SQL – Yapısal Sorgulama Dili

SQL – Yapısal Sorgulama Dili

- IBM tarafından sql standartları tanımlanmış ardından ISO ve ANSI tarafından da bir standart olarak kabule edilmiştir.
- SQL bir programlama dili değildir.
- Program geliştirme aşamasında SQL tek başına yeteli olmadığı için ayrıca **programlama dillerinden faydalanılır.**
- SQL komutları kullanarak veri tabanı, tablo oluşturma, kayıt ekleme, kayıt güncelleme, silme ve listeleme işlemleri yapılır.

SQL – Yapısal Sorgulama Dili

- Standart SQL ifadelerinde fonksiyon, döngü ve karşılaştırma gibi programlamaya yönelik ifadeler kullanılamamaktadır. **Bu nedenle PL/SQL ve T-SQL sorgulama dilleri geliştirilmiştir.**

PL/SQL: Oracle tarafından geliştirilen ve ORACLE veritabanı sistemlerine özel dildir. **Temel SQL ifadelerinin yanında akış kontrolleri ve değişken kullanımına imkan sağlar.**

T-SQL: Microsoft ve Sybase tarafından geliştirilmiştir. **Temel SQL ifadelerinin yanında akış kontrolleri ve değişken kullanımına imkan sağlar.**

SQL – Yapısal Sorgulama Dili

SQL ifadeleri yapısal olara 3 gruba ayrılır.

1) **DDL** (Data Definition Language – Veri tanımlama dili)
2) **DML** (Data Manipulation language – Veri işleme dili)
3) **DCL** (Data Control Language – Veri kontrol dili)

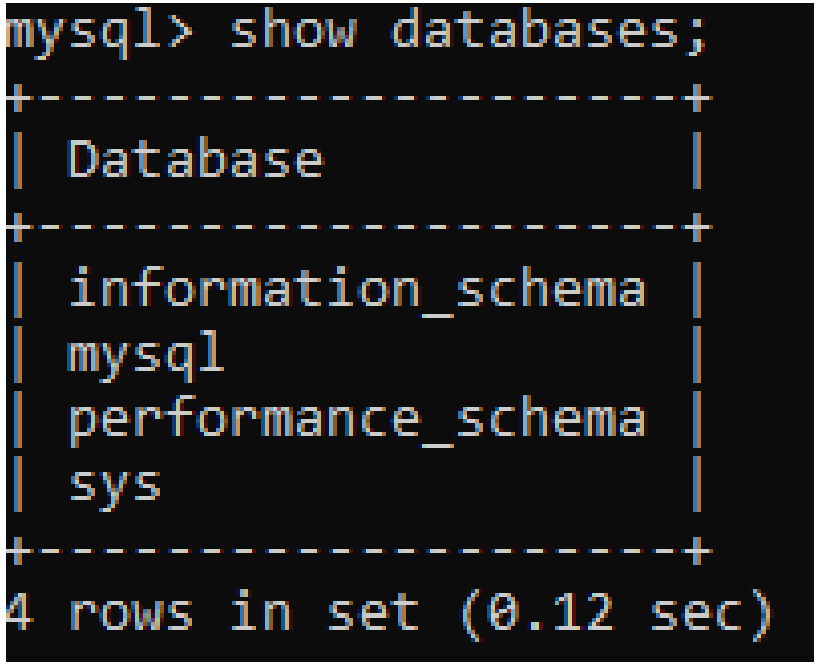
1. DDL (Data Definition Language – Veri tanımlama dili)

Veri Tanımlama Dili (DDL), 3 kategoriye ayrılır.

A. Create
B. Alert
C. Drop
D. Truncate

VAROLAN VERİTABANLARINI LİSTELEME

- MYSQL Server, komut satırı (CLI) veya grafik ara yüzle (GUI-MYQL Workbench, Phpmyadmin vb.) yönetilmektedir.
- Bu bölümde komut satırı kullanılarak işlemler yapılmıştır. MYSQL server yönetici olarak giriş yaptıktan sonra varolan veritabanları görmek için ***show databases*** komutu kullanılır.



A. CREATE

- CREATE, veri tabanı nesnelerini (tablo, index vs.) yaratmada kullanılan komuttur.
- SQL komutları ile veri tabanında işlem yapılabilmesi için önce

veri tabanı sonra da veri tabanında kullanılacak **tablolar** tanımlanmalıdır.

Veritabanı Oluşturma

Kullanımı: CREATE DATABASE veritabani_adi;

Örnek:

```
Create Database kutuphane;
Create Database deneme;
Create Database okul;
Create Database sirket;
```

VERİTABANI OLUŞTURMA (ÖRNEK)

- Örnek olarak aşağıda MYSQL server'da KUTUPHANE adında bir veritabanı oluşturulmuştur.

```
mysql> CREATE DATABASE KUTUPHANE;
Query OK, 1 row affected (0.06 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| kutuphane          |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

VERİTABANINI AKTİF YAPMAK

- Yeni oluşturulan veritabanı üzerinde işlem yapılabilmesi için varolan veritabanları arasından seçilmesi gerekir. Bunun için USE komutu kullanılır.

- **Kullanımı:** USE veritabani_adi;

Örnek : **USE kütüphane**;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| deneme             |
| information_schema |
| kutuphane          |
| mysql              |
| okul               |
| performance_schema |
| sirket             |
| sys                |
+--------------------+
8 rows in set (0.00 sec)
```

TABLOLARI LİSTELEMEK

- Bir veri tabanı içerisinde çok sayıda veri tabanı kütüğü ya da tablo bulunabilir. Mevcut tabloları görmek için yine SHOW komutundan faydalanılır.
- **Kullanımı:** Show tables;

Örneğin aşağıdaki şekilde Kutuphane veritabanında var olan veritabanları listelenmiştir. Bu veritabanında daha önce tablo oluşturulmadığı için geriye boş değer dönmüştür.

```
mysql> use kutuphane;
Database changed
mysql> show tables;
Empty set (0.00 sec)
```

TABLO OLUŞTURMAK (2)

- **CREATE TABLE tablo_adi** deyimi kullanılır. Kullanılacak tüm alanlar bu deyim içerisinde belirtilmelidir.
  - Veritabanı üzerinde oluşturulan yeni tabloda:
    - Sütun adları
    - Veri tipleri
    - Kısıtlamalar (constraint)

      NULL/NOT NULL,

      PRIMARY KEY,

      FOREIGN KEY,

      CHECK,

      UNIQUE

      DEFAULT

      - Otomatik arttırma (Auto increment)

    yer alır.

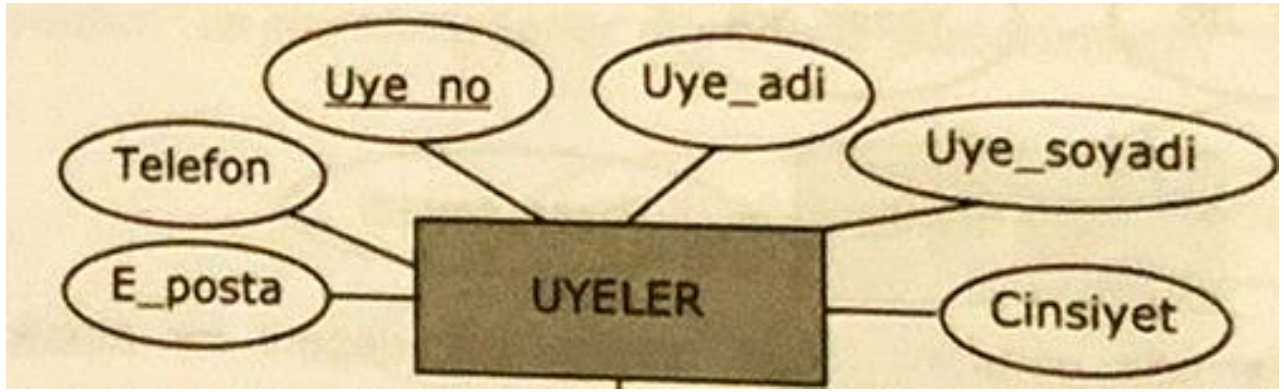
Kullanımı:

```
CREATE TABLE [table name]
(
        [column name] [data type]([size]),
        ….
        [column constraint]….
);
```

TABLO OLUŞTURMAK ÖRNEK 1

Örnek: kutuphane veritabanı projesini aktif yapıp, aşağıda bir tablo oluşturulmuştur.

❑ **CHEN MODEL İLE DAHA ÖNCE YAPILAN TASARIM**



❑ **TASARIMIN VERİTABANINDA TABLOYA DÖNÜŞTÜRÜLMESİ**

```
CREATE TABLE uyeler (
  uye_no INT NOT NULL DEFAULT 150,
  uye_adi VARCHAR(45) NULL,
  soyad VARCHAR(45) NULL,
  cinsiyet ENUM('E', 'K') NOT NULL,
  telefon VARCHAR(11) NULL,
  e_posta VARCHAR(45) NULL,
  PRIMARY KEY (uye_no));
```

TABLO OLUŞTURMAK ÖRNEK 1

Oluşturulan tabloda aşağıdaki şekilde gösterilmiştir.

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

TABLOLARDA – CONSTRAINT (1)

❑ **UNIQUE :** Benzersiz alan oluşturur.

Örneğin Her bir şehrin yalnızca kendine ait bir posta kodu vardır.

```
CREATE TABLE  adres (
   adres_no INT(255) NOT NULL,
   cadde VARCHAR(255) NULL,
   bina_no INT(255) UNSIGNED NULL,
   sehir VARCHAR(255) NULL,
   posta_kodu INT(255) UNSIGNED NULL,
   ulke VARCHAR(255) NULL,
   PRIMARY KEY (adres_no),
   CONSTRAINT UNIQUE (posta_kodu)
);
```

Not: İstenirse tanımlanan tüm CONSTRAINT ifadelerine isim verilebilir.
Örneğin söz konusu tanımlama aşağıdaki şekilde değiştirilebilir.

CONSTRAINT u_pkod UNIQUE (posta_kodu)

TABLOLARDA – CONSTRAINT (2)

❑AUTO INCREMENT

Bir alanın değerinin otomatik olarak birer birer artması istenildiğinde bu tanımlama yapılır.

Örneğin her emanet (ödünç alma) işlemi gerçekleştirildiğinde emanet_no otomatik olarak birer artsın denilebilir.

Böylece benzersiz birincil anahtar alan tanımlamış oluruz

```
CREATE TABLE EMANET (
    emanet_no INT (255) UNSIGNED NOT NULL AUTO_INCREMENT,
    emanet_tarihi DATETIME NOT NULL,
    teslim_tarihi DATETIME NOT NULL,
    emanet_uyeNo INT NOT NULL,
    FOREIGN KEY (emanet_uyeNo) REFERENCES uyeler(uye_no),
    PRIMARY KEY (emanet_no)
);
```

TABLOLARDA – CONSTRAINT (3)

❑ BİRİNCİL ANAHTAR ALAN (PRIMARY KEY)

- Birincil anahtar alan genellikle numeriktir.
- Bu alan boş bırakılamaz.
- Anahtar alan oluşturmak için birincil anahtarın alacağı değer birer birer artması istenir.

Örneğin aşağıda EMANET tablosunda anahtar alan tanımlaması yapılmıştır.

```
CREATE TABLE EMANET (
    emanet_no INT (255) UNSIGNED NOT NULL AUTO_INCREMENT,
    emanet_tarihi DATETIME NOT NULL,
    teslim_tarihi DATETIME NOT NULL,
    emanet_uyeNo INT NOT NULL,
    FOREIGN KEY (emanet_uyeNo) REFERENCES uyeler(uye_no),
    PRIMARY KEY (emanet_no)
);
```

TABLOLARDA – CONSTRAINT (3)

❑ BİRİNCİL ANAHTAR ALAN (PRIMARY KEY)

- EMANET tablosunda tanımlanan CONSTRAINT isimlendirilmesi:

```
CREATE TABLE EMANET (
    emanet_no INT (255) UNSIGNED NOT NULL AUTO_INCREMENT,
    emanet_tarihi DATETIME NOT NULL,
    teslim_tarihi DATETIME NOT NULL,
    emanet_uyeNo INT NOT NULL,
    CONSTRAINT fk_uyeno FOREIGN KEY (emanet_uyeNo) REFERENCES uyeler(uye_no),
    CONSTRAINT pk_emno  PRIMARY KEY (emanet_no)
);
```

TABLOLARDA – CONSTRAINT (4)

❑YABANCIL ANAHTAR ALAN (FOREIGN KEY)

Yabancıl anahtar alan da genellikle numerik olarak tanımlanır.

Örneğin UYELER ve EMANET tabloları *uye_no* alanı ile ilişkilendirilmiştir. UYELER tablosundaki *uye_no* EMANET tablosunda *emanet_uyeNo* adında FOREIGN KEY olarak aşağıda tanımlanmıştır..

```
CREATE TABLE EMANET (
    emanet_no INT (255) UNSIGNED NOT NULL AUTO_INCREMENT,
    emanet_tarihi DATETIME NOT NULL,
    teslim_tarihi DATETIME NOT NULL,
    emanet_uyeNo INT NOT NULL,
    FOREIGN KEY (emanet_uyeNo) REFERENCES uyeler(uye_no),
    PRIMARY KEY (emanet_no)
);
```

TABLOLARDA – CONSTRAINT (4)

❑ BİRDEN FAZLA ANAHTAR ALAN OLUŞTURMAK

```
CREATE TABLE KITAP_KUTUPHANE (
    kutuphane_no INT (255) UNSIGNED NOT NULL,
    ISBN INT (255) UNSIGNED NOT NULL,
    miktar SMALLINT NOT NULL DEFAULT 0,
    FOREIGN KEY (kutuphane_no) REFERENCES kutuphane(kutuphane_no),
    FOREIGN KEY (ISBN) REFERENCES kitap(ISBN),
    PRIMARY KEY (kutuphane_no, ISBN)
);
```

KITAP ve **KUTUPHANE** tabloları arasında çoğa-çok ilişkiden oluşturulmuş **KITAP_KUTUPHANE** tablosunda *kütüphane_no* ve *ISBN;* PK VE FK olarak tanımlanmıştır.

TABLOLARDA – CONSTRAINT (5)

❑ DEFAULT

Tablo içerisinde veri girişi yapılmadan NULL olarak bırakılan sütunlara otomatik olarak veri girişi için kullanılır.

Örneğin KITAP_KUTUPHANE tablosunda bulunan kitap sayısının varsayılan değeri 0 olarak atanması aşağıda gösterilmiştir.

```
CREATE TABLE KITAP_KUTUPHANE (
    kutuphane_no INT (255) UNSIGNED NOT NULL,
    ISBN INT (255) UNSIGNED NOT NULL,
    miktar SMALLINT NOT NULL DEFAULT 0,
    FOREIGN KEY (kutuphane_no) REFERENCES kutuphane(kutuphane_no),
    FOREIGN KEY (ISBN) REFERENCES kitap(ISBN),
    PRIMARY KEY (kutuphane_no, ISBN)
);
```

Eğer sözel bir veri varsayılan olarak tanımlanacaksa ‘ ‘ içerine alınır.
Örneğin ULKE varchar (255) DEFAULT ‘TÜRKİYE’

TABLOLARDA – CONSTRAINT (6)

❑ CHECK

CHECK ifadesi CONTSRAINT olarak kullanılır.

Örneğin bir sütunun alacağı değer belirli bir aralıkta olması istenebilir.

```
CREATE TABLE OGRENCI_NOT (
    ogrno INT (255) UNSIGNED NOT NULL,
    vize TINYINT (2)  NOT NULL CHECK (vize >= 0 AND vize <=100),
    final TINYINT (2)   NOT NULL CHECK (final >= 0 AND final <=100),
    PRIMARY KEY (ogrno)
);
```

Karşılaştırma operatörleri	
Küçük	<
Büyük	>
Küçük eşit	<=
Büyük eşit	>=
Eşit	=
Eşit değil	<>

Not: Öğrencinin notu, negatif veya 100 üzerinde girildiğinde bu tabloya kayıt işlemi gerçekleşmeyecektir.

TABLOLARDA – CONSTRAINT (7)

❑ CHECK

Koşul ifadesi birden fazla olduğu durumlarda CHECK CONTSRAINT olarak aşağıdaki şekildeki gibi tanımlanır.

```
CREATE TABLE hayvan(
        ID INT NOT NULL,
        ad VARCHAR(30) NOT NULL,
        yetiştiren VARCHAR(20) NOT NULL,
        yaş INT,
        Cinsiyet VARCHAR(9),
        PRIMARY KEY(ID),
        check(Cinsiyet in ('Erkek', 'Dişi', 'Tespit edilmedi') )
        );
```

ÖRNEK UYGULAMA

Örnek olarak bir araç kiralama veritabanı için veri tabanı şeması aşağıda verilmiştir.

- MUSTERI (m_kod, mad, msoyad, madres, mtel)
- ARAC (arac_no, model, marka, plaka, fiyat)
- KIRALAMA (mkod, aracno, tarih, saat, tes_tarihi, tes_saati)

- Altı çizgili olarak verilen sütunlar birincil anahtar (PRIMARY KEY) alanı ifade etmektedir.
- KİRALAMA tablosunun *mkod* ve *aracno* sütunları MÜŞTERİ ve ARAÇ tablosunun birincil anahtar sütunları ve kendisinin FOREIGN KEY sütununu temsil etmektedir.
- ARAÇ tablosunda bulunan model sütunu için girilecek bilgi 1900-2015 arasında olacak şekilde sınırlanmalıdır.

ÖRNEK UYGULAMA

```
CREATE DATABASE uygulama2;
USE uygulama2;

CREATE TABLE musteri (
 mkod       INT            NOT NULL  PRIMARY KEY,
 mad        VARCHAR(50)    NOT NULL,
 msoyad     VARCHAR(50)    NOT NULL,
 madres     VARCHAR(255),
 mtel       VARCHAR(15)
)

CREATE TABLE arac (
 aracno     INT            NOT NULL PRIMARY KEY,
 model      INT            NOT NULL,
 marka      VARCHAR(50),
 plaka      VARCHAR(25),
 fiyat      VARCHAR(15),
 CONSTRAINT chkmodel CHECK(model LIKE '[1-2][0-9][0-9][0-9]'))

CREATE TABLE kiralama (
 mkod       INT            NOT NULL,
 aracno     INT            NOT NULL,
 tarih      VARCHAR(10),
 saat       VARCHAR(8),
 tes_tarih VARCHAR(10),
 tes_saat   VARCHAR(8),
 CONSTRAINT fk_mkod FOREIGN KEY(mkod) REFERENCES musteri(mkod),
 CONSTRAINT fk_aracno FOREIGN KEY(aracno) REFERENCES arac(aracno),
 CONSTRAINT pkkey PRIMARY KEY(mkod, aracno)
)
```

YABANCIL ANAHTAR ALAN (FOREIGN KEY)

- Tablolar arasında oluşturulan FOREIGN KEY ilişkiler referans olarak kullanılan tablolardan **kayıt silme** ve **güncelleme** işleminde ilişkilerden dolayı hata vermesine neden olacaktır.

YABANCIL ANAHTAR ALAN (FOREIGN KEY)

- Örneğin, bir önceki uygulamada müşteri tablosuna 1 müşteri koduyla kaydedilen bir müşteriye araç kiralaması yapıldığını düşünelim.
- Daha sonra müşteri tablosundan 1 olan müşteri kodunu değiştirmek istediğimizde hatayla karşılaşacağız. Çünkü müşteri tablosunun mkod sütunu kiralama tablosunda FOREIGN KEY olarak kullanılmıştır.
- Bu tür sorunların engellenmesi için FOREIGN KEY ilişkileri oluşturulurken güncelleme veya silme işlemi için izin verilmelidir.

CASCADE

- FOREIGN KEY oluşturulurken ***ON DELETE CASCADE*** ifadesi kullanılırsa; referans tablodan silinen satır ile ilişkili olan diğer tablolardaki ilgili satırlar da silinecektir.

- Aynı şeklide FOREIGN KEY oluşturulurken ***ON UPDATE CASCADE*** ifadesi kullanılırsa; referans tabloda güncellenen satır ile ilişkili olan diğer tablolardaki da ilgili satırlar otomatik olarak güncellenecektir.

CASCADE

```
CONSTRAINT fk_mkod FOREIGN KEY ( mkod ) REFERENCES müşteri ( mkod ) ON UPDATE CASCADE ON DELETE CASCADE,
CONSTRAINT fk_aracno FOREIGN KEY ( aracno ) REFERENCES araç ( aracno ) ON UPDATE CASCADE ON DELETE CASCADE,
```

- Örneğin, "müşteri" tablosunda bulunan 1 müşteri kodlu müşteriye araç kiralaması yapıldığını düşünelim.
- Daha sonra "**müşteri**" tablosundan 1 olan müşteri kodu 2 olarak değiştirilirse; otomatik olarak "**kiralama**" tablosunda bulunan 1 müşteri kodları da 2 olacaktır.
- Aynı şekilde 1 kodlu müşteri "**müşteri**" tablosundan silinirse "**kiralama**" tablosundan da 1 nolu müşteriye yapılan kiralamalar silinecektir.

B. ALTER

- Daha önce oluşturulmuş veritabanı nesnesinin özelliğini değiştirmek için kullanılır.
- Yapılmak istenen değişiklik parametre olarak verilir.
- Kullanılan temel parametreler:
  - RENAME
  - CHANGE
  - MODIFY
  - ADD
  - DROP

Tabloda Sütunları Değiştirmek

- **Tablo adını değiştirmek:**
  - **ALTER TABLE** tablo_adı RENAME TO yeni_tablo_adi;
  - Örnek:
```
ALTER DATABASE deneme2  RENAME TO  ARAC_KIRALAMA ;
```
  - Deneme2 adındaki tablo ARAC_KIRALAMA olarak değiştirilmiştir.
- **Tabloda sütun adını değiştirmek:**

ALTER TABLE tablo_adi CHANGE COLUMN sütun_adı özellikler

Örneğin:
```
ALTER TABLE musteri CHANGE COLUMN mad musteri_adi VARCHAR(255) NOT NULL ;
```

mad sütunu, müşteri_adi olarak değiştirilmiştir.

Tabloda Sütunları Değiştirmek

❑ Sütunun veri tipini değiştirmek:

- ALTER TABLE table_name MODIFY COLUMN column_name datatype

Örnek:

```
ALTER TABLE arac MODIFY COLUMN model SMALLINT NOT NULL;
```

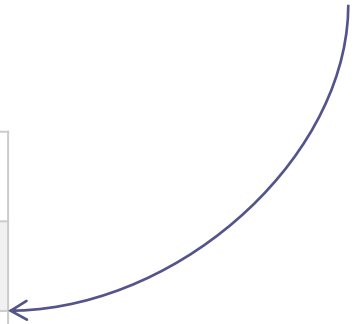
- Model sütununun daha önceki veri tipi, varchar (255) olarak tanımlanmıştır. Bu alanın veri tipi SMALLINT olarak değiştirilmiştir.

Tabloda Sütunlara Özellik Eklemek

- **Örnek:**

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

ALTER TABLE Persons
ADD DateOfBirth **date**



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

Tabloda Sütunlara Kısıtlayıcı (Constraint) Eklemek

```
alter table ogrenci add unique (ogrno);
```

- Var olan öğrenci tablosu için ogr_no sütununa UNIQUE özelliği eklemektedir.

```
alter table uyeler add primary key (eposta);
```

- Var olan öğrenci tablosu için ogr_no sütununa PRIMARY KEY özelliği ekler.

Tabloda Sütunlara Kısıtlayıcı (Constraint) Eklemek

```
alter table ogrenci add unique (ogrno);
```

- Var olan öğrenci tablosu için ogr_no sütununa UNIQUE özelliği eklemektedir.

```
alter table uyeler add primary key (eposta);
```

- Var olan öğrenci tablosu için ogr_no sütununa PRIMARY KEY özelliği ekler.

```
ALTER TABLE emanet ADD CONSTRAINT fk_uyeno FOREIGN KEY (emanet_uyeno)
REFERENCES uyeler(uye_no);
```

- Emanet tablosunda tanımlanmış olan emanet_uyeno sütununa yabancıl anahtar alan özeliğini ekler.

Tabloda Sütunlara Özellik Silmek

```
alter table ogrenci_not drop primary key ;
```

- Öğrenci_not tablosunun birincil anahtar özelliği silinmiştir.

```
ALTER TABLE emanet DROP FOREIGN KEY fk_uyeno;
```

- Emanet tablosunda emanet_uyeNo sütununun yabancıl anahtar özelliği kaldırılmıştır.

C. DROP

- Veritabanı içerisinde var olan nesneleri veya veritabanını silmek için kullanılır.
- **DROP DATABASE** *veritabanı_adı*

Örnek:
```
DROP database deneme;
```

Deneme adında veritabanı kaldırılmıştır.

- **DROP TABLE** *tablo_adı*

Örnek:
```
DROP table hayvan;
```

Hayvan adında veritabanı kaldırılmıştır.

D. TRUNCATE

- Tablodaki verileri silmek için kullanılır.

Örnek:

```
truncate table ogrenci_not;
```

ogrenci_not tablosu kaldırılmamıştır fakat içerisindeki tüm kayıtlar silinmiştir.

Kaynakça

- Turgut Özseven, Veritabanı Yönetim Sistemleri-1 , Ekin Basın Yayın Dağıtım 6. Baskı
- https://dev.mysql.com/doc/refman/5.7/en/sql-data-definition-statements.html
- www.w3schools.com/sql