MYSQL SELECT ALIASES

Tablolara Alias Atama

```
SELECT alias_adi.kolon_adi

FROM tablo_adi alias_adi;
```

```
SELECT u.isim , u.soyisim FROM uyeler u WHERE
u.cinsiyet='K' AND u.isim='Derya'
```

MYSQL SELECT DISTINCT

```
SELECT distinct kolon_adi,kolon_adi
FROM tablo_adi;
```

```
SELECT distinct kolon_adi,* FROM tablo_adi;
```

MYSQL SELECT WHERE

```
SELECT kolon_adi,kolon_adi
FROM tablo_adi WHERE
kolon_adi operator deger;
```

MYSQL SELECT WHERE

Operatör	Açıklama
=	eşit
<>	Eşit değil
>	büyüktür
<	küçüktür
>=	Büyük eşit
<=	Küçük eşit
between	Belitilen değerler arasında
like	Özel arama
in	Belitilen birden fazla değer için

MYSQL SELECT WHERE

Şartları bağlamak için or/and kullanılır.

```
SELECT * FROM Musteriler
WHERE Ulke='Germany'
AND Sehir='Berlin';
```

```
SELECT * FROM Musteriler
WHERE Ulke='Germany'
AND (Sehir='Berlin' OR Sehir='München');
```

MYSQL SELECT WHERE LIKE

```
SELECT kolon_adi(lar)
FROM tablo_adi
WHERE kolon_adi LIKE pattern;
```

MYSQL SELECT WHERE LIKE

Karakterler	Açıklama
%	Karakter dizisi
_	Tek bir karakter
[karakterlistesi]	Eşleşecek karakter aralığı
[^karakterlistesi] veya [! karakterlistesi]	Eşleşmeyecek karakter aralığı

```
SELECT * FROM Musteriler WHERE Sehir LIKE 'ber%';

SELECT * FROM Musteriler WHERE Sehir LIKE '%es%';

SELECT * FROM Musteriler WHERE Sehir LIKE '_erlin';

SELECT * FROM Musteriler WHERE Sehir LIKE 'L_n_on';

SELECT * FROM Musteriler WHERE Sehir LIKE '[bsp]%';

SELECT * FROM Musteriler WHERE Sehir LIKE '[a-c]%';

SELECT * FROM Musteriler WHERE Sehir LIKE '[!bsp]%';

SELECT * FROM Musteriler WHERE Sehir NOT LIKE '[bsp]%';
```

MYSQL SELECT WHERE IN

```
SELECT kolon_adi(lar)
FROM tablo_adi
WHERE kolon_adi IN (deger1,deger2,...);
```

```
SELECT * FROM Musteriler
WHERE Sehir IN ('Paris','London');
```

MYSQL SELECT WHERE BETWEEN

```
SELECT kolon_adi(lar)
FROM tablo_adi
WHERE kolon_adi BETWEEN deger1 AND deger2;


SELECT * FROM Urunler WHERE Fiyat BETWEEN 10 AND 20;
SELECT * FROM Urunler WHERE Fiyat NOT BETWEEN 10 AND 20;
SELECT * FROM Urunler WHERE (Fiyat BETWEEN 10 AND 20)
AND NOT KategoriID IN (1,2,3);
SELECT * FROM Siparisler WHERE OrderDate BETWEEN #07/04/1996#
AND #07/09/1996#;
```

MYSQL SELECT ORDER BY

```
SELECT kolon_adi,kolon_adi
FROM tablo_adi
ORDER BY kolon_adi,kolon_adi ASC|DESC;
```

```
SELECT * FROM Musteriler
ORDER BY Ulke;
```

MYSQL LIMIT

SELECT sorgusu ile dönen kayıtların belli bir sayıda olmasını istememiz halinde LIMIT anahtar sözcüğünü kullanırız. Özellikle binlerce kayıt bulunan çok geniş tablolarda bu komut performans açısından oldukça faydalı olabilir.

Select * from uyeler where 1 limit 10;

Select * from uyeler where 1 limit 5,5;

MYSQL KÜMELEME FONKSİYONLARI

AVG() Fonksiyonu

Ortalama hesaplar. Örneğin çalışanların ortalama ne kadar maaş aldığını hesaplamak istersek:

SELECT AVG(maaş) FROM kisiler

MYSQL KÜMELEME FONKSİYONLARI

COUNT() Fonksiyonu

Bir alandaki değerlerin kaç adet olduğunu yani sayısını hesaplar. Örneğin kaç çalışanın olduğunu hesaplamak istersek:

```
SELECT COUNT(maaş) AS Sayı FROM kisiler
```

Eğer bir kolondaki farklı değerlerin sayısını öğrenmek istiyorsak COUNT ifadesini COUNT (DISTINCT kolon_adı) şeklinde kullanırız. Örneğin kaç farklı ülkeden çalışan olduğunu hesaplamak istersek:

```
SELECT COUNT(DISTINCT ülke) AS ÜlkeSayısı FROM kisiler
```

MYSQL KÜMELEME FONKSİYONLARI

MAX() ve MIN() Fonksiyonları

Bir alandaki en büyük ve en küçük değerleri döndürürler. En yaşlı ve en genç çalışanları belirlemek istersek:

```
SELECT MAX(yas) AS EnYaşlı FROM kisiler
SELECT MIN(yas) AS EnGenç FROM kisiler
```

MYSQL KÜMELEME FONKSİYONLARI

SUM() Fonksiyonu

Sayısal değerler içeren bir kolondaki değerlerin toplamını döndürür. Örneğin aylık ödenen toplam maaşı öğrenmek istersek:

```
SELECT SUM(maaş) AS ToplamMaaş FROM kisiler
```

MYSQL SELECT GROUP BY

```
SELECT kolon_adi
FROM tablo_adi
GROUP BY kolon_adi;
```

```
SELECT ülke FROM `musteriler` GROUP BY ülke;
```

MYSQL HAVING

HAVING ifadesi GROUP BY ifadesi ile birlikte kullanılan bir ifade. Aslında HAVING ifadesinin işlevi WHERE ifadesininkine çok benziyor. Ancak kümeleme fonksiyonları ile WHERE ifadesi birlikte kullanılamadığından HAVING ifadesine ihtiyaç duyulmuştur.

SELECT ülke, AVG(maaş) FROM `kisiler` GROUP BY ülke HAVING AVG(yas) > 30

MYSQL BİRDEN FAZLA TABLO İLE ÇALIŞMAK

Yayınevleri

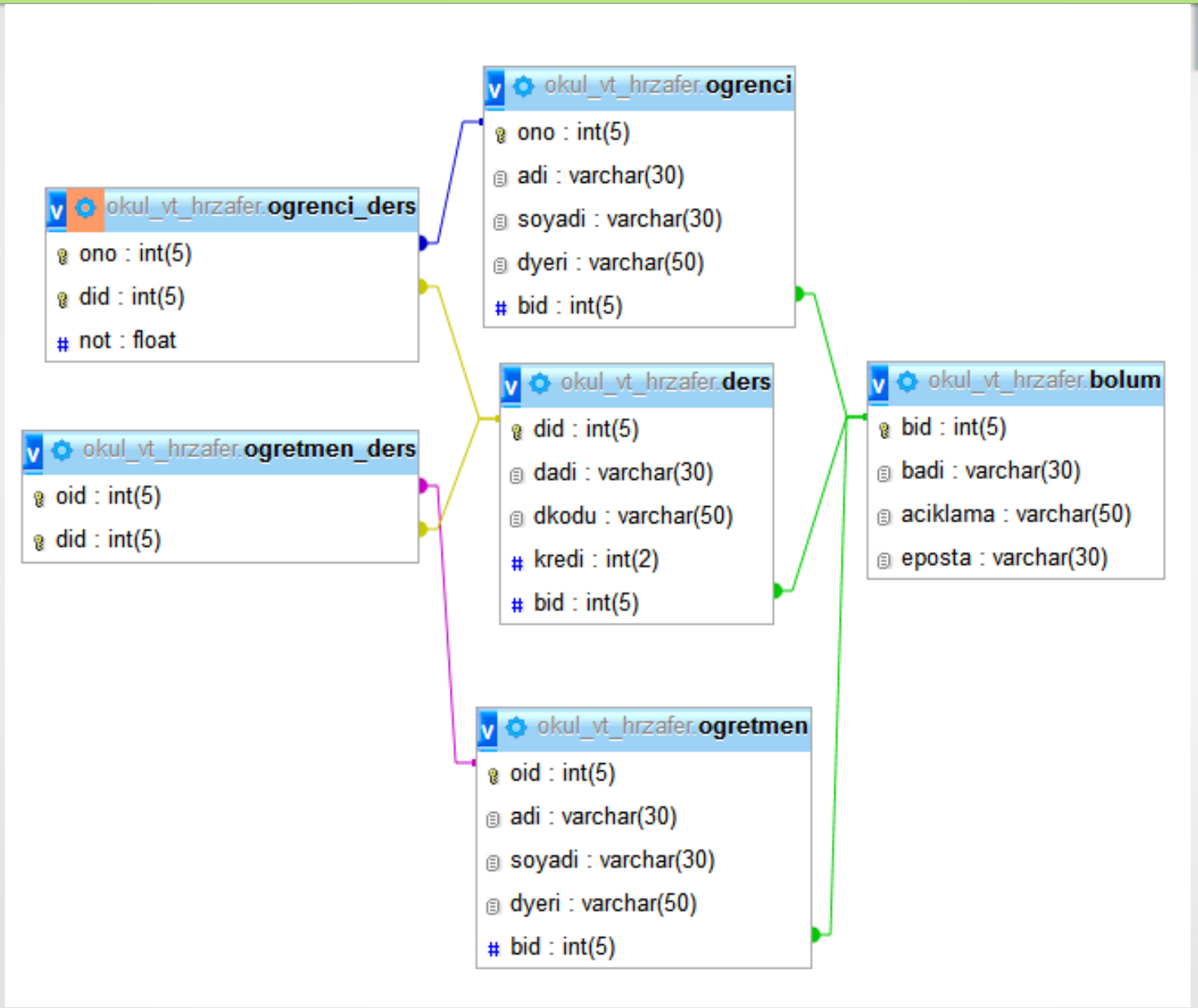
no	isim	sehir
1	A	Ankara
2	B	İstanbul
3	C	İzmir

Kitaplar

no	isim	yv_no
1	5 dakkada Java	1
2	5 bilemedin 6 saatte SQL	1
3	Hakiki SQL	2

```
SELECT * FROM kitaplar , yayinevleri
SELECT * FROM kitaplar, yayinevleri WHERE kitaplar.yv_no = yayinevleri.no
SELECT kitaplar.isim, yayinevleri.isim FROM kitaplar, yayinevleri WHERE kitaplar.yv_no = yayinevleri.no
SELECT K.isim, Y.isim FROM kitaplar K, yayinevleri Y WHERE K.yv_no = Y.no
```

MYSQL ÖRNEK VERİ TABANI

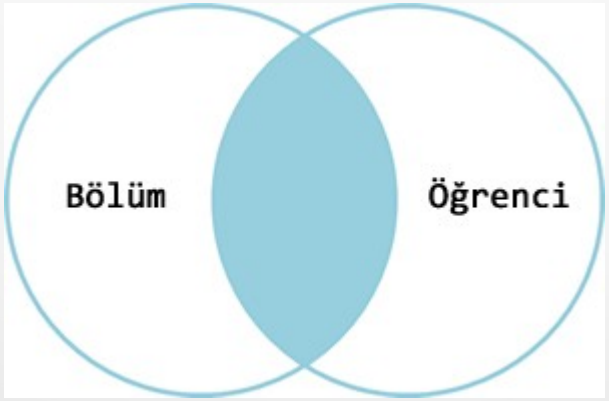


MYSQL JOIN TÜRLERİ

Inner Join

Inner join en çok kullanılan join türüdür ve her iki tablodaki ortak kayıtları döndürür. Bir başka ifade ile iki tablonun kesişimini döndürür. Mesela bölüm ve öğrenci tablolarını birleştirmek istersek

SELECT * FROM bolum b INNER JOIN ogrenci o ON b.bid = o.bid

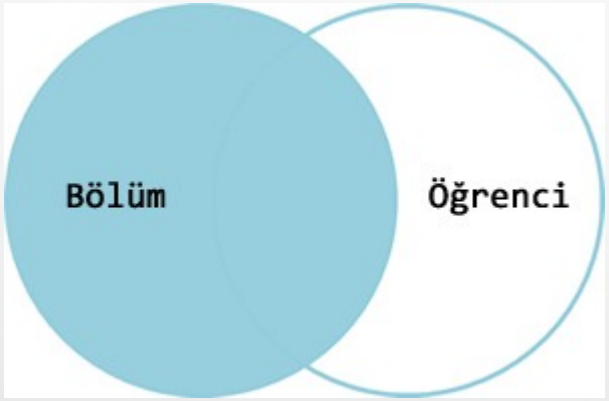


MYSQL JOIN TÜRLERİ

Left join

Eğer bir tablodaki tüm kayıtlar ile diğer tablodaki birleştirme koşulunu sağlayan kayıtları döndürmek istersek left join kullanırız. Mesela tüm bölümleri (hiç öğrencisi olmayanlar da dahil) ve bir bölüme kayıtlı öğrencileri sorgulamak istersek:

SELECT * FROM bolum b LEFT JOIN ogrenci o ON b.bid = o.bid



MYSQL JOIN TÜRLERİ

Right join

sorguyuda left (sol) yerine right (sağ) sözcüğünü kullanmış olsaydık, tüm öğrenciler ile en az bir öğrencisi olan bölümler seçilirdi. Yani left ve right tüm kayıtların seçileceği tablonun sağdaki mi yoksa soldaki mi olacağını belirtiyor yalnzıca.

SELECT * FROM bolum b RIGHT JOIN ogrenci o ON b.bid = o.bid

MYSQL JOIN TÜRLERİ

Full Outer Join

Tüm öğrencileri ve tüm bölümleri seçmek istersek full outer join kullanırız.

SELECT * FROM bolum b FULL OUTER JOIN ogrenci o ON b.bid = o.bid