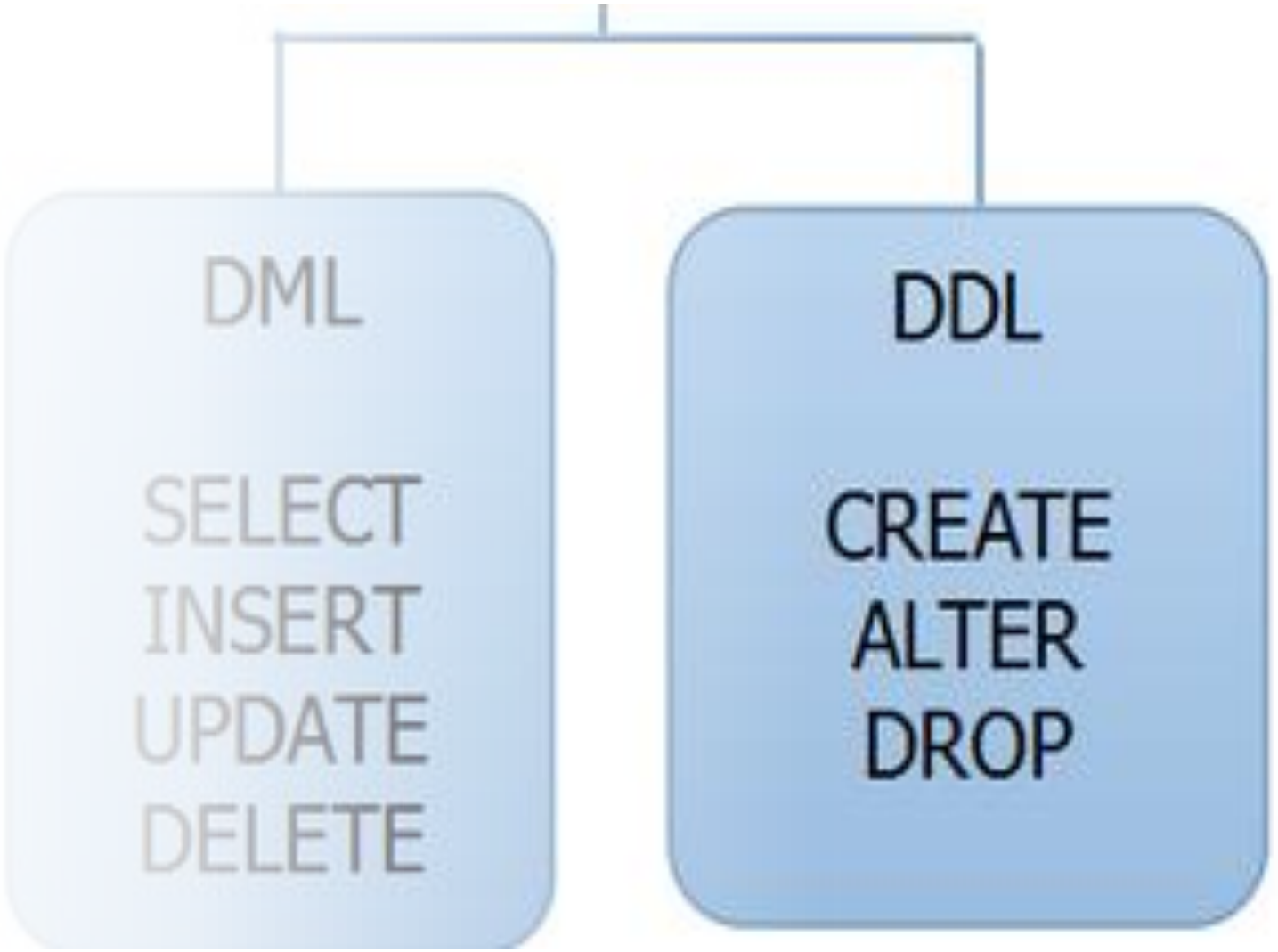
Veri Tabanı Yönetim Sistemleri

Hafta 10 – DML-DDL

Örnek Uygulamalar



SQL DDL (Data Definition Language) Veri Tanımlama Dili

Verilerin tutulduğu nesneler olan tabloların oluşturulmasını, silinmesini ve bazı temel özelliklerinin düzenlenmesini sağlar.

- En yaygın kullanılan DDL komutları şunlardır:

- **CREATE TABLE :** Veri tabanı üzerinde bir tablo oluşturmak için kullanılır.
- **ALTER TABLE :** Tabloda değişiklik yapmak, yeni bir sütun eklemek, sütunun tipini veya uzunluğunu değiştirmek gibi yapısal değişiklikler yapılması için kullanılır.
- **DROP TABLE :** Tabloyu fiziksel olarak siler. (verilerle birlikte)
- **CREATE INDEX :** Index oluşturmak için, tablonun sütun adı üzerinde indeks oluşturur.
- **CREATE VIEW :** Görüntü(View) oluşturmak için kullanılır.
- **DROP VIEW :** Görüntüyü siler.

SQL DML (Data Manipulation Language) Veri İşleme Dili

Veri girmek, değiştirmek, silmek ve verileri almak için kullanılan DML komutlarının tümü olarak tanımlanır. Kısaca veri tabanında bilgi üzerinde çalışmayı sağlar. Bilgiyi çağırma, bilgiye yeni bir şeyler ekleme, bilgiden bir şeyler silme, bilgiyi güncelleştirme işlemlerini yapar.

Temel komutlar:

- **SELECT :** Veri tabanındaki nesneden alan çağırmayı ve gereken bilgiyi görebilmeyi sağlar.
- **UPDATE :** Nesnede bilgi güncellemeyi sağlar.
- **INSERT :** Nesneye alan eklemeyi sağlar.
- **DELETE :** Nesneden alan silmeyi sağlar.
- **TRUNCATE :** Nesnenin içini boşaltır.

SQL DCL (Data Control Language) Veri Kontrol Dili

Veri Kontrol Dili bir veri tabanı kullanıcısı veya rolü ile ilgili izinlerin düzenlenmesini sağlar. DCL komutlarını kullanabilmek için SQL Server'da varsayılan değer (default) olarak yetki sahibi olan gruplar: sysadmin , dbcreator , db_owner , db_securityadmin 'dir. Sunucuya dışarıdan bir erişim sağlamak için bir giriş (login) oluşturulmalıdır.

Temel DCL komutları şunlardır:

- **GRANT :** Veri tabanı kullanıcısına, veri tabanı rolüne veya uygulama rolüne izinler vermek için kullanılan komuttur.
- **DENY :** Kullanıcıların haklarını kısıtlayan komuttur.
- **REVOKE :** Daha önce yapılan tüm kısıtlama ve izinleri iptal eden komuttur. Bir nesneyi oluşturan kullanıcının REVOKE ile nesne üzerindeki yetkilendirme ve kullanma hakkı yok edilemez. REVOKE komutunu, sys_admin rolüne veya db_owner, db_securityadmin sabit veri tabanı rollerine sahip kullanıcılar ve nesne için dbo olan kullanıcı çalıştırabilir.

DDL

- CREATE DATABASE ifadesi yeni veri tabanı oluşturmak için kullanılır.
  - 1) Bir test veri tabanı oluşturalım.

- DROP DATABASE var olan SQL veri tabanını kaldırmayı sağlar.
  - 2) Veri tabanımızı silelim.

- BACKUP DATABASE ifadesi var olan bir veri tabanının tam bir yedeğini oluşturmaya yarar.
- Differential back up son full veri tabanı yedeklemeden sonraki değişen veri tabanı kısımlarının yedeğini alır.

Not: Veritabanı yedeğinin daima mevcut veri tabanından farklı bir disk üzerine alınması herhangi bir disk arızasında veri tabanı ile birlikte yedeklerinizi de kaybetmenizi engeller.
Not: differential back up yedekleme süresini kısaltır(yalnızca değişiklikler yedeklendiği için).

- CREATE TABLE ifadesi veri tabanında yeni bir tablo oluşturmak için kullanılır.
- CREATE TABLE *table_name* (
  column1 datatype,
  column2 datatype,
  column3 datatype,
  ....
  );
- Veri tiplerine bir göz atalım.

VERİ TİPLERİNİN SEÇİLMESİ

SAYISAL VERİ TİPLERİ

bigint	Minimum: -2^63 (-9,223,372,036,854,775,808) Maksimum: 2^63-1 (9,223,372,036,854,775,807)	8 Byte
int	Minimum: -2^31 (-2,147,483,648) Maksimum: 2^31-1 (2,147,483,647)	4 Byte
smallint	Minimum: -2^15 (-32,768) Maksimum: 2^15-1 (32,767)	2 Byte
tinyint	Minimum: 0 Maksimum: 255	1 Byte
bit	0 ya da 1 değerini alır.	Eğer tabloda 8 ya da daha az bit kolonu varsa 1 byte, 8'den fazla ise 2 byte yer kaplar.

	A	B	C	D	E	F	G	H
1	KOLON ADI	VERİ TİPİ	KAPLADIĞI ALAN	SATIR SAYISI	KAPLADIĞI ALAN (BYTE)	KAPLADIĞI ALAN (MB)		
2	ID	INT	4	2.000.000	8.000.000	7,63		
3	COUNTRYID	TINYINT	1	2.000.000	2.000.000	1,91		
4	CITYID	SMALLINT	2	2.000.000	4.000.000	3,81		
5	GENDER	BIT	1	2.000.000	2.000.000	1,91		
6								
7						15,26		
8								

VERİ TİPLERİNİN SEÇİLMESİ

SAYISAL VERİ TİPLERİ

decimal/ numeric	Minimum: – 10^38 +1	Hassasiyetine göre diskte kapladığı alan değişir. 1'den 9 'a kadar Hassasiyet için: 5 byte
	Maksimum: 10^38 – 1.	10'dan 19'a kadar Hassasiyet için: 9 byte 20'den 28'a kadar Hassasiyet için: 13 byte 29'dan 38'e kadar Hassasiyet için: 17 byte
money	Minimum: -922,337,203,685,477.5808 Maksimum: 922,337,203,685,477.5807	8 Byte
smallmoney	Minimum: – 214,748.3648 Maksimum: 214,748.3647	4 Byte
float	-1.79308 ile –2.23308, 0	7 basamağa kadar 4 Byte
	2.23308 ile 1.79308	15 basamağa kadar 8 Byte
Real	-3.438 ile -1.1838, 0 1.1838 ile 3.438	4 Byte

Decimal(18,2) : 18 digit, virgülden sonra 2 digit alabilir. Toplamda 18 digit.

Fazla karakterler girsek otomatik olarak yuvarlama yapabilir.

Money: özel bir decimal tipi gibi virgülden sonra 4 digit alır.

String Veri Tipleri

```
Declare @c as Char(30);
Declare @n as Nchar(30);
Declare @v as varchar(30);
Set @c = 'Merhaba MSSQL Server!';
Set @n = 'Merhaba MSSQL Server!';
Set @v = 'Merhaba MSSQL Server!';
Select DATALENGTH(@c),
DATALENGTH(@n),DATALENGTH(@v);
```

Veri Tipi	Sınıfı	Boyut	Veri Yapısı
Char	Karakter	Değişir	Sabit uzunlukta karakter verisi.Atanan değer uzunluğundan kısa olan değerler atanan uzunluğa tamamlanır. Veri Unicode değildir. Belirlenmiş max uzunluk 8000 karakterdir.
VarChar	Karakter	Değişir	Değişken uzunlukta karakter verisi. Kısa değerler boyuta tamamlanmaz. Unicode değildir. 8000 karakter alabilir fakat çok geniş karakter alanı berlirlemek için (2^31 byte kadar) varchar(MAX) kullanılabilir.
Text	Karakter	Değişir	Daha önceki versiyonları desteklemek amacıyla eklenmiştir. Bunun yerine varchar(Max) kullanımı tavsiye edilmektedir.
Nchar	Unicode	Değişir	Sabit uzunlukta Unicode karakter verisi. Atanan değer uzunluğundan kısa olan değerler atanan boyuta tamamlanır. Belirlenmiş maksimum uzunluk 4000 karakterdir.
NVarChar	Unicode	Değişir	Değişken uzunlukta Unicode karakter verisi. Kısa değerler tamamlanmaz. Belirlenmiş maksimum uzunluk 4000 karakterdir, fakat çok geniş karakter alanı (maksimum 2^31 byte'a kadar) olarak belirlemek için max anahtar sözcüğünü kullanabilirsiniz.
Ntext	Unicode	Değişir	Text veri tipi gibi, bu veri tipide sadece eski versiyonlara destek amacıyla mevcuttur. Yerine, **varbinary(max)** kullanın.

- Char(10): Karakter sayısı maksimum 10'u geçemez. Sayıya tamamlayacak şekilde boşluk atılır. Her karakter 1 byte yer kaplar. Yani 0-255 rakam değeri olabilir.
- Nchar(50): Sayıya tamamlayana kadar sonuna boşluk atar.
- Ntext : Sonuna boşluk atarak tamamlamaz.
- Nvarchar(50): Boşlukla doldurma yapmaz.
- Nvarchar(MAX): boşlukla doldurma yapmaz.
- Text : Boşlukla doldurma yapmaz.
- Varchar(50) : Boşlukla doldurma yapmaz.
- Varchar(MAX) : Boşlukla doldurma yapmaz.

NOTLAR;

- Varchar; girilen veri ne kadarsa hafızada o kadar yer kaplar. O yüzden kullanımı esnek ve avantajlıdır.
- Char; kullanımı neden gerekebilir?
  - öncelikle uzunluğu sabit veri sağlanmaktadır. Arama işlemleri yaparken varcharda önce verinin boyutu bulunması gerekebilir.
  - Char için arama daha kolay gerçekleştirilebilir.
  - TC, Telefon gibi sabit sayılı değerlerde char kullanılabilir. İsim adres gibi verilerde char kullanımı pek uygun değildir.
- Nvarchar; N değeri Unicode kısaltmasıdır. Unicode: uluslararası karakterlerin desteklendiğini göstermektedir. Char, varchar gibi tiplerde karakter 1 byte kaplamaktadır bu ise 255 farklı karakter demektir. Yabancı karakter girmek istediğimizde 255 değeri yeterli olmayabilir. Örneğin Japonca bir karakter saklanması istenebilir. Dezavantajı her karakter 2 byte yer kaplamaktadır.
- Char() vb. içerisine en fazla 8000 karakter uzunluğa izin verilmektedir. Bunun sebebi Sql Server'da en küçük veri birimi «Page» denilen yapılardır. Bize bir dosya gibi görünse de sql server geri planda bunları 8 KB'lık page ler halinde saklar. 8000 karakter bir page doldurduğu için maksimum o kadar tutulabilir. Bu nedenle nchar() içerisinde maksimum 4000 kullanılabilmektedir.
- Varchar(MAX) nvarchar(MAX) ise 8000 karakteri veya 4000 karakteri(Unicode dahil) geçebilen verilerde 2 GB kadar veri tutabilmemizi sağlayan MAX değeri içermektedir. Örn; Kitap vb. saklanmak istenirse.
- Text ve ntext eski sürümlerde olup artık kullanılmamaktadır. Uyumluluğu sağlamak adına yer alır.

Char vs Nchar

	Char	NChar
Avantaj	Depolama alanı değişken tanımlanırken belirtilen boyuta eşit olduğu için daha az fiziksel alana ihtiyaç duyar	Birden fazla dili ve bölge desteği sağlar. Böylece herhangi bir Unicode karakteri dönüşüm kaygısı olmaksızın saklanabilir.
Dezavantaj	Varchar sütunda Unicode tutmak için bir dönüşüm tekniğine ihtiyaç duyar	Depolama alanı değişken tanımlanırlen belirtilen boyutun iki katı kadar olduğu için daha fazla fiziksel alana ihtiyaç duyar
Kullanım	Saklamayı düşündüğünüz data Unicode olmayan ASCII karakterleri ise faydalıdır.	Saklamayı düşündüğünüz data farklı dil ve bölgeler içeriyorsa faydalıdır. Gecelekte Unicode karakterler kullanmayı planlıyorsanız Nchar kullanmanız tavsiye edilir.

TARİH-SAAT VERİ TİPLERİ

date	Minimum: 0001-01-01	4 Byte
	Maksimum: 9999-12-31	
smalldate	Minimum: 1900-01-01	3 Byte
	Maksimum: 2079-06-06	
datetime	Minimum: 1753-01-01 00:00:00.000	8 Byte
	Maksimum: 9999-12-31 23:59:59.997	
datetime2	Minimum: 0001-01-01 00:00:00.0000000	1-2 Hassasiyet İçin = 6 Byte
	Maksimum: 9999-12-31 23:59:59.9999999	3-4 Hassasiyet İçin = 7 Byte
		5-7 Hassasiyet İçin = 8 Byte
datetimeoffset	Minimum: 0001-01-01 00:00:00.0000000	1-2 Hassasiyet İçin = 8 Byte
	Maksimum: 9999-12-31 23:59:59.9999999	3-4 Hassasiyet İçin = 9 Byte
	Time zone offset Aralığı: -14:00 / +14:00	5-7 Hassasiyet İçin = 10 Byte
time	Minimum: 00:00:00.0000000	5 Byte(Default olarak kullanılırsa)
	Maksimum: 23:59:59.9999999	

DİĞER VERİ TİPLERİ

image		Maksimum değeri: 2^31-1 (2,147,483,647) Byte
binary	0 ile 8000 arasında	Tanımlandığı değer kadar Byte.
		Binary(10) -> 10 Byte
varbinary	0 ile 8000 arasında	Tanımlandığı değer + 2 Byte
varbinary(MAX)	0 ile 2 147 483 647 arasında	Tanımlandığı değer + 2 Byte
		Maksimum değeri: 2^31-1 (2,147,483,647) Byte
sql_variant		Bazı veri tiplerinin değerlerini saklamak için kullanılır.
		Aşağıdakiler hariç:
		varchar(max),varbinary(max),nvarchar(max),xml,text,
		ntext,image,rowversion(timestamp),sql_variant,
		geography,hierarchyid,geometry,User-defined types, datetimeoffset
Xml		Xml veriler için kullanılır.
Table		Sonradan kullanım amacıyla bir sonuç
		kümesini saklamak için kullanılır.

uniqueidentifier	GUID(global olarak tekilliği garanti eder) veriyi tutar.
	select NEWID() script'ini çalıştırdığınızda aşağıdaki
	gibi bir GUID veri oluşturur.
	A4C5DB26-7F18-4B4F-A898-E7DE26A8446A
	Bazen veritabanlarında tekilliği sağlamak için kullanılır.
	Ama bu amaçla kullanıldığında genelde performansı düşürür.
hierarchyid	Hiyerarşik yapılarda, hiyerarşideki pozisyonları
	temsil etmek için kullanılır.
geography	Dünyadaki koordinat sistemini tutar.
	Dünyanın eğimlerini de hesaba katarak.
geometry	Euclidean (flat) sistemi ile koordinat sistemini tutar.
	Sadece 2 düzlem üzerinden hesaplanır.
	Dünyanın eğimlerini hesaba katmaz.

Tablo Oluşturma

- Personel tablosu oluşturalım (ID,İsim,Soyisim,Adres,Şehir)

- **Tablo Oluşturma Kuralları:**
  - Tablo adı veri tabanı içerisinde eşsiz olmalıdır.
  - Sütun adları Tablo içerisinde eşsiz olmalıdır.
  - Tablo isimlerine Altçizgi (_) dışında özel veya sayısal karakterler ile başlamayın.
  - Tablo isimlerinde boşluk karakteri kullanmayın.
  - Her nesne adı minimum 1, maksimum 128 karakter içermelidir.
  - Bir tablonun sahip olabileceği maksimum sütun sayısı 1024 sütundur.

- DROP TABLE ifadesi mevcut tabloyu kaldırmaya yarar.

  - Bir önceki tablolarımızı kopyalayıp oluşan kopyaları veri tabanından silelim.

ALTER

- ALTER TABLE ifadesi mevcut veri tabanında tablo sütunları üzerinde ekleme, silme ve değişiklik yapmada kullanılır. Aynı zamanda mevcut tablolar üzerinde çeşitli kısıtlamaları ekleme ve kaldırma işlemlerinde de kullanılır.
  - Veri tipinin boyutu ve türü değiştirilebilir.
  - Mevcut tabloya yeni bir alan eklenebilir.
  - Alan adı değiştirilebilir.
  - Tablodan sütun kaldırılabilir…

- Personel Tablosu Oluşturalım (ID,Ad,Soyad,Sehir,Maas)

- Sütun Ekleyelim:

- ALTER TABLE *table_name*
  ADD *column_name datatype*;

Sütun Silelim:

ALTER TABLE *table_name*
DROP COLUMN *column_name*;

Sütun Değiştirelim:

ALTER TABLE *table_name*
ALTER COLUMN *column_name datatype*;

Örn:

- Personel tablosuna Doğum Tarihi alanı ekleyelim.(date)
- Sehir alanını int olacak şekilde güncelleyelim.
- Maas alanını silelim.

TRUNCATE

Var olan bir tablodan mevcut kayıtları kalıcı olarak silmek için kullanılabilir.

TRUNCATE TABLE ‘Table Name’

SP_RENAME

- Tabloya yeni bir isim vermek için kullanılabilir.
- SP_Rename ‘Old Table Name’,’New Table Name’

Not: sp_help : Tablo yapısını görmek için kullanılabilir.

Not: mevcut veri tabanındaki tabloları görüntülemek için

- Select * from sysobjects where XTYPE=‘u’ kullanılabilir.

SQL Kısıtlar

- Kısıtlar tablo verileri için özel kurallar belirlemede kullanılırlar.
- Tabloya yerleştirilecek veri tipini sınırlandırmaya yaradıkları için doğruluk ve güvenlik sağlarlar. Kısıtlama ve eylem arasında herhangi bir ihlal durumu oluşursa eylem iptal edilir. Tablo veya sütun düzeyinde uygulanabilirler.
- Sık kullanılan kısıtlar:

NOT NULL - Sütunun NULL değer olmamasını sağlar.

UNIQUE - Sütundaki tüm değerlerin farklı olmasını gerektirir

PRIMARY KEY - NOT NULL ve UNIQUE kısıtlarının bir kombinasyonudur. Tablodaki her bir satır eşsiz olarak belirtir.

FOREIGN KEY - Diğer bir tabloda bulunan bir kayıt/satırı benzersiz bir şekilde gösterir.

CHECK - bir sütundaki tüm değerlerin belirli bir koşulu sağlaması gerektiğini gösterir.

DEFAULT - Bir değer belirtilmediği durumlarda sütun için varsayılan bir değer ayarlar.

INDEX - Veri tabanında çok hızlı olarak veri oluşturma ve veri çağırmada kullanılır

UNIQUE

UNIQUE kısıtı sütundaki tüm değerlerin farklı olmasını gerektirir.

UNIQUE vePRIMARY KEY kısıtlarının her ikisi de bir sütun veya sütun grubunun benzersizliğini garanti altına alır.

PRIMARY KEY kısıtı otomatik olarak bir UNIQUE kısıtına sahiptir. Ancak her bir tabloda birden fazla UNIQUE kısıtı oluşturabiliyorken yalnızca bir tane PRIMARY KEY kısıtı oluşturulabilir.

Ayrıca UNIQUE NULL değerlere izin verebilirken PRIMARY KEY kısıtı NULL değerlere izin vermez.

Bir tabloda birden fazla UNIQUE Key oluşturabiliriz.

```
CREATE TABLE Persons (
ID int NOT NULL UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int );
```

Bir UNIQUE kısıt adı oluşturmak ve birden fazla alanı için UNIQUE kısıt tanımlamak için:

```
CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
CONSTRAINT UC_Person UNIQUE (ID,LastName) );
```

Zaten oluşturulmuş bir tabloda UNIQUE;

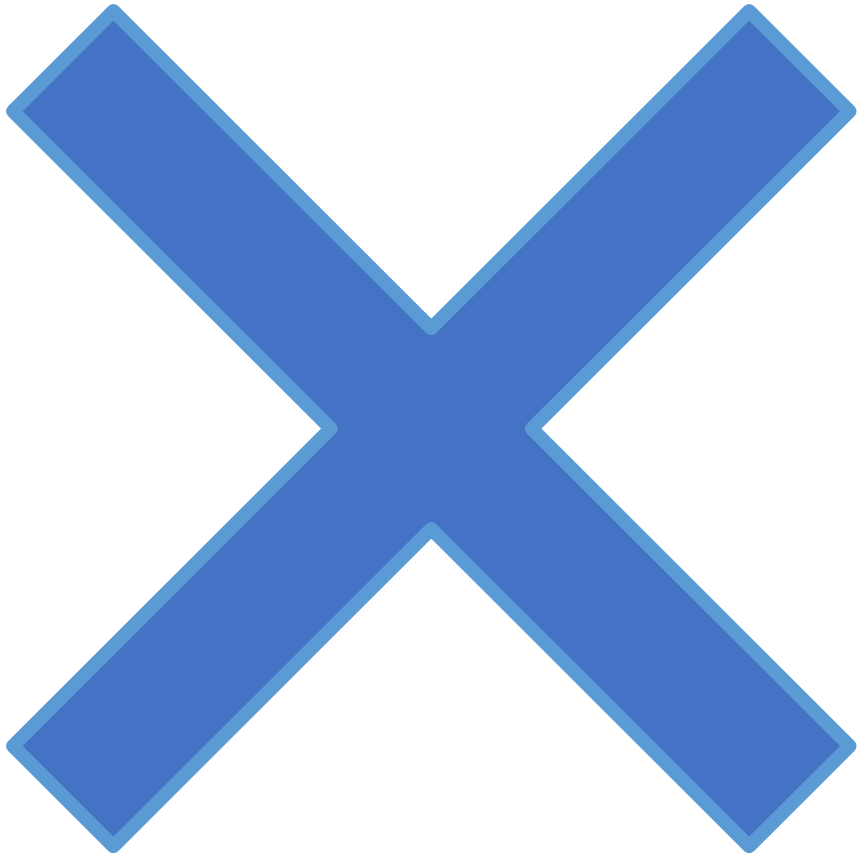
```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);
```

```
ALTER TABLE Persons
DROP CONSTRAINT UC_Person;
```

NOT NULL

- NOT NULL kısıtı sütunun NULL değerleri kabul etmemesini sağlar. Bir alanın daima değer içeriyor olmasını gerektirir.
- CREATE TABLE Persons (
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255) NOT NULL,
  Age int
  );



CHECK

- CHECK bir alandaki değer aralığını sınırlandırmak için kullanılır.
- Tek bir alanda tanımlanırsa bu alan için yalnızca belirli değerler girilmesine izin verir. Tablo üzerinde tanımlanırsa belirli alanlardaki değerleri satır üzerindeki diğer alanlardaki değerlere dayalı sınırlandırabilir.

- ```
  CREATE TABLE Persons (
      ID int NOT NULL,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255),
      Age int CHECK (Age>=18)
  );
  ```

- ```
  CREATE TABLE Persons (
      ID int NOT NULL,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255),
      Age int,
      City varchar(255),
      CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sakarya')
  );
  ```

Var olan tabloda değişiklik:

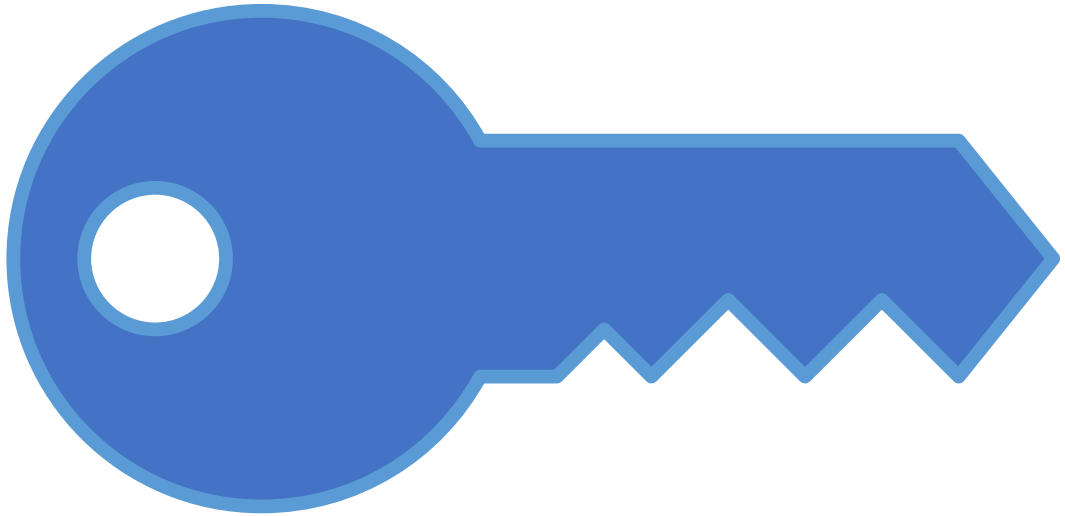
```
ALTER TABLE Persons
ADD CHECK (Age>=18);
```

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City='Sandnes');
```

```
ALTER TABLE Persons
DROP CONSTRAINT CHK_PersonAge;
```

PRIMARY KEY

- Bir alana NULL değerler girilmesini engellemenin yanı sıra tekrarlı veriye de izin vermeyen UNIQUE ve NOT NULL kombinasyonu bir kısıttır. Tabloda yalnızca bir PRIMARY KEY oluşturulur fakat bu key birkaç alandan oluşabilir.



- CREATE TABLE Persons (
  ID int NOT NULL PRIMARY KEY,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int
  );

- CREATE TABLE Persons (
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int,
  CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
  );

- DİKKAT: Yukarıdaki kısıtta yalnızca bir tane PRIMARY KEY vardır ancak değeri iki alandan oluşmaktadır.

Mevcut tabloya PRIMARY KEY düzenlemesi yapmak için:

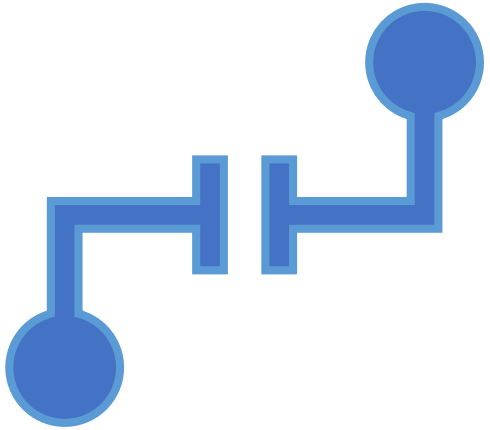
```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);
```

- DİKKAT: PRIMARY KEY Olacak alanlar tablo oluşturulurken NULL değer alamayacak şekilde tanımlanmış olmalıdırlar.

```
ALTER TABLE Persons
DROP CONSTRAINT PK_Person;
```

FOREIGN KEY



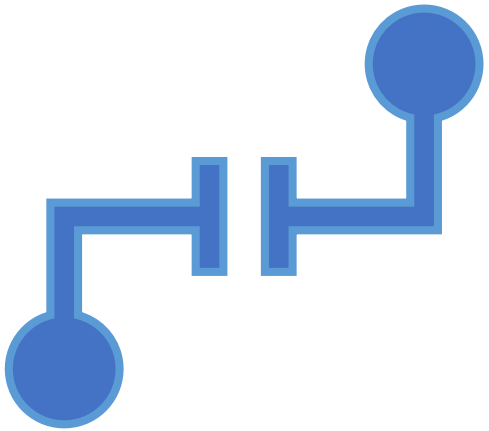
- Veri tabanındaki en önemli kavramlardan biri tablolar arasındaki ilişkilerin oluşturulmasıdır. Bu ilişkiler birden fazla tablolarda tutulan verilerin ilişkilendirilmesi ve etkili bir şekilde elde edilmesi için bir mekanizma oluşturur. FOREIGN KEY tabloları bağlamak için kullanılır.
- FOREIGN KEY, diğer bir tablodaki PRIMARY KEY alana referans olan bir tablodaki alan veya alanlardır.
- FOREIGN KEY içeren tablo child tablo, aday anahtarı içeren diğer tablo ise referans veya parent tablo olarak adlandırılır.

	ID	StundentNo	DeptID	Name	Surname	BirthDate
1	1	b000000000	1	Muhammed	Kotan	NULL
2	2	b000000001	1	Ömer Faruk	Seymen	NULL
3	3	b000000002	1	Yasin Sena	Atlan	NULL
4	6	b000000003	2	Hüseyin	Demirci	NULL
5	7	b000000004	2	Özgür	Yıldız	NULL
6	9	b000000005	3	Necati	Kısmet	NULL
7	10	b000000006	5	Ayetullah	Akar	NULL
8	12	b000000007	6	Ahmet	Erengil	NULL

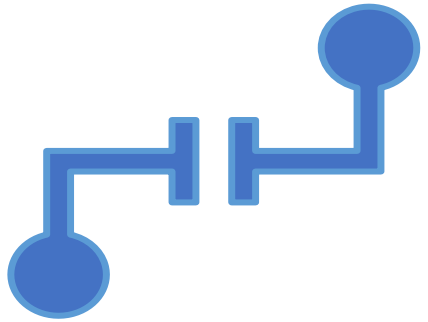
	BolumID	bolumAdi	fakulteID
1	1	Bilişim Sistemleri Mühendisliği	1
2	2	Bilgisayar Mühendisliği	1
3	3	Yazılım Mühendisliği	1
4	4	Endüstri Mühendisliği	2
5	5	Makina Mühendisliği	2

- FOREIGN KEY tablolar arası bağlantıları ortadan kaldıracak eylemleri engellemek için kullanılır.
- FOREIGN KEY kısıtı aynı zamanda bu alana girilecek geçersiz bir veri durumunu da engeller çünkü veri işaret edilen tablodaki değerlerden biri olmak zorundadır.

- CREATE TABLE Orders (
  OrderID int NOT NULL PRIMARY KEY,
  OrderNumber int NOT NULL,
  PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
  );



- CREATE TABLE Orders (
  OrderID int NOT NULL,
  OrderNumber int NOT NULL,
  PersonID int,
  PRIMARY KEY (OrderID),
  CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
  REFERENCES Persons(PersonID)
  );



- FOREIGN KEY kısıtı oluşturulup tablo bağlantısı sağlandığında bazı kurallar ele alınmalıdır.

  - FK alanında, Ana tablonun referans alınan sütununda bulunmayan bir veri girişi yapılmamalıdır.
  - Child tabloda ne olacağını tam belirtmeden child tabloda karşılığı olan bir refarans alınan sütun değeri Ana tabloda güncellenmemelidir.
  - Child tabloda ne olacağını tam belirtmeden child tabloda karşılığı olan bir refarans alınan sütun değeri Ana tablodan silinmemelidir.

- Eğer Child tabloda karşılığı olan Ana tablodaki bir kaydı silmek veya güncellemek istiyorsak Cascade kuralları olarak bilinen bazı silme ve güncelleme işlemlerini ele almamız gerekir.

- On Delete Cascade: Child tablo tarafından referans alınan Ana tablodaki bir anahtar değeri silmek için kullanılır. Çocuk tablodaki bu FK içeren tüm satırlarda aynı zamanda silinir.

- On Update Cascade: Child tablo tarafından referans alınan Ana tablodaki bir anahtar değeri güncellemek için kullanılır. Çocuk tablodaki bu FK içeren tüm satırlarda aynı zamanda güncellenir.

DEFAULT

Bir alan için varsayılan değer sağlamak için kullanılır.

Eğer başka bir değer belirtilmediyse tüm yeni kayıtlara bu varsayılan değer atanacaktır.

- ```
  CREATE TABLE Persons (
      ID int NOT NULL,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255),
      Age int,
      City varchar(255) DEFAULT 'Sakarya'
  );
  ```

- ```
  CREATE TABLE Orders (
      ID int NOT NULL,
      OrderNumber int NOT NULL,
      OrderDate date DEFAULT GETDATE()
  );
  ```

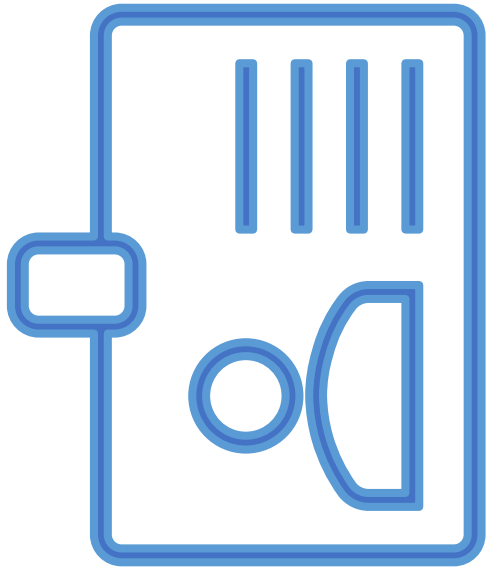
- ```
  ALTER TABLE Persons
  ADD CONSTRAINT df_City
  DEFAULT 'Sakarya' FOR City;
  ```

- ```
  ALTER TABLE Persons
  ALTER COLUMN City DROP DEFAULT;
  ```

IDENTITY

- Otomatik artış; bir tabloya yeni bir değer eklendiğinde otomatik olarak üretilecek benzersiz bir sayı sağlar. Genellikle bu değer yeni bir kayıt eklendiğinde otomatik olarak oluşturulmasını isteyeceğimiz Primary Key alanıdır

- CREATE TABLE Persons (
  ID int IDENTITY(1,1) PRIMARY KEY,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int
  );



- IDENTITY(10,5) yapsaydık ne olurdu?

- Yeni kayıt Insert edilirken artık bu değer belirtilmez VTYS tarafından otomatik eklenir.

DML Komutları

- **INSERT**
  - INSERT INTO; tabloya yeni kayıtlar eklerken kullanılır. 2 şekilde eklenebilir.
    - INSERT (INTO) *table_name*
      VALUES (*value1*, *value2*, *value3*, ...);
    - INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
      VALUES (*value1*, *value2*, *value3*, ...);
- Dikkat: tablonun tüm alanlarına ekleme yapılıyorsa sütun alanlarını belirtmeye gerek yoktur ancak sütün adlarının tablodaki ile aynı sırada eklendiğine dikkat edilmelidir.
- Örn: Personel Tablosuna Kayıt Ekleyiniz.
- (Hem Northwind hem yeni tablo üzerinde)

DML Komutları

- **UPDATE**

- Tablodaki verileri güncellemek için kullanılır. Tüm kayıtların veya sadece sadece belirtilen koşula (WHERE) uyan kayıtların güncellenmesi sağlanabilir.

- UPDATE *table_name*
  SET *column1* = *value1*, *column2* = *value2*, ...
  WHERE *condition*;

- **Not:** Tablodaki kayıtları güncellerken dikkatli olunmalıdır! UPDATE ifadesindeki WHERE öbeğine dikkat ediniz. WHERE çıkarılırsa tablodaki tüm kayıtlar güncellenecektir!

- Örn: Personel tablosundaki kişilerin maaşlarını güncelleyiniz.

DML Komutları

- **DELETE**
- Tablodan kayıtları silmek için kullanılır.

- DELETE FROM *table_name* WHERE *condition*;

- **Not:** Kayıtları silerken dikkatli olunuz! DELETE ifadesindeki WHERE öbeğine dikkat ediniz. WHERE çıkarılırsa tablodaki tüm kayıtlar silinecektir.
  - DELETE FROM *table_name*;
  - Tüm satırları siler. Tablo yapısı, özellikleri, indexler etkilenmeyecektir.

- Örn: Personel tablosundan belirli ID ye sahip müşteriyi siliniz.

TRUNCATE – DELETE

TRUNCATE	DELETE
DDL Komutu	DML Komutu
Kalıcı bir silmedir	Geri alınabilir bir silmedir.
Belirli bir kayıt silinemez	Belirli bir kayıt silinebilir
WHERE desteklenmez	WHERE desteklenir
Veri geri alınamaz	Veri geri alınabilir
IDENTITY resetlenir	IDENTITY resetlenmez

Kopyalama Örnekleri

- SELECT INTO bir tabloyu diğer bir tabloya kopyalar.
- SELECT *
  INTO *newtable* [IN *externaldb*]
  FROM *oldtable*
  WHERE *condition*;

- Yalnızca belirli alanları koypala;
- SELECT *column1*, *column2*, *column3*, ...
  INTO *newtable* [IN *externaldb*]
  FROM *oldtable*
  WHERE *condition;*

- *Yeni tablo, eski tabloda tanımlanan sütun alanları ve tipleriyle oluşturulacaktır. AS kullanılarak yeni sütun adları verilebilir.*

Kopyalama Örnekleri

- Customer tablosunun backup kopyasını oluşturalım.
  - SELECT * INTO CustomersBackup2019
    FROM Customers;
- Yalnızca belirli alanları kopyala;
  - SELECT CompanyName, ContactName INTO CustomersBackup2017
    FROM Customers;
- Yalnızca Alman müşterileri yeni tabloya kopyala
  - SELECT * INTO CustomersGermany
    FROM Customers
    WHERE Country = 'Germany';
- Birden fazla tablodaki verileri yeni bir tabloya kopyala;
  - SELECT Customers.CompanyName, Orders.OrderID
    INTO CustomersOrderBackup2017
    FROM Customers
    LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

Kopyalama Örnekleri

- INSERT INTO SELECT bir tablodan verileri kopyalar ve diğer bir tabloya ekler.
- Hedef ve kaynak tablo arasındaki veri tiplerinin eşleşmesi gerekir.
- Hedef tablodaki mevcut kayıtlar etkilenmez.

- Tüm alanları kopyala;

  - INSERT INTO *table2*
    SELECT * FROM *table1*
    WHERE *condition*;

- Belirli alanları kopyala;

  - INSERT INTO *table2* (*column1*, *column2*, *column3*, ...)
    SELECT *column1*, *column2*, *column3*, ...
    FROM *table1*
    WHERE *condition*;

Kopyalama Örnekleri

- Suppliers tablosundaki alanlar Customers tablosuna kopyalanacaktır. (veri içermeyen alanlar NULL alacaktır.)
  - INSERT INTO Customers (CompanyName, City, Country)
    SELECT CompanyName, City, Country FROM Suppliers;

- Tüm alanları doldur;
  - INSERT INTO Customers (CompanyName, ContactName, Address, City, PostalCode, Country)
    SELECT CompanyName, ContactName, Address, City, PostalCode, Country FROM Suppliers;

- Yalnızca Alman tedarikçileri al;
  - INSERT INTO Customers (CompanyName, City, Country)
    SELECT SupplierName, City, Country FROM Suppliers
    WHERE Country='Germany';

REFERANSLAR

- «SQL Server 2017» Abaküs yayınları – İsmail Adar
- Carlos Coronel, Steven Morris, and Peter Rob, Database Systems: Design, Implementation, and Management, Cengage Learning.
- BTK AKADEMİ – «Uygulamalarla SQL Öğreniyorum» - Ömer Çolakoğlu