

Batuhan Atabey – 170111006 – 2. Öğretim Bilgisayar Programcılığı 2.Dönem Veri Tabanı Ders İçeriklerim.

İçindekiler

<u>Veri Tabanı nedir ?</u>	1
<u>Veri Tabanı Yönetim Sistemleri</u>	1
<u>SQL Server ve Avantajları</u>	1
<u>DİE 02 - Normalizasyon nedir? Normalizasyon kuralları nelerdir?</u>	3
<u>1.Normalizasyon Kuralları Nelerdir?</u>	3
<u>1.1 Genel Normalizasyon Kuralları</u>	3
<u>1.2 Normalizasyon Kuralları :</u>	3
<u>ODEV 03 - MS-SQL dilinde kullanılan veri tipleri nelerdir? Örnek tablo yapıları ve veri girdileri kodlayınız.</u>	13
<u>DİE 03 - MS-SQL dilinde kullanılan veri tipleri nelerdir? Örnek tablo yapıları ve veri girdileri kodlayınız.</u>	20
<u>DİE 04 - MS-SQL dilinde DateTime Fonksyonlarının Kullanımını açıklayınız. Her birine 1'er adet örnek kodlayınız. DBCC CHECKIDENT kavramını açıklayınız ve her birine örnek kodlamalar gerçekleştiriniz.</u>	27
<u>DATETIME</u>	27
<u>DBC CHECKIDENT</u>	32
<u>DİE 05 - Şema ,index , view kavramları nedir ?</u>	34
<u>Şema(Schema) kavramı nedir? Açıklayınız.</u>	34
<u>Görünüm(View) kavramı nedir? Açıklayınız.</u>	38
<u>ÖDEV 05 - Saklı Yordam (Stored Procedur) kavramı nedir? Açıklayınız. Tetikleyici (Trigger) kavramı nedir? Açıklayınız.</u>	41
<u>Saklı Yordam (Stored Procedur):</u>	41
<u>Tetikleyici (Trigger) kavramı nedir?</u>	42
<u>DİE 06 – MERGE KOMUTUNUN KULLANIMI , TRANSACTION MİMARİSİ</u>	45
<u>MERGE KULLANIMI :</u>	45
<u>TRANSACTION MİMARİSİ</u>	46
<u>DİE 08 - "Veri tabanı kullanıcı, rolleri ve yetkileri nelerdir? Açıklayınız. Kendi veritabanınızda kullanıcı, rollerini ve yetkilerini oluşturunuz. (8 Adet) Veritabanı yedekini alma ve yedekten geri yükleme işlemlerini açıklayınız."</u>	49
<u>1) SQL Serverda yeni kullanıcı (User) Ekleme:</u>	49

<u>2) Server Roles Kullanıcıya rol verme</u>	52
<u>2.2. DataBase Role</u>	58
<u>2.2.1. Bu kuralların prosedür ile yönetilmesi:</u>	59
<u>3)YEDEK ALMA(Back Up)</u>	60 ⁴⁾
<u>DİE09 – SQL İnjeksiyon & Video Raporlama</u>	64
<u>Sql İnjeksiyon Nedir?</u>	64
<u>SQL İnjeksiyon Nasıl Çalıştırılır ?</u>	64
<u>SQL İnjeksiyon ile veri alma:</u>	64
<u>DİE 11. İstanbul Bilişim Kongresi Konu Başlıkları</u>	66
<u>Dijital Dönüşüm ve Verinin Önemi</u>	66
<u>Veri Madenciliği</u>	67
<u>Geleceğin Yeni Veri Otobanı ve İnterneti</u>	67
<u>Verinizin Değeri Güvenliği Kadardır</u>	67

DİE 01 - Veri Tabanı ve Veritabanı Yönetim Sistemleri,SQL SERVER ve Avantajları

Günümüzde yaşanan ve var olan bilgi çağının bize getirdiği bir getiri olan veri ve veri tabanı konuları hayatımıza girmiş olmakla beraber bazı gereksinimleri yanında getirdi. Bunlardan bazıları verileri sınıflandırmak , verileri depolamak ve saklamak olmasının yanı sıra bunları güvenli bir bölgede tutmamız için bize bazı ihtiyaçlar doğurdu. Bu ihtiyaçları ve tanımlarını birlikte inceleyelim.

Veri Tabanı nedir ?

Veri tabanının içinde yatan veriyi tanımlamamız gerekirse , insanların ihtiyaç duyduğu bilgileri saklamak için başvurduğu bir yapıdır.

Veri tabanın gereksinimi ise verileri düzenli bir şekilde depolanma ihtiyacından doğmuştur. Veri tabanı yönetilebilir , güncellenebilir , taşınabilir ve birbirleri arasında etkileşimi olan yapılardır. Bu özellikler bu yapıyı daha kullanabilir hale getirmek için gerekli olan mertyallerdir.

Veri Tabanı Yönetim Sistemleri

Kişisel bilgisayarların daha fazla yaygınlaşması sebebiyle veri tabanı sistemleri ortaya çıkmıştır. Veri tabanına en basit örnek olarak aslında günlük hayatımızda kullandığımız , okuldayken , işteyken veya herhangi bir yerdeyken aklimiza gelen fikirleri veya o an not alınması gereklili olan şeyleri kağıda dökmemiz günlük hayatımızda kullandığımız bir veri tabanı yönetim sistemi olarak algılayabiliriz. O kağıda verileri biz kendimiz saklayıp , silip , güncelleyebiliriz .Bunun bilgisayardaki karşılığına ise Not Defteri uygulamasını örnek verebiliriz.

Günümüzde ise bu sistemleri; devlet sistemlerinde , banka sistemlerinde , e ticaret sistemlerinde kullandığımızı görebiliriz. Bunun dışında bir çok veri kullanımının yapıldığı sistemlerde de rastlamak mümkündür.

SQL Server ve Avantajları

Sql server verileri depolamamızı , dizayn edip düzenlememizi sağlayan microsoftun çıkardığı bir programdır. Bu program 2005 yılında çıkış yapmış olup günümüzde kadar güncellemeler halinde büyüp yelpazesini genişletip gelmiştir. Günümüzde de web dizayn eden kişiler tarafından oldukça sevilen ve ilgi gören bir programdır.

Avantajları:

- Sql server diğer uygulamalarla kıyaslandığında daha uyumlu ve kullanışlı olduğu için insanlar genellikle sql server kullanmayı sefer.
- Microsoft programları ile daha çok entegre olduğu için kullanıbilirliği daha fazladır.
- Veriler birbirinden bağımsız şekilde tutulduğu için bir veride yapılan değişiklik diğer verileri etkilemez.
- Bir işlem yaparken güvenlik nedeniyle yetkisi dahilinde olmayan bir yere bir kişi giremez.
- Bütün veriler planlı tekrarlar dışında bir kez yer alır ve bu yüzdede veri tekrarı olmaz.

Dezavantajları:

- Yabancı kaynaklar için fazla avantaj sunmuyor.
- Performans için kullanabilir özellikler bazı fedalara neden olmakla beraber bazı sorunlarda getirebiliyor.
- İyi bir sql yazılımcısı olmayan kişinin yaptığı hatalar sonucu sistemde bazı açıklıklar meydana gelebilir. Bu açıklardan faydalananmaya çalışan insanlar içinde bu bir kapı niteliğindedir. Bu yüzden güvenilirlik açısından sağlam bir firma veya kişi ile çalışmamanız durumunda bilgileriniz sızabilir, çalınabilir.

- Uygulama ile veri arasında oluşan yazılım demeti boyutun fazla olmasına sebep olur.
- Sql serverin ücretli olmasıdır.
- Tüm sistem bir veri ağında veya bilgisayarda toplandığından dolayı bir problemden bütün ağı sarsar.

DİE 02 - Normalizasyon nedir? Normalizasyon kuralları nelerdir?

1.Normalizasyon Nedir ?

Veri karmaşasını ortadan kaldırmak için yapılan basitleştirme hareketlerine normalizyon denir.Normalizyon verileri sadeleştirmeyi amaçlamaktır.Bunun için birden çok veriyi kontrol edip daha sonrasında bu verileri analizleri sonucundan daha basit ve kullanılabilir hale getirmeye çalışmaktadır.

2.Normalizasyon Kuralları Nelerdir?

2.1 Genel Normalizasyon Kuralları

1. Veri Bütünlüğünü Sağlamak: Bir veri tabloasunda gereksiz veri tekrarı olan bölümleri varsa bunları ortadan kaldırmak için yaptığımız bir normalleştirme yani basite indirgeme işlemidir.
2. Tasarım bütünlüğü : Hazırlıyaçığımız tasarımın bütünlüğüne hitap eden bir veri ağı oluşturmalıyız.Ne biraz fazlasını nede biraz eksigini .

2.2 Normalizasyon Kuralları :

2.2.1. Normal Form (1 NF)

- Tekrarlanan sütün gruplarını ortadan kaldırmak için kullanılır.
- Birden fazla bilgi tek sutunda olamaz.

KODLAR:

YANLIŞ HALİ:

```
create database batuhanatabeydie02  
use batuhanatabeydie02
```

```
create table ogrenci_bilgiler  
(  
og_id int not null identity(1,1),  
og_no int primary key,  
og_ad nvarchar(50),  
og_soyad nvarchar(50),
```

```

arabamarkası nvarchar(max)
)

insert into ogrenci_bilgiler (og_no, og_ad, og_soyad, arabamarkası) VALUES
(1,'Batuhan','Atabey','Toyota , Hyundai'),
(2,'Serhat','Temel','BWM , Auodi'),
(3,'Barış','Hikmet','Mersedes, Honda'),
(4,'Turan','Dogru','Mersedes,BMW'),
(5,'Sercan','Avci','BMW,Toyota')

```

```
SELECT * FROM ogrenci_bilgiler
```

	og_id	og_no	og_ad	og_soyad	arabamarkası
1	1	1	Batuhan	Atabey	Toyota , Hyundai
2	2	2	Serhat	Temel	BWM , Auodi
3	3	3	Bans	Hikmet	Mersedes, Honda
4	4	4	Turan	Dogru	Mersedes,BMW
5	5	5	Sercan	Avci	BMW,Toyota

DOĞRU HALİ:

```
create database batuhanatabeydie02
use batuhanatabeydie02
```

```
create table ogrenci_bilgiler
(
og_id int not null identity(1,1),
og_no int primary key,
og_ad nvarchar(50),
og_soyad nvarchar(50),
arabamarkası nvarchar(max)
)
```

```
create table araca_bilgileri
()
```

```

ab_id int not null identity (1,1),
ab_no int primary key,
arabamarkası nvarchar(max)
)

```

```

insert into ogrenci_bilgiler (og_no, og_ad, og_soyad, arabamarkası) VALUES
(1,'Batuhan','Atabey','Toyota'),
(2,'Batuhan','Atabey','Hundai'),
(3,'Serhat','Temel','BWM'),
(4,'Serhat','Temel','Auodi'),
(5,'Barış','Hikmet','Mersedes'),
(6,'Barış','Hikmet','Honda'),
(7,'Turan','Dogru','Mersedes'),
(8,'Turan','Dogru','BMW'),
(9,'Sercan','Avcı','BMW') ,
(10,'Sercan','Avcı','Toyota')

```

```
SELECT * FROM ogrenci_bilgiler
```

	og_id	og_no	og_ad	og_soyad	arabamarkası
1	1	1	Batuhan	Atabey	Toyota
2	2	2	Batuhan	Atabey	Hundai
3	3	3	Serhat	Temel	BWM
4	4	4	Serhat	Temel	Auodi
5	5	5	Bans	Hikmet	Mersedes
6	6	6	Bans	Hikmet	Honda
7	7	7	Turan	Dogru	Mersedes
8	8	8	Turan	Dogru	BMW
9	9	9	Sercan	Avcı	BMW
10	10	10	Sercan	Avcı	Toyota

2.2.2 . Normal Form (2NF)

1. Tabloda bir birincil anahtar olmalıdır ve anahtar olmayan sütunlar birincil anahtara bağlı olmalıdır.

2. Örneğin bizim oluşturduğumuz tabloda öğrenci bilgilerimiz ve sahip oldukları araba çeşitleri aynı tabloda yer alıyordu fakat 2NF kurallarını göre bu tablolar birbirinden ayrı olmalıdır bu yüzden bu iki tabloyu ayrı ayrı kodlayıp yeniden yazalım .

Yukardaki tablonun N2 Kurallarına göre Yazdığımız Hali:

KODLAR:

```
create database batuhanatabeydie02
use batuhanatabeydie02
```

```
create table ogrenci_bilgiler
(
og_id int not null identity(1,1),
og_no int primary key,
og_ad nvarchar(50),
og_soyad nvarchar(50),
arabamarkası nvarchar(max)
)
```

```
create table arac_bilgileri
(
ab_id int not null identity (1,1),
ab_no int primary key,
arabamarkası nvarchar(max)
)
```

```
insert into ogrenci_bilgiler (og_no, og_ad, og_soyad) VALUES
(1,'Batuhan','Atabey'),
(2,'Serhat','Temel'),
(3,'Barış','Hikmet'),
```

```
(4, 'Turan', 'Dogru'),  
(5, 'Sercan', 'Avcı')
```

```
insert into arac_bilgileri (ab_no, arabamarkası) VALUES  
(1, 'Hundai'),  
(2, 'Mersedes'),  
(3, 'BWM'),  
(4, 'Honda'),  
(5, 'Toyota')
```

```
SELECT * FROM ogrenci_bilgiler  
SELECT * FROM arac_bilgileri
```

	og_id	og_no	og_ad	og_soyad
1	1	1	Batuhan	Atabey
2	2	2	Serhat	Temel
3	3	3	Bans	Hikmet
4	4	4	Turan	Dogru
5	5	5	Sercan	Avcı

	ab_id	ab_no	arabamarkası
1	1	1	Hundai
2	2	2	Mersedes
3	3	3	BWM
4	4	4	Honda
5	5	5	Toyota

Normal Form(3NF)

1.Bu bölümde tabloları en verimli şekilde bölmek bizim amacımız olacaktır.

2. Gereksiz veri karmaşasının önüne geçmek için bazı verileri başka tablolara kaydetmeliyiz
3. Örneğin bizim yaptığımız öğrencilerin araba tablosunda öğrenci bilgilerini , araç isimlerini ve araç üreticilerinin isimlerini 3 farklı ayrı tabloya bölmemiz gereklidir.
- 4.Faydalı ise istediğimiz tablodan , güncelleme ,silme , veya yeniden veri ekleme işimizi daha da kolaylaştırır. Bu da zamanda tasarruf etmemizi sağlar.

KOD ÖRNEKLERİ:

```

create database batuhanatabeydie02
use batuhanatabeydie02

create table ogrenci_bilgiler
(
og_id int not null identity(1,1),
og_no int primary key,
og_ad nvarchar(50),
og_soyad nvarchar(50),
arabamarkası nvarchar(max)
)

create table arac_bilgileri
(
ab_id int not null identity (1,1),
ab_no int primary key,
arabamarkası nvarchar(max)
)

create table arac_uretici_bilgileri
(
au_id int not null identity (1,1),
au_no int primary key,
arabamarkası nvarchar(max),
arabaureticiad nvarchar (max)
)

insert into ogrenci_bilgiler (og_no, og_ad, og_soyad) VALUES
(1,'Batuhan','Atabey'),
(2,'Serhat','Temel'),
(3,'Barış','Hikmet'),
(4,'Turan','Dogru'),
(5,'Sercan','Avcı')

insert into arac_bilgileri (ab_no, arabamarkası) VALUES
(1,'Hyundai'),
(2,'Mersedes'),

```

```

(3,'BWM'),
(4,'Honda'),
(5,'Toyota')

insert into arac_uretici_bilgileri (au_no, arabamarkası, arabaureticiad) VALUES
(1,'Hundai','Hasan'),
(2,'Mersedes','Mehmet'),
(3,'BWM','Necip'),
(4,'Honda','Sevda'),
(5,'Toyota','Zeki')

```

```

SELECT * FROM ogrenci_bilgiler
SELECT * FROM arac_bilgileri
SELECT * FROM arac_uretici_bilgileri

```

	og_id	og_no	og_ad	og_soyad
1	1	1	Batuhan	Atabey
2	2	2	Serhat	Temel
3	3	3	Bans	Hikmet
4	4	4	Turan	Dogru
5	5	5	Sercan	Avcı

	ab_id	ab_no	arabamarkası
1	1	1	Hundai
2	2	2	Mersedes
3	3	3	BWM
4	4	4	Honda
5	5	5	Toyota

	au_id	au_no	arabamarkası	arabaureticiad
1	1	1	Hundai	Hasan
2	2	2	Mersedes	Mehmet
3	3	3	BWM	Necip
4	4	4	Honda	Sevda
5	5	5	Toyota	Zeki

Normal Form (4NF)

1. Bu bölümde bölünmüş tablolar arasında 1nf bağlantısı ararız ve daha sonrasında bu tabloları birincil anahtarları bir olanları böläp farklı tablolar yapabiliriz.

2. Örneğin yukarıda yaptığımız öğrencilerin arabalarını ve arabaların üretici isimlerini gösterdiğimiz tabloda , artık araba notları aynı olan araba üreticilerini yani farklı arabaları üreten bir kişinin bağlantılı bir tabloyla gösterilmesi olabilir.

KODLAR:

```
create database batuhanatabeydie02  
use batuhanatabeydie02
```

```
create table ogrenci_bilgiler  
(  
og_id int not null identity(1,1),  
og_no int primary key,  
og_ad nvarchar(50),  
og_soyad nvarchar(50),  
arabamarkası nvarchar(max)  
)
```

```
create table arac_bilgileri  
(  
ab_id int not null identity (1,1),  
ab_no int primary key,  
arabamarkası nvarchar(max),  
arabaureticiad nvarchar (max)  
)
```

```
create table arac_uretici_bilgileri  
(  
au_id int not null identity (1,1),  
au_no int primary key,  
arabamarkası nvarchar(max),  
arabaureticiad nvarchar (max),  
arabaureticino int  
)
```

```
insert into ogrenci_bilgiler (og_no, og_ad, og_soyad) VALUES  
(1,'Batuhan','Atabey'),  
(2,'Serhat','Temel'),  
(3,'Baris','Hikmet'),  
(4,'Turan','Dogru'),  
(5,'Sercan','Avci')
```

```
insert into arac_bilgileri (ab_no, arabamarkası) VALUES  
(1,'Hundai'),  
(2,'Mersedes'),  
(3,'BMW'),  
(4,'Honda'),  
(5,'Toyota')
```

```

insert into arac_uretici_bilgileri (au_no, arabamarkasi,
arabaureticiad,arabaureticino) VALUES
(1,'Hundai','Hasan',2),
(2,'Mersedes','Hasan',2),
(3,'BWM','Necip',3),
(4,'Honda','Sevda',5),
(5,'Toyota','Zeki',5)

```

```

SELECT * FROM ogrenci_bilgiler
SELECT * FROM arac_bilgileri
SELECT * FROM arac_uretici_bilgileri

```

	og_id	og_no	og_ad	og_soyad
1	1	1	Batuhan	Atabey
2	2	2	Serhat	Temel
3	3	3	Bans	Hikmet
4	4	4	Turan	Dogru
5	5	5	Sercan	Avcı

	ab_id	ab_no	arabamarkasi
1	1	1	Hundai
2	2	2	Mersedes
3	3	3	BWM
4	4	4	Honda
5	5	5	Toyota

	au_id	au_no	arabamarkasi	arabaureticiad	arabaureticino
1	1	1	Hundai	Hasan	2
2	2	2	Mersedes	Hasan	2
3	3	3	BWM	Necip	3
4	4	4	Honda	Sevda	5
5	5	5	Toyota	Zeki	5

Resimde gördüğümüz gibi bazı üreticiler birden fazla araba ürettiği için bu arabaureticinoyu Değiştirmeden buradan kullanarak aralarında bir bağ oluşturabiliriz.

2.2.5 Normal Form (5NF)

1. Bu bölüm son forumdur .
2. Bu bölümde verilerin tekrarlanmasılığını önlemek amaçlı tedbirler alınır .

3. Verilerinin tekrarlanması hem yer açısından hem de bazı sorunları ortaya çıkarması açısından engellemek önemlidir.

ODEV 03 - MS-SQL dilinde kullanılan veri tipleri nelerdir? Örnek tablo yapıları ve veri girdileri kodlayınız.

1.İNT

Sayısal verileri tutmamızı sağlar. 4 byte büyüklüğünde yaklaşık -2 milyar ile +2 milyar arasında değer alabilen tamsayı veri tipidir.

KODLAR:

```
create table ogrencinumara(  
id int identity(1,1) primary key,  
ogrencino int,  
)  
insert into ogrencinumara values  
(170111006)
```

2.BİGİNT

Dosyaları data(hafıza bölümünde) saklamamıza yarar. 1-800 arası değerler alabilir. 8 Byte büyüklüğünde ve -2^63 ile +2^63 arasında değerler alır.

KODLAR:

```
create table kimlikbilgileri(  
id int identity(1,1) primary key,  
tcno bigint,  
)  
  
insert into kimlikbilgileri values  
(5486986884)
```

3. Varchar (50)

Karakter dizilerini tutmamızı sağlar. 1 ile 8000 arasında veri tutabilir.

KODLAR:

```
create table ogrenciismi(  
id int identity(1,1) primary key,  
ogrenciad nvarchar(50)  
)  
  
insert into ogrenciismi values  
('Batuhan Atabey ')
```

4. Varchar(max)

Varchardan farklı olarak **2,147,483,647** veri kayıt edebilir.

KODLAR:

```
create table hikaye(
```

```
id int identity(1,1) primary key,  
jamesfordunhikayesi varchar(max),  
)  
  
insert into hikaye values  
('James ford ilk basta iflas etmis daha sonrasında yeniden toparlanıp , yeni bir iş kurup başarılı olmuştur.')
```

5. Bit

Sıfır ve birlerden oluşur. Data bölümünde veriyi saklamamızı sağlar.

KODLAR:

```
create table cinsiyet(  
id int identity(1,1) primary key,  
cinsiyet bit,  
ModifiedDate date default (getdate())  
)  
  
insert into cinsiyet(cinsiyet) values (1);  
insert into cinsiyet(cinsiyet) values (0);
```

6. Smallint

Int ile aynı özellikleri taşır fakat daha az sayı seçmemizi sağlar.Daha az yer kaplar. -32.768 ile + 32.768 arası sayıları tutar.

KODLAR:

```
create table Kapinumarası(  
id int identity(1,1) primary key,  
kapinumarası smallint,  
)  
insert into kapinumarası values (252);
```

7.Char

Bütün karakterleri yazabildiğimiz bölümdür.Ekstra bir özelliği olarak char(20) yazdığımızda 20 değer girebiliriz.Parantez içine yazılan değer kadar girebiliceğimiz değer olabilir.

KODLAR:

```
create table Ulkeismi(  
id int identity(1,1) primary key,  
ulkeismi char,  
)  
insert into ulkeismi values ('İspanya');
```

8. Date time

Date ile aynı özelliklerini taşır ekstra olarak. Saat , dakika ve saniye verilerini de girebiliriz.

KODLAR:

```
create table urunalissaati(  
id int identity(1,1) primary key,  
urunalissaati datetime,  
mesaj varchar(256),  
modi datetime default(getdate())  
)
```

```
INSERT urunalissaati(mesaj) VALUES(  
'Batuhan Atabey, BWM. CSO 09-02-08 Buy 2,000 6.10')
```

9. Float

Ortalama değer almamızı sağlayan veri tipidir. Virgülden sonra göstermemizi kaç sayı istediğimizi belirtip o kadar sayıyı görüntülememizi sağlar.

KODLAR:

```
create table pisayisi(  
id int identity(1,1) primary key,  
pisayisi float,  
)  
insert into pisayisi values (3.14);
```

10. Nchar

Char ile aynı özellikleri taşıır ekstra olarak Unicode karakterler kullanmamızı sağlar.

KODLAR:

```
create table adressbilgisi(  
id int identity(1,1) primary key,  
adres nchar,  
)  
insert into adressbilgisi values (54465);
```

11. Money & 12. Smallmoney

Small Money: 4 Byte uzunluğundadır. -214 000 ile 214 000 arasında parasal değer tutmak için kullanılır.

Money: 8 Byte uzunluğundadır. -922 Milyar ile 922 Milyar arası sayı tutmamızı sağlar.

KODLAR:

```
create table nufusbilgisi(  
id int identity(1,1) primary key,  
ulkenufus money,  
sehirnufus smallmoney,  
)
```

```
insert into nufusbilgisi values (80000000,13000000);
```

13. Tinyint

1 Byte büyüklüğündedir. 0 ile 255 arasında değer alır.

KODLAR:

```
create table sokakno(
id int identity(1,1) primary key,
sokakno tinyint,
)
insert into sokakno values (253);
```

14. Geometry

Koordinat girmemizi sağlar. Örneğin X=50 , Y=35

KODLAR:

```
IF OBJECT_ID ( 'dbo.istkonum ', 'U' ) IS NOT NULL
    DROP TABLE dbo.istkonum;
GO
```

```
CREATE TABLE istkonum
( id int IDENTITY (1,1),
konum1 geometry,
konum2 AS konum1.STAsText() );
GO
```

```
INSERT INTO istkonum (konum1)
VALUES (geometry::STGeomFromText('FATİH (100 100, 20 180, 180 180)', 0));
```

```
INSERT INTO istkonum (konum1)
VALUES (geometry::STGeomFromText('AVCILAR ((0 0, 150 0, 150 150, 0 150, 0 0))', 0));
GO
```

```
select * from istkonum
```

15. Nvarchar

KODLAR:

```
create table fatihbolgesiokulismileri(
id int identity(1,1) primary key,
okulismi nvarchar,
)
```

```
insert into fatihbolgesiokulismileri values
('Fatih Meslek Yüksek Okulu'),
('Fatih Kız Meslek Lisesi'),
('Fatih Anadolu Lisesi')
```

16. Nvarchar(50)

Unicode verileri için geçerlidir. En fazla 4.000 karater kullanabiliriz. Buda 8.000 byte yer kaplar.

KODLAR:

```
create table fatihbolgesiokuladresi(  
id int identity(1,1) primary key,  
adresi nvarchar(50),  
)
```

```
insert into fatihbolgesiokuladresi values  
('İstanbul / Fatih / KocaMustafaPasa cad. No:23')
```

17. Nvarchar(max)

Nvarchardan farklı olarak 2^{31} Byte yer kaplar.

KODLAR:

```
create table fatihokuluozgecmisi(  
id int identity(1,1) primary key,  
ozgecmis nvarchar(max),  
)
```

```
insert into fatihokuluozgecmisi values  
('İstanbul Fatih Okulumuz 1905 yılında kurulmuş köklü bir kuruluşdur. Kuran kişi Fatih Pasadır')
```

18. Decimal

Virgülü bir sayının virgülde sonra kaç adet sayıyı girmemizi belirten veri tipidir.

KODLAR:

```
CREATE TABLE altınfiyatları  
(  
ceyrek decimal(5,2),  
tam numeric(10,5)  
);  
  
GO  
INSERT INTO altınfiyatları VALUES (123, 12345.12);  
GO  
SELECT ceyrek, tam  
FROM altınfiyatları
```

19. Double

Int ile aynı özellikleri taşır fakat daha fazla değer girmemizi sağlar.

KODLAR:

```
create table arabalar
(
id int identity(1,1) primary key,
genelarabalar double
)

insert into arabalar values
(500000000000)
```

20. Varbinary

KODLAR:

```
create table koynufussayimi(
id int identity(1,1) primary key,
koynufus varbinary,
)
```

```
insert into koynufussayimi values
(500000000000)
```

21. Varbinary(50)

Dosyaları veri tabanında kaydetmemizi sağlar.8000 Byte veri tutabilir.

KODLAR:

```
create table sehirnufussayimi(
id int identity(1,1) primary key,
sehirnufus varbinary(50),
)
```

```
insert into sehirnufussayimi values
(500000000000)
```

22. Varbinary(max)

Varbinary(50) den farklı olarak 2^{31} Byte yer kaplar.

KODLAR:

```
create table ulkenufussayimi(
id int identity(1,1) primary key,
koynufus varbinary(max),
)
```

```
insert into ulkenufussayimi values
(500000000000)
```

23. Image

Resim dosyalarını tutmamızı ve görüntülememizi sağlar.

KODLAR:

```
create table futbolsimge(
```

```
id int identity(1,1) primary key,  
galatasaray image,  
)
```

```
insert into futbolsimge values ('C:\Users\OGRENCI.MMF-LAB305-.000\Desktop')
```

24. Real

Bütün sayıları tutmamızı sağlar.

KODLAR:

```
create table dunyanufus(  
id int identity(1,1) primary key,  
dunyanufus image,  
)
```

```
insert into dunyanufus values (8560000000000)
```

25. Xml

Hiyerarşik verileri tutmamızı sağlar.

KODLAR:

```
DECLARE @myDoc xml  
DECLARE @ProdID int  
SET @myDoc = '<Root>  
<ProductDescription ProductID="1" ProductName="Road Bike">  
<Features>  
    <\Warranty>1 year parts and labor</Warranty>  
    <Maintenance>3 year parts and labor extended maintenance is available</Maintenance>  
</Features>  
</ProductDescription>  
</Root>'  
  
SET @ProdID = @myDoc.value('(/Root/ProductDescription/@ProductID)[1]', 'int' )  
SELECT @ProdID
```

DİE 03 - MS-SQL dilinde kullanılan veri tipleri nelerdir? Örnek tablo yapıları ve veri girdileri kodlayınız.

1.İNT

Sayısal verileri tutmamızı sağlar. 4 byte büyüklüğünde yaklaşık -2 milyar ile +2 milyar arasında değer alabilen tamsayı veri tipidir.

KODLAR:

```
create table ogrencinumara(  
id int identity(1,1) primary key,  
ogrencino int,  
)  
insert into ogrencinumara values  
(170111006)
```

2.BİGİNT

Dosyaları data(hafıza bölümünde) saklamamıza yarar. 1-800 arası değerler alabilir. 8 Byte büyüklüğünde ve -2^63 ile +2^63 arasında değerler alır.

KODLAR:

```
create table kimlikbilgileri(  
id int identity(1,1) primary key,  
tcno bigint,  
)  
  
insert into kimlikbilgileri values  
(5486986884)
```

3. Varchar (50)

Karakter dizilerini tutmamızı sağlar. 1 ile 8000 arasında veri tutabilir.

KODLAR:

```
create table ogrenciismi(  
id int identity(1,1) primary key,  
ogrenciad nvarchar(50)  
)  
  
insert into ogrenciismi values  
('Batuhan Atabey ')
```

4. Varchar(max)

Varchardan farklı olarak **2,147,483,647** veri kayıt edebilir.

KODLAR:

```
create table hikaye(  
id int identity(1,1) primary key,  
jamesfordunhikayesi varchar(max),
```

)

```
insert into hikaye values  
('James ford ilk basta iflas etmis daha sonrasında yeniden toparlanıp , yeni bir iş kurup başarılı  
olmuştur.')
```

5. Bit

Sıfır ve birlerden oluşur. Data bölümünde veriyi saklamamızı sağlar.

KODLAR:

```
create table cinsiyet(  
id int identity(1,1) primary key,  
cinsiyet bit,  
ModifiedDate date default (getdate())  
)
```

```
insert into cinsiyet(cinsiyet) values (1);  
insert into cinsiyet(cinsiyet) values (0);
```

6. Smallint

**İnt ile aynı özellikleri taşıır fakat daha az sayı seçmemizi sağlar.Daha az yer kaplar. -32.768
ile + 32.768 arası sayıları tutar.**

KODLAR:

```
create table Kapınumarası(  
id int identity(1,1) primary key,  
kapınumarası smallint,  
)  
insert into kapınumarası values (252);
```

7.Char

**Bütün karakterleri yazabildiğimiz bölümdür.Ekstra bir özelliği olarak char(20)
yazdığımızda 20 değer girebiliriz.Parantez içine yazılan değer kadar girebiliceğimiz değer olabilir.**

KODLAR:

```
create table Ulkeismi(  
id int identity(1,1) primary key,  
ulkeismi char,  
)  
insert into ulkeismi values ('İspanya');
```

8. Date time

Date ile aynı özelliklerini taşır ekstra olarak. Saat , dakika ve saniye verilerini de girebiliriz.

KODLAR:

```
create table urunalissaati(  
id int identity(1,1) primary key,  
urunalissaati datetime,  
mesaj varchar(256),  
modi datetime default(getdate())  
)
```

```
INSERT urunalissaati(mesaj) VALUES(  
'Batuhan Atabey, BWM. CSO 09-02-08 Buy 2,000 6.10')
```

9. Float

Ortalama değer almamızı sağlayan veri tipidir. Virgülden sonra göstermemizi kaç sayı istediğimizi belirtip o kadar sayıyı görüntülememizi sağlar.

KODLAR:

```
create table pisayisi(  
id int identity(1,1) primary key,  
pisayisi float,  
)  
insert into pisayisi values (3.14);
```

10. Nchar

Char ile aynı özellikleri taşıır ekstra olarak Unicode karakterler kullanmamızı sağlar.

KODLAR:

```
create table adressbilgisi(  
id int identity(1,1) primary key,  
adres nchar,  
)  
insert into adressbilgisi values (54465);
```

11. Money & 12. Smallmoney

Small Money: 4 Byte uzunluğundadır. -214 000 ile 214 000 arasında parasal değer tutmak için kullanılır.

Money: 8 Byte uzunluğundadır. -922 Milyar ile 922 Milyar arası sayı tutmamızı sağlar.

KODLAR:

```
create table nufusbilgisi(  
id int identity(1,1) primary key,  
ulkenufus money,  
sehirnufus smallmoney,  
)  
insert into nufusbilgisi values (80000000,13000000);
```

13. Tinyint

1 Byte büyüklüğündedir. 0 ile 255 arasında değer alır.

KODLAR:

```
create table sokakno(
id int identity(1,1) primary key,
sokakno tinyint,
)
insert into sokakno values (253);
```

14. Geometry

Koordinat girmemizi sağlar. Örneğin X=50 , Y=35

KODLAR:

```
IF OBJECT_ID ( 'dbo.istkonum ', 'U' ) IS NOT NULL
DROP TABLE dbo.istkonum;
GO
```

```
CREATE TABLE istkonum
( id int IDENTITY (1,1),
konum1 geometry,
konum2 AS konum1.STAsText() );
GO
```

```
INSERT INTO istkonum (konum1)
VALUES (geometry::STGeomFromText('FATİH (100 100, 20 180, 180 180)', 0));
```

```
INSERT INTO istkonum (konum1)
VALUES (geometry::STGeomFromText('AVCILAR ((0 0, 150 0, 150 150, 0 150, 0 0))', 0));
GO
```

```
select * from istkonum
```

15. Nvarchar

KODLAR:

```
create table fatihbolgesiokulisimleri(
id int identity(1,1) primary key,
okulismi nvarchar,
)
```

```
insert into fatihbolgesiokulisimleri values
('Fatih Meslek Yüksek Okulu'),
('Fatih Kız Meslek Lisesi'),
('Fatih Anadolu Lisesi')
```

16. Nvarchar(50)

Unicode verileri için geçerlidir. En fazla 4.000 karakter kullanabiliriz. Bu da 8.000 byte yer kaplar.

KODLAR:

```
create table fatihbolgesiokuladresi(  
id int identity(1,1) primary key,  
adresi nvarchar(50),  
)
```

```
insert into fatihbolgesiokuladresi values  
('İstanbul / Fatih / KocaMustafaPasa cad. No:23')
```

17. Nvarchar(max)

Nvarchardan farklı olarak 2^{31} Byte yer kaplar.

KODLAR:

```
create table fatihokuluozgecmisi(  
id int identity(1,1) primary key,  
ozgecmis nvarchar(max),  
)
```

```
insert into fatihokuluozgecmisi values  
('İstanbul Fatih Okulumuz 1905 yılında kurulmuş köklü bir kuruluşdur. Kuran kişi Fatih Pasadır')
```

18. Decimal

Virgülü bir sayının virgülde sonra kaç adet sayıyı girmemizi belirten veri tipidir.

KODLAR:

```
CREATE TABLE altınfiyatları  
(  
ceyrek decimal(5,2),  
tam numeric(10,5)  
);  
  
GO  
INSERT INTO altınfiyatları VALUES (123, 12345.12);  
GO  
SELECT ceyrek, tam  
FROM altınfiyatları
```

19. Double

Int ile aynı özellikleri taşır fakat daha fazla değer girmemizi sağlar.

KODLAR:

```
create table arabalar
```

```
(  
id int identity(1,1) primary key,  
genelarabalar double  
)
```

```
insert into arabalar values  
(500000000000)
```

20. Varbinary

KODLAR:

```
create table koynufussayimi(  
id int identity(1,1) primary key,  
koynufus varbinary,  
)
```

```
insert into koynufussayimi values  
(500000000000)
```

21. Varbinary(50)

Dosyaları veri tabanında kaydetmemizi sağlar.8000 Byte veri tutabilir.

KODLAR:

```
create table sehirnufussayimi(  
id int identity(1,1) primary key,  
sehirnufus varbinary(50),  
)
```

```
insert into sehirnufussayimi values  
(500000000000)
```

22. Varbinary(max)

Varbinary(50) den farklı olarak 2^{31} Byte yer kaplar.

KODLAR:

```
create table ulkenufussayimi(  
id int identity(1,1) primary key,  
koynufus varbinary(max),  
)
```

```
insert into ulkenufussayimi values  
(500000000000)
```

23. Image

Resim dosyalarını tutmamızı ve görüntülememizi sağlar.

KODLAR:

```
create table futbolsimge(  
id int identity(1,1) primary key,
```

```
galatasaray image,  
)  
insert into futbolsimge values ('C:\Users\OGRENCI.MMF-LAB305-.000\Desktop')
```

24. Real

Bütün sayıları tutmamızı sağlar.

KODLAR:

```
create table dunyanufus(  
id int identity(1,1) primary key,  
dunyanufus image,  
)  
insert into dunyanufus values (8560000000000)
```

25. Xml

Hiyerarşik verileri tutmamızı sağlar.

KODLAR:

```
DECLARE @myDoc xml  
DECLARE @ProdID int  
SET @myDoc = '<Root>  
<ProductDescription ProductID="1" ProductName="Road Bike">  
<Features>  
    <Warranty>1 year parts and labor</Warranty>  
    <Maintenance>3 year parts and labor extended maintenance is available</Maintenance>  
</Features>  
</ProductDescription>  
</Root>'  
  
SET @ProdID = @myDoc.value('(/Root/ProductDescription/@ProductID)[1]', 'int' )  
SELECT @ProdID
```

DİE 04 - MS-SQL dilinde DateTime Fonksiyonlarının Kullanımını açıklayınız. Her birine 1'er adet örnek kodlayınız. DBCC CHECKIDENT kavramını açıklayınız ve her birine örnek kodlamalar gerçekleştiriniz.

DATETIME

Datetime sql dilinde zaman fonksiyonlarını kullanabildiğimiz bir gereksinimdir. Veriler içerisinde yıl, ay, gün, saat vb gibi zaman kavramlarını ekleyip, güncelleştirmemizi sağlar.

Datetime kullanım alanları olarak zaman gereksinimleri olan bölümleri örnek gösterebiliriz. Mesala bir araç üretim firmasının ne zaman araçları satığını ve ne zaman teslim edeceğini gösteren bir veri tabanı uygulaması yapabiliriz.

Datetime Fonksiyonları:

GETDATE: Sistemin o anki mevcut bilgilerini ekrana yazdırın fonksiyondur.

Select getdate() Şeklinde kullanabiliriz.

```
create table aracbilgiler
(
    ID int,
    ureticifirma nvarchar (50),
    aracmodeli nvarchar (50),
    aracalimtarihi DATETIME
)

insert into aracbilgiler(ID,ureticifirma,aracmodeli,aracalimtarihi) values (1,'BWM x7','X7 Jeep',GETDATE())
insert into aracbilgiler(ID,ureticifirma,aracmodeli,aracalimtarihi) values (2,'BWM x8','Businnes',GETDATE())
insert into aracbilgiler(ID,ureticifirma,aracmodeli,aracalimtarihi) values (3,'BWM x9','2012 Sports',GETDATE())
```

Select* from aracbilgiler

	ID	ureticifirma	aracmodeli	aracalimtarihi
1	1	BWM x7	X7 Jeep	2018-10-17 19:17:50.780
2	2	BWM x8	Businnes	2018-10-17 19:17:50.780
3	3	BWM x9	2012 Sports	2018-10-17 19:17:50.780

DATEDD: Tarihlerde gün, ay, yıl vb. gibi saat ayarlamaları eklememizi sağlar.

Yaptığımız örnekden yolla çıkararak örneğin bir aracın teslim tarihini girmemiz gerekiyorsa. Bu fonksiyondan faydalananabiliriz.

```
create table aracbilgiler
(
    ID int,
    ureticifirma nvarchar (50),
    aracmodeli nvarchar (50),
    aracalimtarihi DATETIME
)
```

ID int,

```

ureticifirma nvarchar (50),
aracmodeli nvarchar (50),
alimtarihi DATETIME,
teslimtarihi DATETIME
)

insert into aracbilgiler(ID,ureticifirma,aracmodeli,teslimtarihi) values (1,'BWM
x7','X7 Jeep',DATEADD(DAY,20,GETDATE()))
insert into aracbilgiler(ID,ureticifirma,aracmodeli,teslimtarihi) values (2,'BWM
x8','Businnes',,DATEADD(MONTH,10,GETDATE()))
insert into aracbilgiler(ID,ureticifirma,aracmodeli,teslimtarihi) values (3,'BWM
x9','2012 Sports',DATEADD(YEAR,2018,GETDATE()))

```

4	2018-11-06 19:32:55.130
5	2019-08-17 19:32:55.130
6	4036-10-17 19:32:55.130

DATEPART: Tarih bilgilerinin istediğimiz bölümünü ekrana yazdırımızı sağlar.Buda bir ürünün teslim tarihini yoksa ürünün aldığımız tarihimi göstermemize yarar.

```

create table aracbilgiler
(
ID int,
ureticifirma nvarchar (50),
aracmodeli nvarchar (50),
alimtarihi DATETIME,
teslimtarihi DATETIME
)

insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (1,'BWM x7','X7
Jeep',DATEPART(DAY,GETDATE()))
insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (2,'BWM
x8','Businnes',DATEPART(MONTH,GETDATE()))
insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (3,'BWM
x9','2012 Sports',DATEPART(YEAR,GETDATE()))

select alimtarihi from aracbilgiler

```

7	1900-01-18 00:00:00.000
8	1900-01-11 00:00:00.000
9	1905-07-12 00:00:00.000

DATEIFF:İki zamanın farkını göstermemize yarar.Örneğin aracı aldığımız süreyle teslim arasında ne kadar olduğunu öğrenmek için bu fonksiyonu kullanabiliriz.Buda bizim aracın teslimine ne kadar kaldığını görmemizi sağlar.

```
create table aracbilgiler
(
ID int,
ureticifirma nvarchar (50),
aracmodeli nvarchar (50),
alimtarihi DATETIME,
teslimtarihi DATETIME
)

insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (1,'BWM x7','X7
Jeep',GETDATE())
insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (2,'BWM
x8','Businnes',GETDATE())
insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (3,'BWM
x9','2012 Sports',GETDATE())

SELECT DATEDIFF(month,alimtarihi,teslimtarihi) AS teslimenekadarvar From aracbilgiler
select TODATETIMEOFFSET(SYSDATETIMEOFFSET(),'-05:00')
```

	(No column name)
1	2018-10-17 19:51:14.3553816 -05:00

DAY:İstediğimiz bölümün sadace ay bilgilerini gösterir.Örneğimiz üzerinden gidersek aldığımız araç hangi ayda teslim ediliceğini görebilmemizi sağlayan fonksiyondur.

```
create table aracbilgiler
(
ID int,
ureticifirma nvarchar (50),
aracmodeli nvarchar (50),
alimtarihi DATETIME,
teslimtarihi DATETIME
)

insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (1,'BWM x7','X7
Jeep',GETDATE())
insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (2,'BWM
x8','Businnes',GETDATE())
```

```
insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (3,'BWM  
x9','2012 Sports',GETDATE())
```

```
insert into aracbilgiler(ID,ureticifirma,aracmodeli,teslimtarihi) values (1,'BWM  
x7','X7 Jeep',DATEADD(DAY,20,GETDATE()))  
select day(getdate())
```

(No column name)
17

MONTH:

```
insert into aracbilgiler(ID,ureticifirma,aracmodeli,teslimtarihi) values (1,'BWM  
x7','X7 Jeep',DATEADD(MONTH,20,GETDATE()))  
select MONTH(getdate())
```

(No column name)
10

YEAR:

```
insert into aracbilgiler(ID,ureticifirma,aracmodeli,teslimtarihi) values (1,'BWM  
x7','X7 Jeep',DATEADD(YEAR,20,GETDATE()))  
  
select YEAR(getdate())
```

(No column name)
1 2018

SYSDATETIME:Bu değer bölgesel saat farkı göstermeden döner.Yani herkes için tek bir zaman dilimini farz alır.

select SYSdatetime() Şeklinde kullanılır.

```
2018-10-17 20:08:38.0812190
```

TODATETIMEOFFSET:Herhangi bir zaman değerini koruyarak başka bir zaman dilimi eklemeye yarayan fonksiyondur.

```
create table aracbilgiler
(
    ID int,
    ureticifirma nvarchar (50),
    aracmodeli nvarchar (50),
    alimtarihi DATETIME,
    teslimtarihi DATETIME,
    teslimenekadarvar DATETIME
)

insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (1,'BWM x7','X7
Jeep',GETDATE())
insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (2,'BWM
x8','Businnes',GETDATE())
insert into aracbilgiler(ID,ureticifirma,aracmodeli,alimtarihi) values (3,'BWM
x9','2012 Sports',GETDATE())

select TODATETIMEOFFSET(SYSDATETIMEOFFSET(),'-05:00')
```

```
(No column name)
```

```
2018-10-17 20:11:55.5239415 -05:00
```

DBC CHECKIDENT

DBCC CHECKIDENT kavramını açıklayınız ve her birine örnek kodlamalar gerçekleştiriniz.

Üzerinde IDENTITY alanı bulunan bir tabloda o alandaki sayılar verilmiş ritmik değere göre ardışık gider.Her yeni kayıtta bu sayı artar.Bu kayıtlardan biri silindiği

Zaman o satırda ait identity değeri bir daha kullanılmaz.Dolayısıyla bir süre sonra silmelerden dolayı bu sayı dizisinde çok fazla atalamar söz konusu olur.Bazı durumlarda bu alanı baştan numaralandırma durumumuz olabilir.

```
create table checkident
(
    id int primary key identity(1,1),
    firmasaati datetime default(getdate()),
    satisveteslimfarkı int,
    teslimtarihiiekle datetime,
    yillar int,
)

insert into checkident (satisveteslimfarkı,teslimtarihiiekle,yillar)
values(DATEDIFF(YEAR,'2010-02-01',GETDATE()),DATEADD(DAY,14,'2018-10-01'),DATEPART(YEAR,'2020-10-01'))

insert into checkident (satisveteslimfarkı,teslimtarihiiekle,yillar)
values(DATEDIFF(YEAR,'2010-02-01',GETDATE()),DATEADD(DAY,14,'2018-10-01'),DATEPART(YEAR,'2020-10-01'))

insert into checkident (satisveteslimfarkı,teslimtarihiiekle,yillar)
values(DATEDIFF(YEAR,'2010-02-01',GETDATE()),DATEADD(DAY,14,'2018-10-01'),DATEPART(YEAR,'2020-10-01'))

insert into checkident (satisveteslimfarkı,teslimtarihiiekle,yillar)
values(DATEDIFF(YEAR,'2010-02-01',GETDATE()),DATEADD(DAY,14,'2018-10-01'),DATEPART(YEAR,'2020-10-01'))

insert into checkident(satisveteslimfarkı,teslimtarihiiekle,yillar)
values(DATEDIFF(YEAR,'2010-02-01',GETDATE()),DATEADD(DAY,14,'2018-10-01'),DATEPART(YEAR,'2020-10-01'))

select*from checkident
```

1	1	2018-10-17 20:33:51.060	8	2018-10-15 00:00:00.000	2020
2	2	2018-10-17 20:33:51.060	8	2018-10-15 00:00:00.000	2020
3	3	2018-10-17 20:33:51.060	8	2018-10-15 00:00:00.000	2020
4	4	2018-10-17 20:33:51.060	8	2018-10-15 00:00:00.000	2020
5	5	2018-10-17 20:33:51.060	8	2018-10-15 00:00:00.000	2020

DELETE KOMUTUNU KULLANARAK O İD (IDENTITY) Oları silebiliriz sadece onu sileriz.

```
DELETE FROM checkident WHERE id=4
```

1	1	2018-10-17 20:33:51.060	8	2018-10-15 00:00:00.000	2020
2	2	2018-10-17 20:33:51.060	8	2018-10-15 00:00:00.000	2020
3	3	2018-10-17 20:33:51.060	8	2018-10-15 00:00:00.000	2020
4	5	2018-10-17 20:33:51.060	8	2018-10-15 00:00:00.000	2020

RESETLEMEMİZİ SAĞLAYAN FONKSİYON

DBCC CHECKIDENT (checkident, RESEED,3)

```
Messages
Checking identity information: current identity value '15'.
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
```

DBCC CHECKDB('database adı'): Veritabanının yapısal durumunu kontrol etmek amacıyla kullanılır.

DBCC CHECKDB('database adı',REPAIR_REBUILD): Bozulan veri tabanını onarır. Veri kaybı oluşmaz.

DBCC SHOWCONTING: Bu komut veritabanındaki tabloların indeks parçalanma bilgisi gösterir.

DBC REINDEX: Bu komut veri tabanındaki tabloya bağlı indeksleri yeniden oluşturur.

DBCC CHECKIDENT (checkident, RESEED,3)

```
insert into checkident (satisveteslimfarkı,teslimtarihiEkle,yillar)
values(DATEDIFF(YEAR,'2010-02-01',GETDATE()),DATEADD(DAY,14,'2018-10-01'),DATEPART(YEAR,'2020-10-01'))
select*from checkident
```

DBCC CHECKIDENT (checkident, RESEED)

```
insert into checkident (satisveteslimfarkı,teslimtarihiEkle,yillar)
values(DATEDIFF(YEAR,'2010-02-01',GETDATE()),DATEADD(DAY,14,'2018-10-01'),DATEPART(YEAR,'2020-10-01'))
```

select*from checkident

DBCC CHECKIDENT (checkidentkullanimı, NORESEED)

DİE 05 - Şema ,index , view kavramları nedir ?

Şema(Schema) kavramı nedir? Açıklayınız.

Şema(Schema) Nedir?

Veri tabanının alt kapsayıcısı olarak tanımlayabiliriz. Büyük yazılımlar birden fazla objeyi bir arada tutup ,yönetmeyi gerektirir. Örneğin bir fabrikanın ürünlerinin stokları çok fazla boyutlu ve rakamlı verilerle doludur. Objelerin sayıları fazlalığı veri tabanını yönetmemizi zorlaştırır. Bu yüzden şema kavramına ihtiyaç duyuyoruz. Örneğin SQL server da dbo scheması default değerdedir bir değişiklik yapılmadığı takdirde dbo şemasına bağlı olarak oluşturulur.

- **KULLANIMI:**

Create Schema Authorization kullancı_ismi [şema_ögesi [...]]

Create Schema şema_ismi

[Authorization kullancı_ismi] [şema_ögesi[...]]

- **PARAMETRELER:**

şema_ismi

Oluşturulacak şemaya isim vermemizi sağlar.

Kullancı_ismi

Şemanın sahibi olacak ismi belirlemizi sağlar.

Şema_ögesi

Şemanın içindeki öğeleri seçmemizi sağlar ve o öğeyi tanımlamamızı sağlar.

- **UYUMLULUK:**

PostgreSQL'in de kabul ettiği SQL standart “CREATE SCHEMA” içinde “DEFAULT CHARACTER SET” deyimine izin verir.

- **KODLAR:**

```
Create Schema felsefe  
go
```

```
create table felsefe.ogrenci  
(  
ogrID int primary key identity(1,1),  
ogrNo int,  
ogrAd nvarchar(50),  
ogrSoyad nvarchar(50),  
)
```

```
create table felsefe.hoca  
(
```

```

hocaID int primary key identity(1,1),
hocaNo int,
hocaAd nvarchar(50),
hocaSoyad nvarchar(50),

)

create table felsefe.ders
(
dersID int primary key identity(1,1),
dersNo int,
dersIsim nvarchar(50),

)

create table felsefe.notlar
(
notID int primary key identity(1,1),
ogrID int,
dersID int,
hocaID int,
Vize int,
Final int,
)

insert into felsefe.ogrenci(ogrAd,ogrSoyad,ogrNo)
values('Cem','Pusat',170111006)

insert into felsefe.ders(dersIsim,dersNo)
values('Felsefe',2)

insert into felsefe.hoca(hocaAd,hocaSoyad,hocaNo)
values('Mustafa','Atabey',1)

insert into felsefe.notlar(ogrID,dersID,hocaID,Vize,Final)
values(170111006,2,1,60,80)

```

```

Create Schema bilgisayar
go

create table bilgisayar.ogrenci
(
ogrID int primary key identity(1,1),
ogrNo int,
ogrAd nvarchar(50),
ogrSoyad nvarchar(50),
)

create table bilgisayar.hoca
(
hocaID int primary key identity(1,1),
hocaNo int,
hocaAd nvarchar(50),
hocaSoyad nvarchar(50),
)
```

```
create table bilgisayar.ders
(
dersID int primary key identity(1,1),
dersNo int,
dersIsim nvarchar(50),

)

create table bilgisayar.notlar
(
notID int primary key identity(1,1),
ogrID int,
dersID int,
hocaID int,
Vize int,
Final int,
)

insert into bilgisayar.ogrenci(ogrAd,ogrSoyad,ogrNo)
values('Batuhan','Atabey',170111006)

insert into bilgisayar.ders(dersIsim,dersNo)
values('Bilgisayar',2)

insert into bilgisayar.hoca(hocaAd,hocaSoyad,hocaNo)
values('Ferhat','Perçin',1)

insert into bilgisayar.notlar(ogrID,dersID,hocaID,Vize,Final)
values(170111006,2,1,60,80)

Create Schema hukuk
go

create table hukuk.ogrenci
(
ogrID int primary key identity(1,1),
ogrNo int,
ogrAd nvarchar(50),
ogrSoyad nvarchar(50),
)

create table hukuk.hoca
(
hocaID int primary key identity(1,1),
hocaNo int,
hocaAd nvarchar(50),
hocaSoyad nvarchar(50),
)

create table hukuk.ders
(
dersID int primary key identity(1,1),
dersNo int,
dersIsim nvarchar(50),
)

create table hukuk.notlar
(
notID int primary key identity(1,1),
ogrID int,
```

```
dersID int,
hocaID int,
Vize int,
Final int,
)

insert into hukuk.ogrenci(ogrAd,ogrSoyad,ogrNo)
values('Mustafa','Aktaş',120)

insert into hukuk.ders(dersIsim,dersNo)
values('Hukuk',2)

insert into hukuk.hoca(hocaAd,hocaSoyad,hocaNo)
values('Murat','Taş',11)

insert into hukuk.notlar(ogrID,dersID,hocaID,Vize,Final)
values(120,2,11,60,50)

Create Schema tarih
go

create table tarih.ogrenci
(
ogrID int primary key identity(1,1),
ogrNo int,
ogrAd nvarchar(50),
ogrSoyad nvarchar(50),
)

create table tarih.hoca
(
hocaID int primary key identity(1,1),
hocaNo int,
hocaAd nvarchar(50),
hocaSoyad nvarchar(50),
)

create table tarih.ders
(
dersID int primary key identity(1,1),
dersNo int,
dersIsim nvarchar(50),
)

create table tarih.notlar
(
notID int primary key identity(1,1),
ogrID int,
dersID int,
hocaID int,
Vize int,
Final int,
)

insert into tarih.ogrenci(ogrAd,ogrSoyad,ogrNo)
values('Yekta','Pusat',110)

insert into tarih.ders(dersIsim,dersNo)
values('Tarih',2)
```

```

insert into tarih.hoca(hocaAd,hocaSoyad,hocaNo)
values('Burak','Temel',11)

insert into tarih.notlar(ogrID,dersID,hocaID,Vize,Final)
values(110,2,11,40,50)

```

`select * from tarih.`

The diagram shows a view named "batuha" represented by a yellow-bordered box. Inside the box, there are four entries: "ders", "hoca", "notlar", and "ogrenci", each preceded by a small grid icon representing a table.

Select * from tarih.ders

	dersID	dersNo	dersIsim
1	1	2	Tarih

Select * from tarih.hoca

	hocaID	hocaNo	hocaAd	hocaSoyad
1	1	11	Burak	Temel

Select * from tarih.notlar

	notID	ogrID	dersID	hocaID	Vize	Final
1	1	110	2	11	40	50

Select * from tarih.ogrenci

	ogrID	ogrNo	ogrAd	ogrSoyad
1	1	110	Yekta	Pusat

Görünüm(View) kavramı nedir? Açıklayınız.

SQL de , View temel olarak sanal tablolar oluşturup onlar üzerinden işlem yapmamızı sağlar.

Viewler dosyaları saklamaz sadace istendiğinde veriye ulaşıp onu bize sunar bu yüzden daha performansı elverişlidir.

- CREATE VIEW:

View oluştururmamızı sağlar.

KODLAR:

```
Create View FelsefeBolumu  
as  
Select ogrAd,ogrSoyad,ogrNo  
from felsefe.ogrenci  
  
select * from FelsefeBolumu
```

	ogrAd	ogrSoyad	ogrNo
1	Cem	Pusat	170111006

- ALTER VIEW:

Yeniden tablo eklememizi sağlar.

Örneğin numarası 10dan büyük öğrencileri gösteren tablo kodlaması aşağıdaki gibidir.

```
ALTER VIEW numarası10danbuyukogrencileribul
```

```
AS
```

```
SELECT ogrAd, ogrNo FROM Bilgisayar.ogrenci WHERE ogrNo>17
```

	ogrAd	ogrNo
1	Batuhan	170111006

- DROP VIEW:

View dosyalarını silmemizi sağlar.

```
drop view numarası10danbuyukogrencileribul
```

- INNER JOIN YAPISI KULLANIMI :

```
Create View BilgisayarBolumu2  
as  
Select ogrAd,ogrSoyad,ogrNo,dersIsim,dersNo,hocaNo,hocaAd,hocaSoyad,notID,Vize,Final  
from bilgisayar.notlar as n  
join bilgisayar.ogrenci as o on n.ogrID=o.ogrID  
join bilgisayar.hoca as h on h.hocaID=n.hocaID  
join bilgisayar.ders as d on d.dersID=n.dersID  
  
select * from BilgisayarBolumu2
```

Index kavramı nedir ? Açıklayınız.

Veri tabanındaki sorguları daha performanslı getirmemizi sağlar. Buda bu komutu kullanışlı bir hale getirir.

İndexler ikiye ayrılır :

1. Clustered(Kümelenmiş) Index:

İndexin verildiği değeri kolon değerine göre otomatik sıralanır.

```
Create table ogr
(
ogrNo int,
ogrAd nvarchar(50),
ogrSoyad nvarchar(50)
)

create clustered index ix_ogr_Ad
on ogr (ogrAd)
```

2. Nonclustered(Kümelenmemiş) Index:

İndexin verildiği değeri kolon değerine göre sıralamaz.

```
create nonclustered index ix_No
on ogr (ogrNo)

create nonclustered index ix_Soyad
on ogr(ogrSoyad)
```

**ÖDEV 05 - Saklı Yordam (Stored Procedur) kavramı nedir? Açıklayınız.
Tetikleyici (Trigger) kavramı nedir? Açıklayınız.**

Saklı Yordam (Stored Procedure):

- Prosedürler dışardan veri alabilir ve geriye bilgi veya veri aktarabilir.
- Prosedürleri yazmamızın amacın yazılan kodu herhangi bir yerde bir daha kullanabiliriz.
- Bir kere yazdığımızda komut derlenir ve çalışmaya başlar.
- Yer ve zamanda tasarruf etmemizi sağlar.
- İşlemci sunucularda bulunması yerine sql kodlarının içinde yani veri tabanında bulunması daha güvenli kullanım avantajı veriyor.

SQLDe Kullanabileceğimiz prosedürler:

- Yorum Satırı prosedürü:

Eğer bir kodlamamın içine not yazmamız gerekiyor bu prosedürden yararlanabiliriz. Örneğin yapan kişinin ismini bu kodlamayla oraya ekleyebiliriz ve bu sayede programı inceleyen bir kişi programı yazan kişiye ulaşmak isterse buradan onun iletişim bilgilerini alıp onunla irtibata geçebilir.

ÖRNEK:

```
/*
Yapımcı = Batuhan Atabey
Mail = batuhanatabey44@gmail.com
*/
```

-GİRİŞ:

Bir prosedürü aşağıdaki gibi yazabiliriz.

```
Create PROCEDURE <procedür ismi>
AS
Begin
<Sql Komutları >
END
```

ÖRNEK:

```
create database batuhan132
use batuhan132

create table arabalar(
araba_id int identity (1,1) primary key,
araba_ad nvarchar(50),
araba_uretici nvarchar(50),
```

```

araba_yil int,
)

insert into arabalar (araba_ad,araba_uretici,araba_yil) VALUES
('BWMx7','BWM',2007),
('MBusinnes','Mersedes',2018),
('Auodia8','Auodi',2000),
('JaguarJeep','Jaguar',2005),
('WWBusines','Wolswogen',2009)

create proc yilp
@yil int
as
begin
select * from arabalar where araba_yil=@yil
end

exec yilp 2007

```

Bu yazdığımız prosedürde , hangi yılı girersek o yılda üretilen araçları görmemizi sağlıyor.

	araba_id	araba_ad	araba_uretici	araba_yil
1	1	BWMx7	BWM	2007

Tetikleyici (Trigger) kavramı nedir?

2.2) Trigger Ne zaman Kullanılır?

- Değişikleri kontrol etmemizi sağlar.
- Birincil anahtar üretmemizi sağlar.
- Karmaşık iş kurallarını gerçekleştirmemizi sağlar.
- Nesneden meydana gelebilicek değişikleri kontrol etmemizi sağlar.

Tabloda belirli olaylar meydana geldiğinde çalışan kodlamalar bütünüdür.

2.3) TRIGGER ÇEŞİTLERİ

Çeşit Trigger bulunmaktadır:

- INSERT
- UPDATE
- DELETE

2.3.1) İNSET:

Belirli gruplara ekleme yapmamızı sağlar.

ÖRNEK KULLANIMI:

```
CREATE TRIGGER trigger_ismi  
ON tablo_ismi  
AFTER veya INSTEAD OF INSERT  
AS  
SQL İfadeleri
```

ÖRNEK:

```
create trigger TriggerInsert  
for insert  
as begin  
declare @b int  
select @b=araba_id from insert into ureticiler (uretici_id,araba_id) VALUES (1,@b)
```

Girdiğimiz değeri , alıp üretici ve araba idisi karşılaştırıp bize eğer uyuşuyorsa ekrana yazdırır.

2.3.2) UPDATE

Belirli gruplarda güncelleme olanağı sağlar.

ÖRNEK KULLANIMI:

```
CREATE TRIGGER trigger_ismi  
ON tablo_ismi  
AFTER veya INSTEAD OF UPDATE  
AS
```

SQL İfadeleri

ÖRNEK:

```
create trigger TriggerUpdate
on Products
after delete
as
select * from arabalar where araba_yil=2007 where araba_id=5
```

2.3.3) DELETE

Belirli grupları silmemizi sağlar.

ÖRNEK KULLANIMI:

```
CREATE TRIGGER trigger_ismi
ON tablo_ismi
AFTER veya INSTEAD OF DELETE
AS
```

SQL İfadeleri

ÖRNEK:

```
create trigger TriggerDelete
on Products
after delete
as
select * from arabalar
```

Arabalar tablosunu silmemizi sağlar.

MERGE KULLANIMI :

SQL Server 2008 Güncellemesi ile aktif olmuştur. Merge komutu; Insert , Delete ,Update gibi komutları ayrı ayrı değil de tek bir seferde yapmamıza olanak sağlayan komutdur.Merge komutu veri ambarı oluşturma konusunda geliştirme bölümünde gerçekleştirilen ETL süreçlerinin parçası olan ETL süreçlerinin bir bölüm olup Slowly Changing Dimensions Yapılarını MERGE kullanarak tasarılayabiliriz.

```
create database batuhan566668
use batuhan566668

create schema ik
go
create table ik.isci(
isId int,
isAd varchar(50),
isSoyad varchar(50),
isMaas int,
)

create table ik.iscibilgi(
isbId int,
isbAd varchar(50),
isbSoyad varchar(50),
isbMaas int
)

insert into ik.isci(isId,isAd,isSoyad,isMaas) VALUES
(1,'Batuhan','Atabey',6000),
(2,'Turan','Mert',9000),
(3,'Bugra','Turhan',4000)

insert into ik.iscibilgi (isbId,isbAd,isbSoyad,isbMaas) VALUES
(1,'Mustafa','Atabey',4000),
(2,'Hesan','Mert',4000),
(3,'Bugra','Turhan',3000)

MERGE INTO ik.isci as i
using ik.iscibilgi as ib
on i.isId=ib.isbId
When matched then
update set i.isAd=ib.isbAd, /*İşçi bilgi tablosunda bulunan ve işçi bilgiden bulunan bilgileri güncelememizi sağlar.*/
           i.isSoyad=ib.isbSoyad,
           i.ismaas=ib.isbmaas
When not matched BY TARGET then /* İçci bilgi tablosunda bulunmayan bilgileri ,içi tablosuna geçirir.*/
           insert (isId,isAd,isSoyad,ismaas) values
(ib.isbId,ib.isbAd,ib.isbSoyad,ib.isbmaas)
When not matched BY SOURCE then /* Girilen verileri silmemizi sağlar */
           delete
output $action as YapilanIslem, deleted.isId,inserted.isId;
```

```

create database batuhan002
use batuhan002

create table HesapBilgiler(
hesapId int primary key identity(1,1) not null,
Ad nvarchar(50),
Soyad nvarchar(50),
bakiye int
)

insert into HesapBilgiler(Ad,Soyad,hesapId,bakiye) VALUES
('Batuhan','Atabey',22222,5000)
('Cem','Asalet',41234,5000)
('Fatih','Bulgan',1,5000)

BEGIN TRANSACTION
UPDATE HesapBilgiler SET Bakiye = Bakiye - 3000
WHERE hesapNo=41234
IF @@ERROR <> 0
ROLLBACK
UPDATE HesapBilgiler SET Bakiye = Bakiye + 3000
WHERE hesapNo=22222
IF @@ERROR <> 0
ROLLBACK
COMMIT TRANSACTION

```

	Yapılan İşlem	isId	isId
1	UPDATE	1	1
2	UPDATE	2	2
3	UPDATE	3	3

TRANSACTION MİMARİSİ

Bir işin basamaklarında herhangi bir hata olduğunda o hataya kadar yapılan işlemleri geri almamızı sağlar.

- **Tutarlılık (Consistency)**
Bölünemezlik kuralının alt yapısıdır. Veri tutarlığını sağlar.
- **İzolasyon(Isolation)**
Veri tabanının istek paketidir. Her transaction ayrı olarak işlenmelidir. Transaction'ın tüm işlemleri gerçekleştirirken başka transaction tarafından görünmelidir.
- **Dayanıklılık(Durability)**
Transactionlar veri tabloları üzerinden işlemler gerçekleştirebilir. Bu işlemleri gerçekleştirebilmek için hatalara dayanıklı olmalıdır.

```

create database batuhan00023
use batuhan00023

create table HesapBilgiler(
hesapId int primary key identity(1,1) not null,
Ad nvarchar(50),
Soyad nvarchar(50),
HesapNo int,
bakiye int
)

insert into HesapBilgiler(Ad,Soyad,hesapNo,bakiye) VALUES
('Batuhan','Atabey',22222,5000),
('Cem','Asalet',41234,5000),
('Fatih','Bulgan',1,5000)

```

	hesapId	Ad	Soyad	HesapNo	bakiye
1	1	Batuhan	Atabey	22222	5000
2	2	Cem	Asalet	41234	5000
3	3	Fatih	Bulgan	1	5000

Aşağıdaki kodlamalarla hesabın bakiyesine ekleme veya çıkarma yapabiliriz.

```

BEGIN TRANSACTION
UPDATE HesapBilgiler SET Bakiye = Bakiye - 3000
WHERE hesapNo=41234
IF @@ERROR <> 0
ROLLBACK
UPDATE HesapBilgiler SET Bakiye = Bakiye + 3000
WHERE hesapNo=22222
IF @@ERROR <> 0
ROLLBACK
COMMIT TRANSACTION

select * from HesapBilgiler

```

	hesapId	Ad	Soyad	HesapNo	bakiye
1	1	Batuhan	Atabey	22222	8000
2	2	Cem	Asalet	41234	2000
3	3	Fatih	Bulgan	1	5000

```

alter proc Havale
@gonderen int,@alan int,@para int,
as
begin
if @gonderen = null
if @alan = null

```

```
begin try
    begin tran havale
        UPDATE HesapBilgiler set bakiye = bakiye-@para where hesapId=@gonderen
        UPDATE HesapBilgiler set bakiye = bakiye+@para where hesapId=@alan
        commit tran havale
    end try
    begin catch
        rollback tran havaleislem
        print 'Hata Meydana Geldi'
    end catch
end

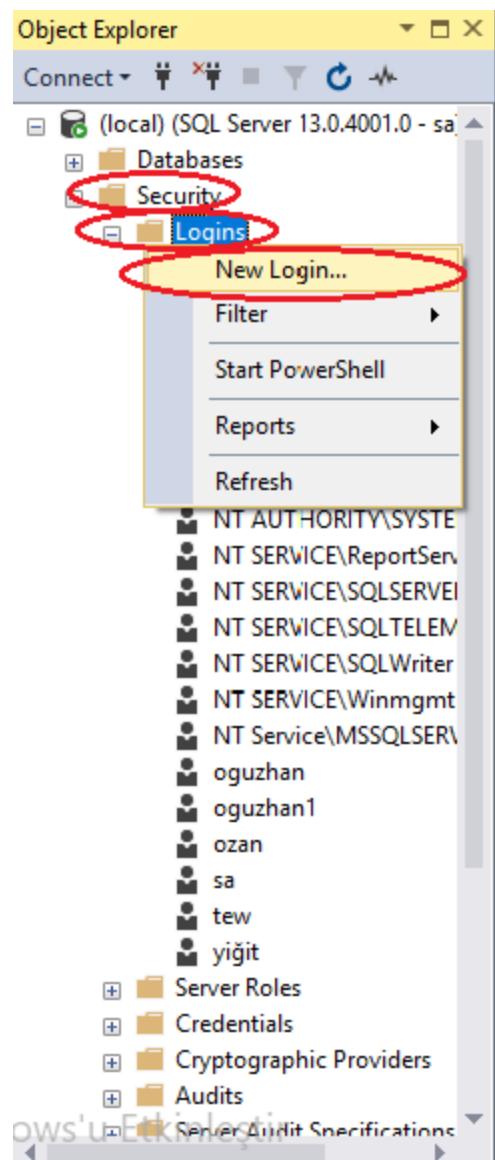
exec havale 2,1,200
```

**DİE 08 - "Veri tabanı kullanıcı, rolleri ve yetkileri nelerdir? Açıklayınız.
Kendi veritabanınızda kullanıcı, rollerini ve yetkilerini oluşturunuz. (8 Adet)
Veritabanı yedegini alma ve yedekten geri yükleme işlemlerini açıklayınız."**

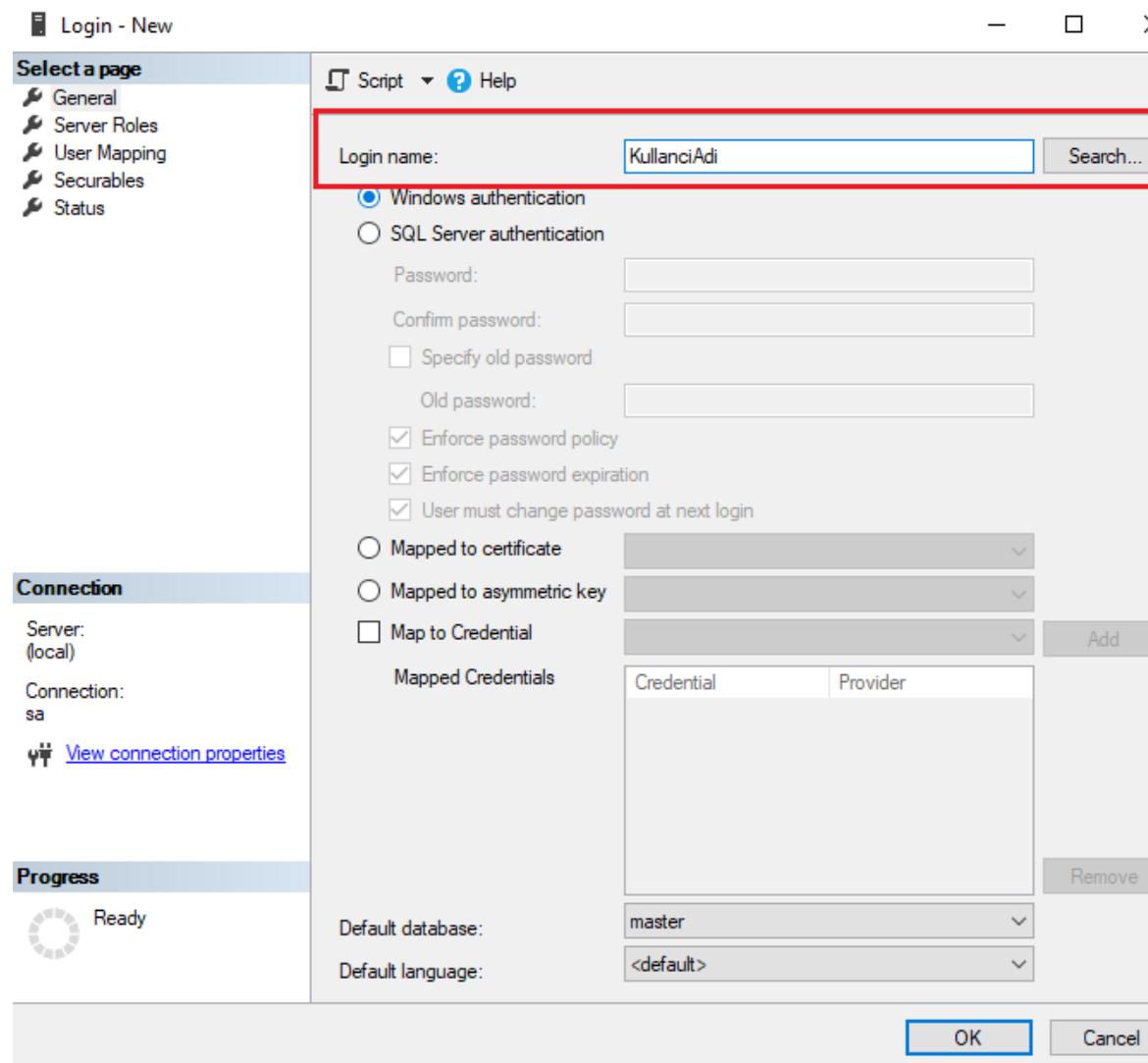
1) SQL Serverda yeni kullanıcı (User) Ekleme:

Yeni kullanıcı eklemek için yapılması gereken işlemler:

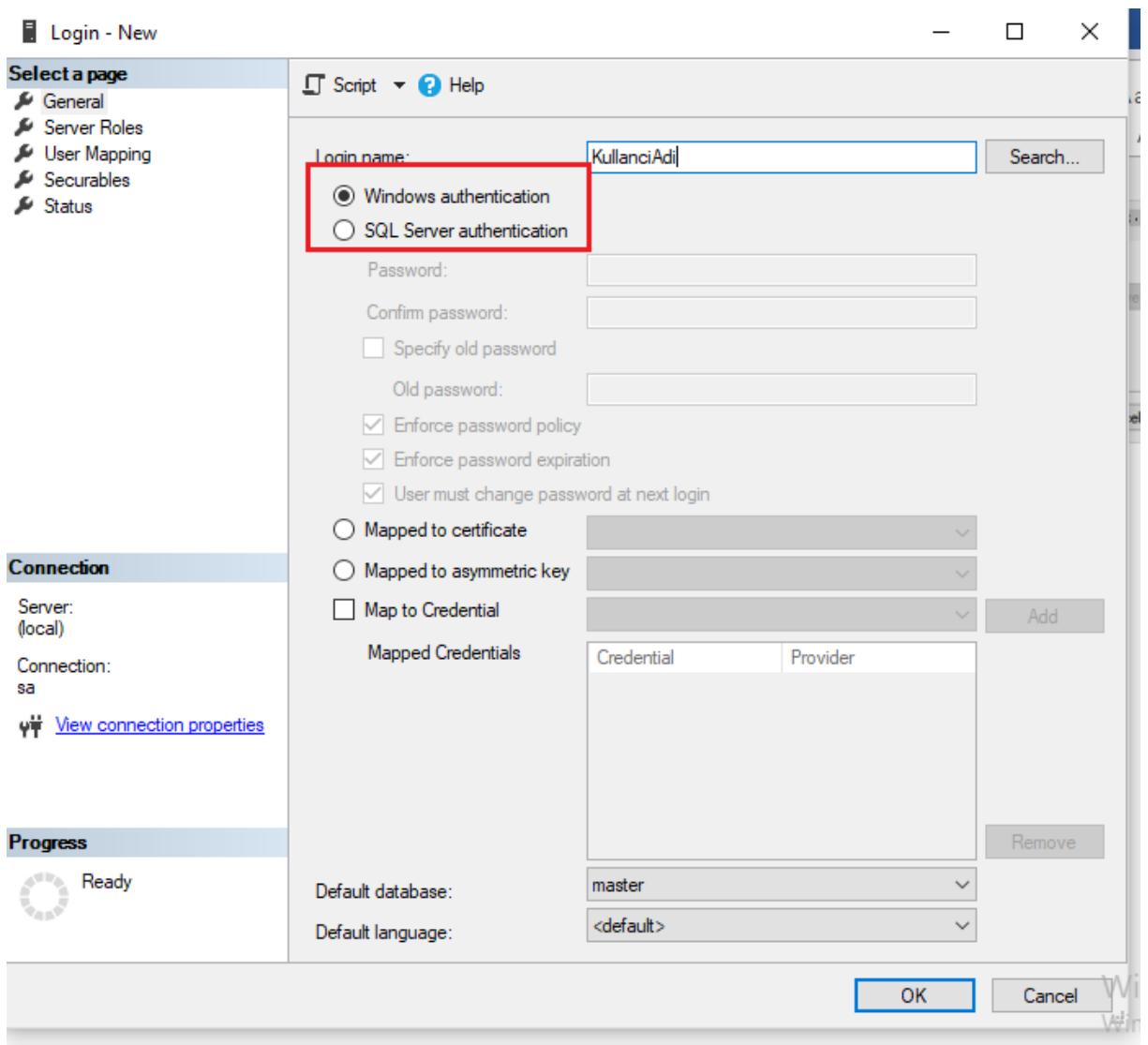
- 1.1. Security Klasörü içindeki Logins klasörüne sağ tıklayarak "New Login" 'i seçiyoruz.



- 1.2. Açılan Login-New Panelinde Login Name bölümüne kullanıcımızın ismini giriyoruz.



1.3. Kullanıcı adımızı girdikten sonra SQL Server Management Studio 'ya nasıl bağlanacağını seçeceğiz. Karşımıza iki seçenek çıkıyor Windows authentication ve .



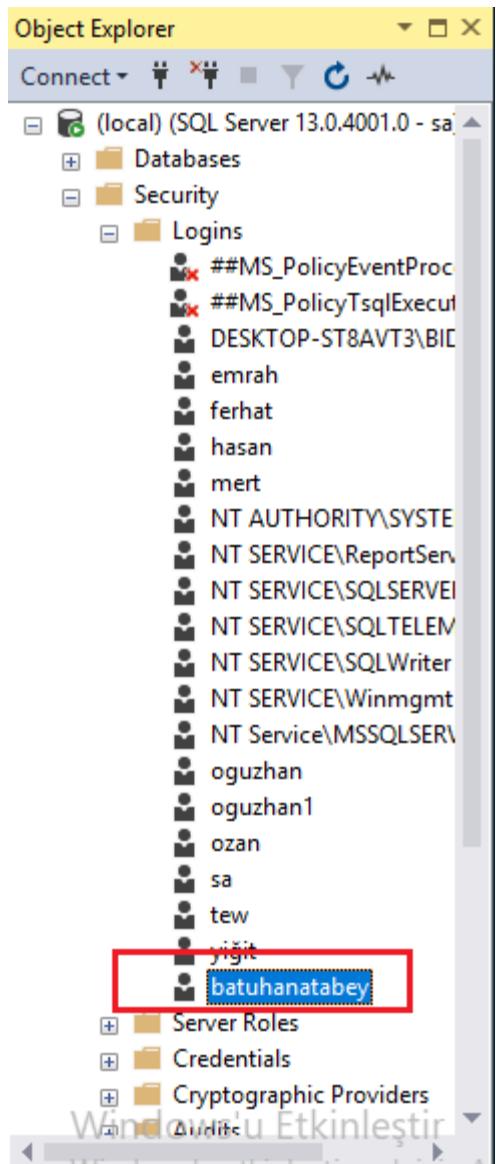
1.3.1. Windows authentication:

- Sadece Windows Hesaplarıyla açılan oturumları kabul eder.
- Kerberos Güvenlik Protokolü : Güçlü parolarlar için hesap kitleme ve parola süresi dolma (Password expiration) özelliğini sağlar.
- Windows authentication Windows ve Domain hesaplarını kabul eder bunun için :
- Domain ortamında ise : [DomainAdı\KullancıAdı] şeklinde belirleriz.
- Workgroup ortamında ise : [BilgisayarAdı(HostName)\KullancıAdı] şeklinde belirleriz.

1.3.2. SQL Server authentication:

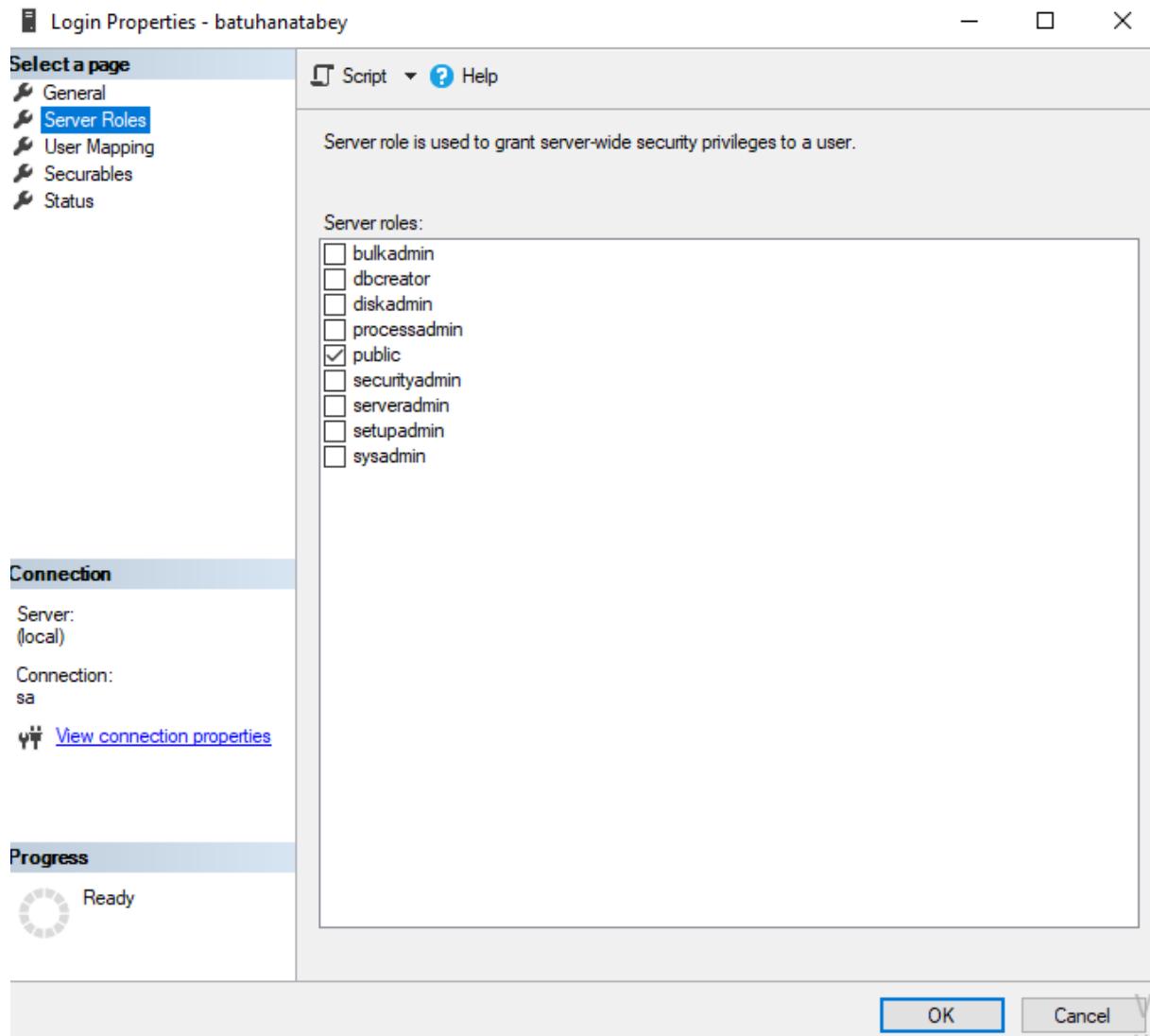
- Burada parolayı saklarken şifreli bir kopyasını değil , parola'nın HASH'ini saklar.
- Kullanıcı tarafından saklanan HASH li şifre ve saklanan HASH li şifre karşılaştırılır eğer doğru ise giriş başarılı sağlanır.

Resimdede görüldüğü gibi kullanıcımızı sisteme ekliyoruz.

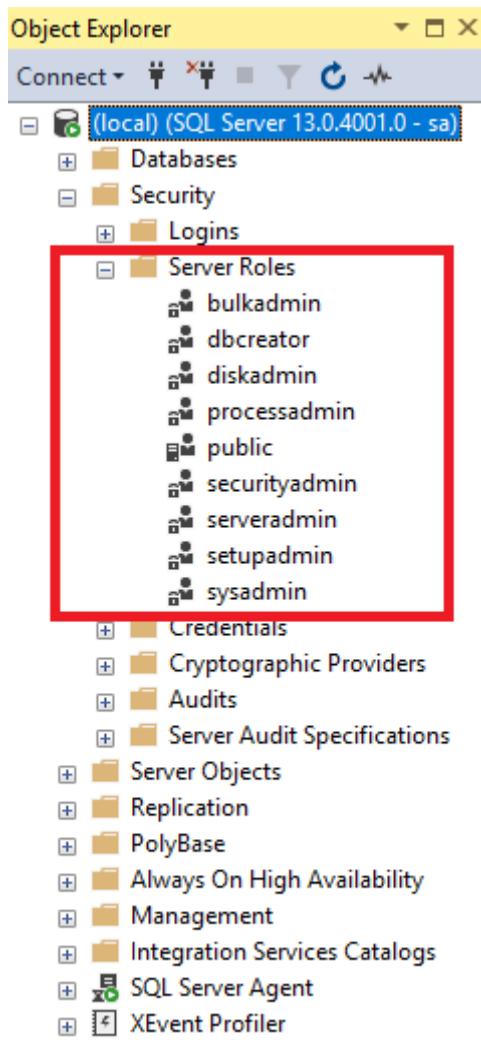


2) Server Roles Kullanıcıya rol verme

- Bu bölümde kullanıcıyı yetki atayabiliriz. 3 Çeşit Yetki türü vardır : Server Role , Database Role , Application Role



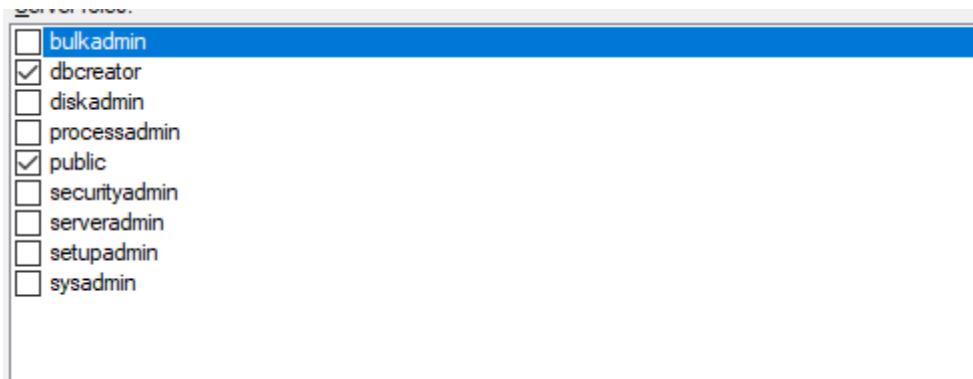
2.1. Server Role:



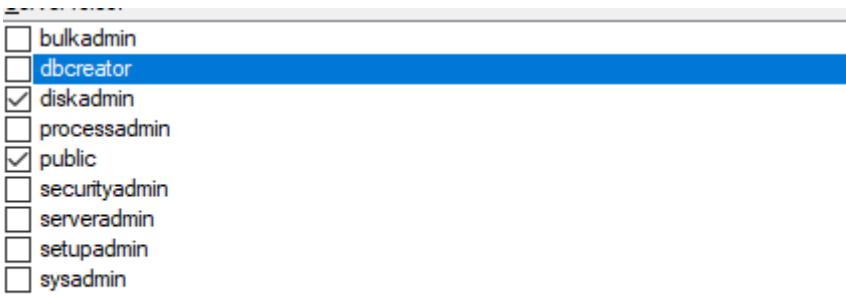
bulkadmin (Bulk Insert Administrator – Çoklu Kayıt Yöneticisi) : Bulk Insert komutuna yetkisi vardır.

Server roles:
<input checked="" type="checkbox"/> bulkadmin
<input type="checkbox"/> dbcreator
<input type="checkbox"/> diskadmin
<input type="checkbox"/> processadmin
<input checked="" type="checkbox"/> public
<input type="checkbox"/> securityadmin
<input type="checkbox"/> serveradmin
<input type="checkbox"/> setupadmin
<input type="checkbox"/> sysadmin

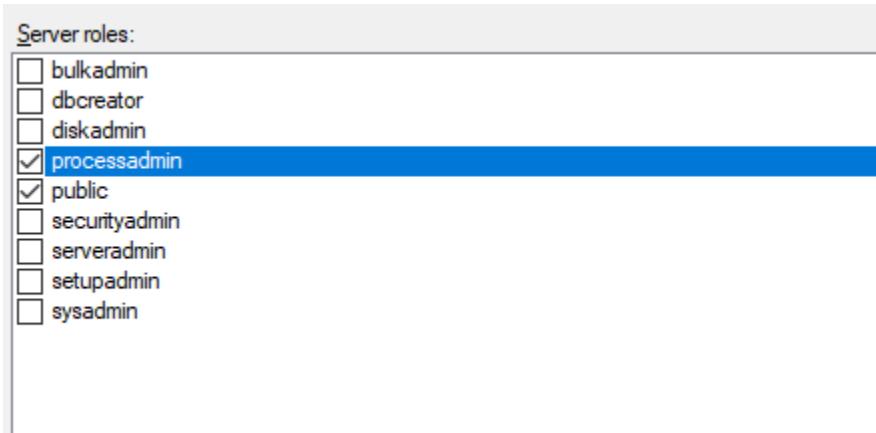
dbcreator (Database Creator – Veritabanı Yöneticisi) : Database oluşturmak , silmek ve düzenlemek.



diskadmin (Disk Administrator – Dosya Yöneticisi) : Disk dosyalarını düzenlemek.



processadmin (Process Administrator – İşlemci Yöneticisi) : İşlemcileri kontrol etmek.



public (Herkese Kısıtlı Hak) : securityadmin (Security Administrator – Güvenlik Yöneticisi) : Herkesin aynı yetkiye sahip olduğu roldür.

Server roles:

- bulkadmin
- dbcreator
- diskadmin
- processadmin
- public
- securityadmin
- serveradmin
- setupadmin
- sysadmin

Securityadmin(Server Administrator – Güvenlik Yöneticisi) : Servera bağlı kullanıcıların bilgilerini değiştirmek , güncellemek ve silmekle görevlidir.

Server roles:

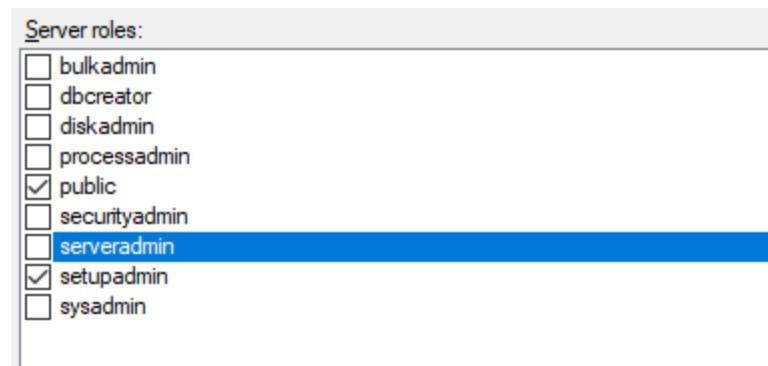
- bulkadmin
- dbcreator
- diskadmin
- processadmin
- public
- securityadmin
- serveradmin
- setupadmin
- sysadmin

serveradmin (Server Administrator – Server Yöneticisi) : Serveri başlatma,durdurma ve yeniden başlatma görevlerini üstlenir.

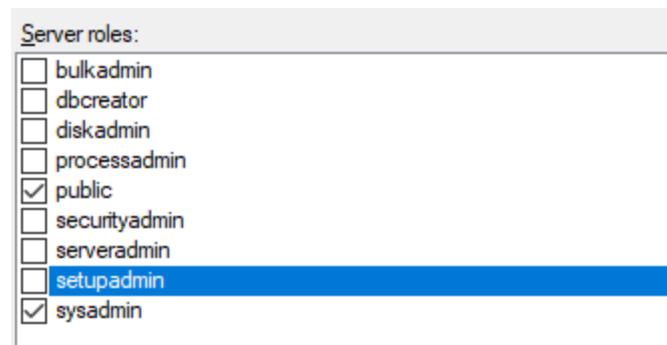
Server roles:

- bulkadmin
- dbcreator
- diskadmin
- processadmin
- public
- securityadmin
- serveradmin
- setupadmin
- sysadmin

setupadmin (Setup Administrator) : Server üzerinde farklı bir veritabanından kullanarak işlem yapmayı sağlar



sysadmin (System Administrator – Sistem Yöneticisi) : Sistem üzerindeki tüm yetkilere sahiptir.



- batuhanatabey-bulkadmin
- batuhanatabey-dbcreator
- batuhanatabey-diskadmin
- batuhanatabey-processadmin
- batuhanatabey-public
- batuhanatabey-securityadmin
- batuhanatabey-serveradmin
- **batuhanatabey-setupadmin**
- batuhanatabey-sysadmin

| Server Roles

2.1.1. Bu kuralların prosedür ile yönetilmesi :

sp_helpsrvrole : Serverda bulunan kural listelerini görüntüler.

sp_helpsrvrolemember : Kullanıcılara tanımlanana kuralları görüntüler.

sp_srvrolepermission : Kuralları tanımlanan yetki listesini görüntüler.

sp_addsrvrolemember : Kullancıya kural ataması yapar Örnek: exec sp_addsrvrolemember @loginname='shrcy', @rolename = 'bulkadmin'

sp_dropsrvrolemember : Kullancıya atanan kuralları silmemizi sağlar. Örnek: exec sp_dropsrvrolemember @loginname ='shrcy', @rolename = 'bulkadmin'

2.2. DataBase Role

Use Mapping Bölümünde bulunan Database Role membership 'den kullanıcılar bu yetkileri atıyalırız.

The screenshot shows the 'User Mapping' dialog box in SSMS. The left sidebar has tabs for 'General', 'Server Roles', 'User Mapping' (which is selected), 'Securables', and 'Status'. The main area has sections for 'Users mapped to this login:' and 'Database role membership for: Banka'. In the 'Database role membership for: Banka' section, the 'public' role is checked, while other roles like 'db_owner', 'db_securityadmin', and 'db_datareader' are not checked.

db_owner : Veri tabanındaki en yüksek yetkiye sahiptir.Silme ,ekleme,düzenleme ,güncelleme ,başlatma/durdurma yetkilerine sahiptir.

db_accessadmin : Kullanıcılara rol verme yetkisine sahiptir.

db_securityadmin : Veritabanındaki kural ve yetkileri yönetir.

db_datareader: "Select" komutunu çalıştırmasına izin verir.

db_datawriter : "Insert , Delete , Update" gibi komutları çalıştırmasına izin verir.

db_ddladmin : “DLL” komutlarını çalıştırmasına izin verir.

db_denydatareader : “Select” komutunu çalıştırılmasını kısıtlar.

db_denydatawriter : “Insert,Delete,Update” gibi komutları çalıştırılmasını kısıtlar.

db_backupoperator : Veri tabanının yedeğini almak için kullanılan kuraldır.

2.2.1. Bu kuralların prosedür ile yönetilmesi:

sp_helprole: Serverdaki kural listelerini ve Application Rolelerini gösterir.

sp_addrolemember: Kullancıya kural ataması yapmamızı sağlar. Örnek: exec sp_addrolemember @membername='shrcy'

sp_droprolemember: Kullancıdaki mevcut kurallı kaldırılmamızı sağlar. Örnek: exec sp_droprolemember @membername = 'shrcy'

sp_helpdpfixedrole: Veri tabanındaki bütün kuralları ve yetki listelerini gösterir.

2.3. Application Role

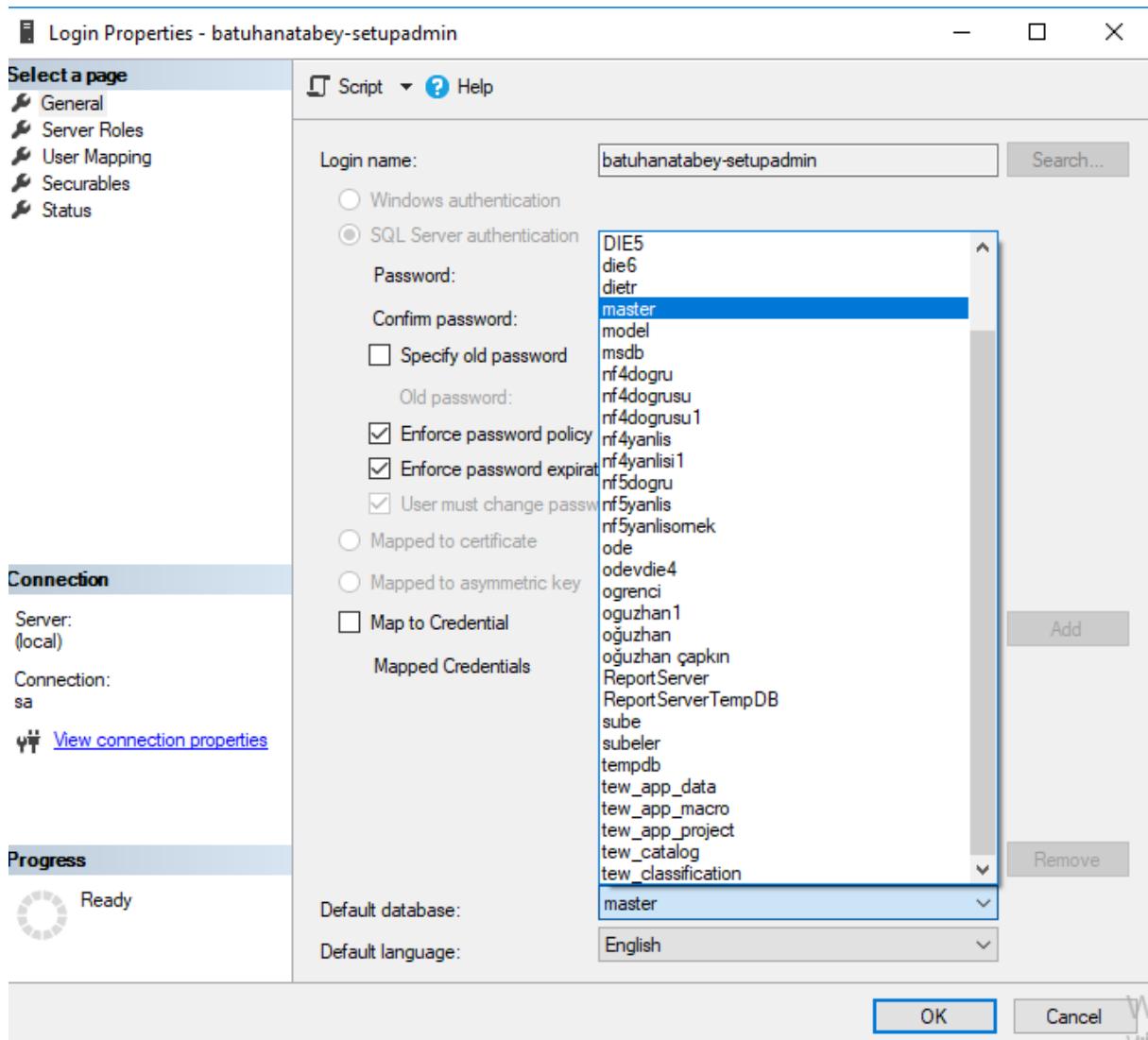
“Quest” hesabı oluşturursak , tek tek kullanıcı hesabı açıp tanımlamamıza gerek yoktur. Aşağıdaki iki prosedür ile bunu kullanılabılır hale veya kullanılamaz hale getirebiliriz.

sp_setapprole : Kullanılabilir hale getirir.

sp_unsetapprole: Kullanılamaz hale getirir.

2.4 Rollerin Kullanım Alanları Belirlemek:

Login Properties penceresinde Default Database bölümünde rolün kullanılacağı bölümü belirlememizi sağlar.



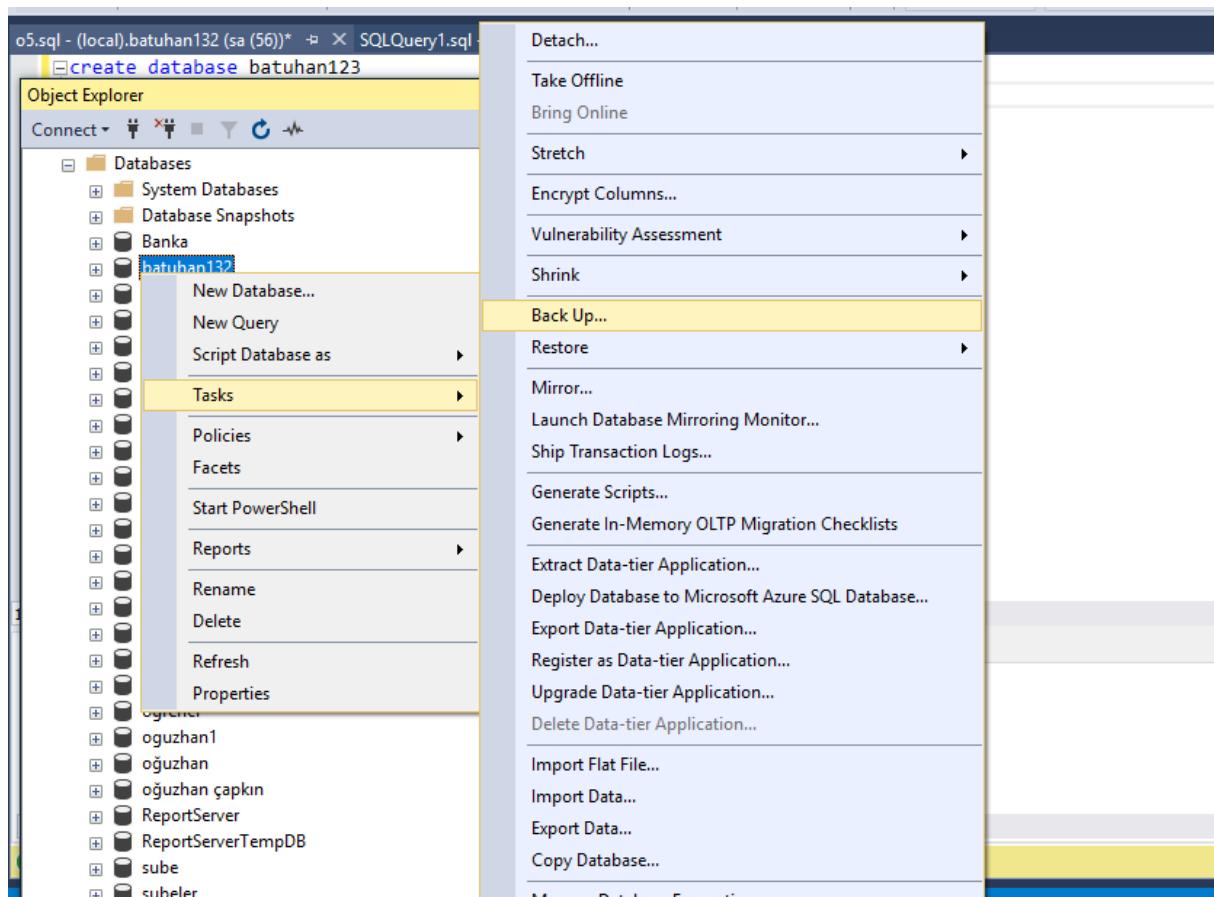
YEDEK ALMA(Back Up)

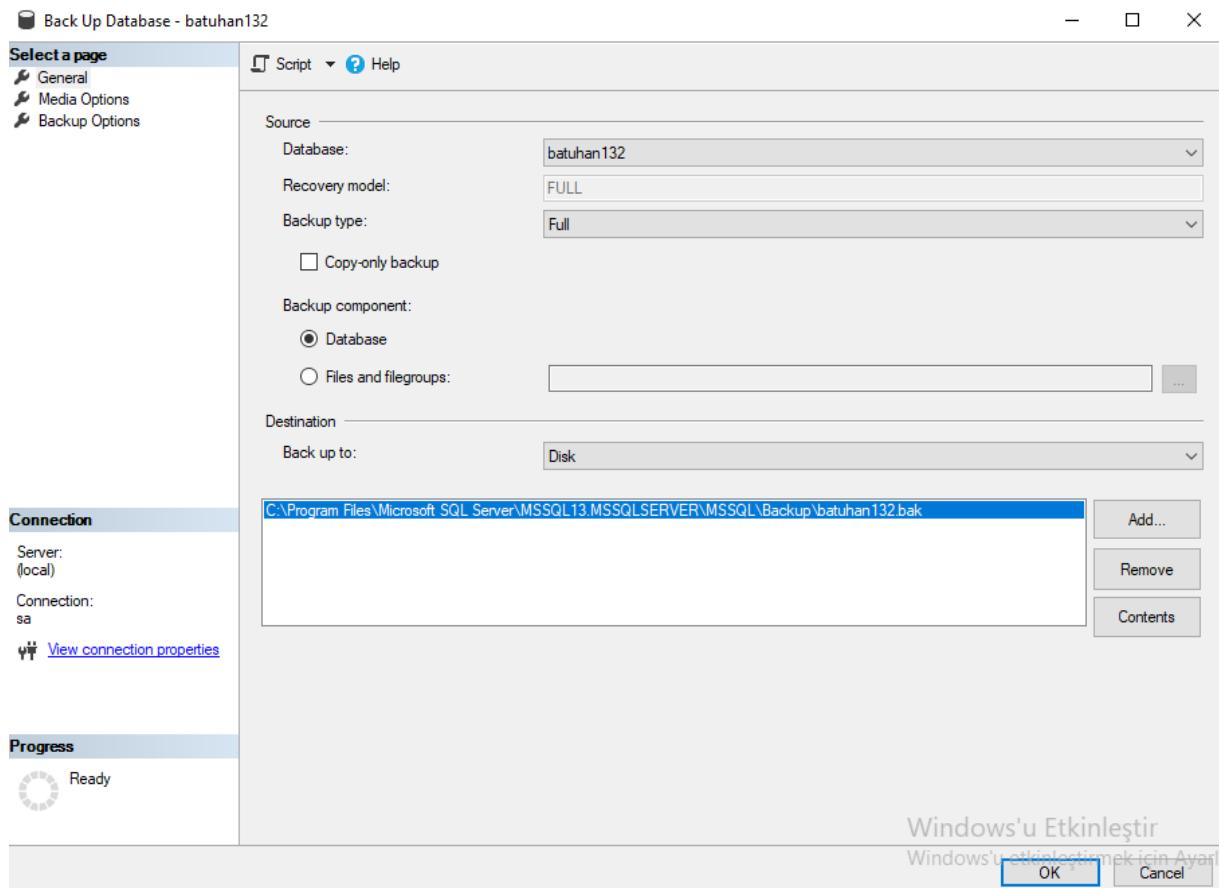
Yedek alma işlemini yapmamızı sağlar.

Full BackUp: Veritabanındaki herşey yedeğin içine kopyalanır. Elimizde bulunan bir full backup ile sistemin herşeyini yükleyebiliriz.

Differential Backup: Çok fazla verinin bulunduğu veri tabanlarından full backup almak uzun süre ve çok fazla yer kapsar. Bu yüzden sadace belirtilen yerlerin yedeğini almamızı sağlar.

Transaction Log Backup: Transaction , ilgili veritabanındaki yapılan her işlemin bilgilerinin tutulduğu log dosyasıdır.





1.1. Source Bölümü Ayarları:

Database: Hangi veri tabanının yedeğini almak istersek onu seçeriz.

Recovery Model: Logların nasıl kayıt edileceğini belirler.

Backup Type:

Copy Only Backup: Bu yedek alma işlemi backup zincirine dahil edilicekmi edilmiyicekmi

Backup Component: Tüm veri tabanını mı yoksa belirlenen alanınımı yedeğini alacağız.

1.2. Backup Set Bölümü Ayarları:

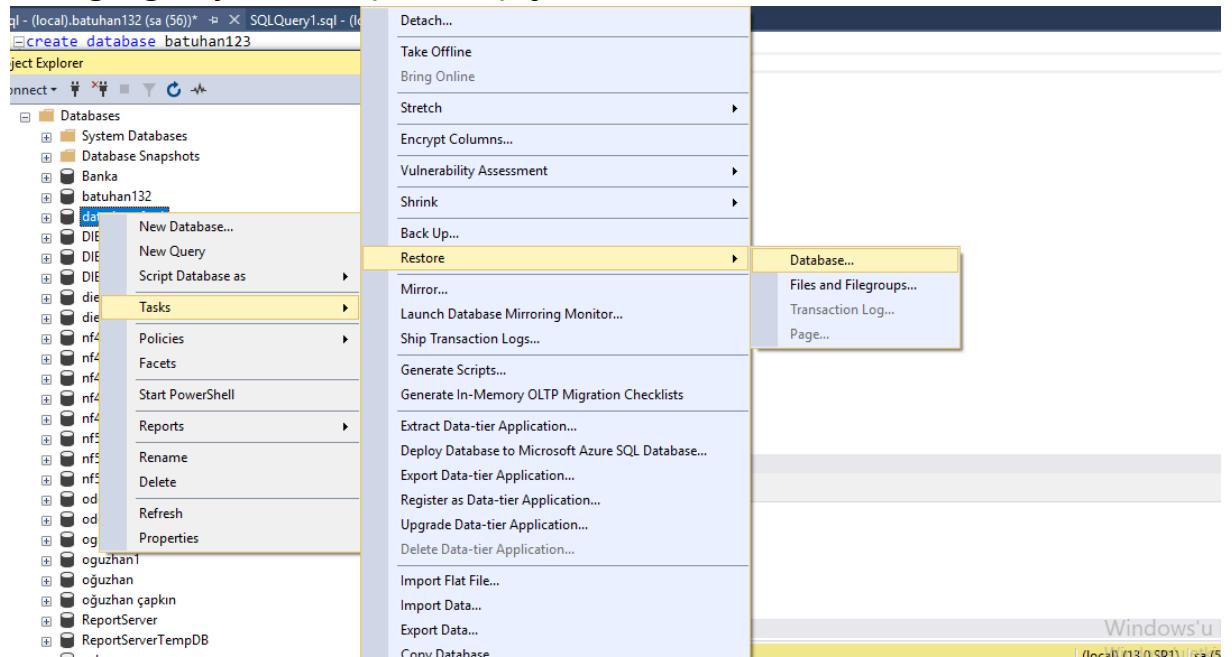
Name: Yedeğin ismi

Description: Açıklama bölümündür

1.3. Destination Bölümü Ayarları:

Backup To: Yedeğin nerede saklanacağını belirleriz.

Yedeğin geri yüklemesi(Restore) İşlemi:



DİE09 – SQL İnjection & Video Raporlama

Sql İnjection Nedir?

Internet sitelerinin bir çoğu veri tabanından yararlanır. Sql injectionu kullanma amacımız SQL sorgularına müdahale ederek bilgiler elde edilmektedir. Sql injection sayesinde üye bilgileri , yönetici bilgileri gibi kullanılan kullanıcı bilgilerini ve hedefimizdeki SQL veri tabanındaki diğer bilgileri alabilmemizi sağlar.

- Sisteme giriş yapılabilir.
- Yetkisiz alanlara giriş yapılabilir.
- Sistemin veritabanını değiştirebilir.

SQL İnjection Nasıl Çalıştırılır ?

Örnek üzerinden gitmek gereklidir :

SQL kullanılan bir sitede , kullanıcı adı = user ve şifre = 123 ise “Giriş Yap” Buttonuna bastığımızda aşağıdaki sorgu çalışacaktır.

```
SELECT * FROM kullanıcılar WHERE isim = 'user' AND şifre ='123'
```

Yazılım sonuç buluyorsa giriş yapılacaktır ,eğer bulunmaz ise izin verilmeyecektir.

Bir başka örnekde kullanıcı adı ve şifre alanına ‘OR 1=1- yazdığımızda çalışacak sorguya bakalım.

```
SELECT * FROM kullanıcılar WHERE isim=' OR 1=1 ' AND şifre=' OR 1=1 --'
```

Sorgu gönderilecektir eğer yazılım girdileri engelliyorsa SQL İnjection engelleyip girişe izin verilmeyecek.

SQL İnjection ile veri alma:

Şimdi öncelikle hedef bir site belirlememiz gerekiyor . Biz <http://www.pro9.co.uk/> sitesinden verileri almayı deniyelim.

<http://www.pro9.co.uk/html/print.php?sid=255>

yazdığımızda o idye ait bilgileri alabiliriz.

← → C Güvenli değil | www.pro9.co.uk/html/print.php?sid=255

The 2006 Mosconi Cup - tickets now on sale!
A Pro9 - Europe's No.1 Pool Player Resource Article
<http://www.pro9.co.uk/html/>

Date: Wednesday, October 25 2006 @ 07:55:37 UTC
Topic: Mosconi Cup

The 2006 Mosconi Cup
Cruise Terminal
Wilhelminakade 699
Rotterdam
Holland
www.mosconicup.com (2005 website)
www.cruiceterminalrotterdam.nl

7 - 10 December 2006



- <http://www.pro9.co.uk/html/print.php?sid=259> yazdığımızda 259 numaralı idye ait bilgileri alırız.

← → C Güvenli değil | www.pro9.co.uk/html/print.php?sid=259

Mickey Flynn's Announce 8-Ball and 10-Ball Events for
2007
A Pro9 - Europe's No.1 Pool Player Resource Article
<http://www.pro9.co.uk/html/>

Date: Saturday, October 28 2006 @ 13:09:16 UTC
Topic: Mickey Flynn's

MickeyFlynn's American Pool
103 Mill Road
Cambridge
CB1 2AZ
Tel: 01223 309 000
www.mickeyflynn's.co.uk

Wednesday 17 January 2007



DİE 11. İstanbul Bilişim Kongresi Konu Başlıklarları

Dijital Dönüşüm ve Verinin Önemi

21. yüzyıl dijital bir çağ olmakla beraber yapay zekanın da önemi bize anlatan bir çağdayız.

Buda insanların bazı kodları hayatının bir parçası olmaya başlamadığını gösteriyor. Örneğin canlılara ait faaliyetleri kodlar yardımıyla dijital materyaller yardımıyla hayatı geçirebiliriz.

Dijitalizasyonla birlikte ortaya çıkan bir çok akıllı makineler hayatımıza girdi. Bunlara bağlı olarak hayatımızda farklı kategorilerde ürünler meydana geldi.

Bu çağda veriye bağlı olarak gelişeceği bazı olaylar meydana gelebilir, bu olaylarda en karlı olmak için en faydalı ve en çok veriye sahip olmak gereklidir. Hangi ülkenin bu özelliği ön planda ise o ülke kendini diğer ülkelerden daha çok geliştirebilir buda 21.yüzyılda verinin ne kadar önemli olduğunu bize gösteriyor.



Veri Madenciliği

Dijital çağda verilerin toplanması ve saklanması sonucunda bir veri ağının oluşumu sağlanmıştır. Bu kadar büyük verilerin karmaşısında istediğimiz verilerin bulunması da zorlaşmıştır. Bu verilerin çeşitleri gün geçikçe artmaktadır ve daha da büyük bir veri ağını oluşturmaya devam ediyor. Bu kadar fazla verinin içinde istediğimiz veriyi bulabilmemiz açısından veri madenciliği çok önemlidir.

Veri madenciliği farklı şekillerde ticari işletmeler tarafından kullanılmıştır.

- Parakendecilik : Market mağazalarında ürünlerin fiyatları ,teslim tarihleri ve satış tarihleri hakkında verileri toplanmasında yararlanır.
- Bankacılık: Müşterinin kişisel bilgilerini(tcno,ad,soyad,vb..) saklar ve buna göre işlemler yaparak faaliyet sürdürür.
- Pazarlama : Örneğin bir kişiye bir ürün satmak istediğinizde onun neyerde hoşlandığınızı bilmeniz sizin satış yapma olasılığınızı artırır. Bir kişinin sevdiği rengi bilirseniz ona göre bir seçim sunabilirsiniz buda müşterinizi memnun eder ve satış yapma oranınızı artırır.
- İnsan Kaynakları Yönetimi: İnsanların hangi konularda daha iyi olduğunu anlayabilmemiz için verilerden yararlanabiliyoruz bu verilerle onlara uygun konumu bulup onları o bölüde görevlendiririz.
- Polislik : İnsanların önceden suç işlemleri görebilir ve buna göre tedbirler alıp bu durumun önüne geçebiliriz.

Geleceğin Yeni Veri Otobanı ve İnterneti

Bilgisayarlar birbiriyle iletişime geçen sanal varlıklarlardır.Birbirleri ile kablolu yada kablosuz Bağlantı kurabilerler.Bir bilgisayardan diğerine herhangi bir bilgi,resim,fotoğraf veya video yolladığınızda bu veri hatlar yardımıyla karşı bilgisayara ulaşır işte bu hatlara veri otobanları deriz.

Veri otobanlarının önemi aslında çok büyük bir önem arz etmektedir. Bir örnekden yola çıkmak gerekirse bir otomobil ile yolda ilerlenen yolunuzun nasıl olduğun önemlidir. Yol dar ise trafik yavaş ilerler yol geniş ise rahat ilerler.Yol engebeli ve çukurlu ise yavaş ilerle , yol sorunsuz ve düz ise rahat ilerler. İşte bu veri içinde geçerlidir örneğin bir veri boyutundan küçük bir veri otobanı ile karşı tarafı aktarılmak istenirse bu o verinin oldukça yavaş ve sorunlu aktarılmasına sebep olur.

Verinizin Değeri Güvenliği Kadardır

Günümüzde değerli olan veri zaman geçikçe daha da değer almaya devam ediyor.

Gelecekte veri dünyanın vazgeçilemez bir parçası olacaktır. Buda verinin değerini artırıyor , gelecekte veri kimin elinde ise o her zaman diğerlerine göre önde olacaktır. Buda insanoğlunun aklına bir çok soru getiriyor. Acaba benim bilgilerim başkasının elinde var mı ? gibi. Günümüzde birçok şirket belli bir para karşılığında bazı böülümlerden veri alabiliyor bu verileri de size ulaşmak algınızı değiştirmek için kullanıyor. Örneğin bir ayakkabı sitesine girdiğiniz zaman oradan ayakkabı almamış olsanız dahil daha sonrasında size girmiş olduğunuz farklı sitelerde o ayakkabı firmasının sitesini gösterip sizi oraya yönlendirmeye çalışırlar buda sizin aklınıza o ayakkabı alma fikrini yeniden getirir ve belki de ilk gördüğünüzde almadığınız ayakkabıyı sonrasında alicaksınız. Buda sizi verilerinizin aslında bir değer olduğunu gösteriyor bu yüzden girmiş olduğumuz sitelerin güvenirliginden emin olmadan hiçbir siteye kişisel bilgilerimizi vermemiz gerektiğini bize gösteriyor

KAYNAKÇALAR

- **DİE 01 - Veri tabanı , VeriTabanı yönetim sistemi, SQL server avantajları**

1. 1.Madde Veri tabanı ve 2. Madde Veri tabanı yönetim sistemleri:

<https://teknokoliker.com/2012/05/veri-taban-nedir.htm>

2. 3. Madde Sql Server ve Avantajları :

<https://teknokoliker.com/2012/05/veri-taban-nedir.html>

<http://selahattin.iosoriochong.com/mysql-vs-ms-sql-kullanmanin-avantajlari-ve-dezavantajlari/>

- **DİE 02 - Normalizasyon Nedir Normalizasyon kuralları nelerdir**

<http://www.buraksecer.com/veri-tabani-normalizasyonunun-amaclari-ve-kurallari/>

- **ODEV 03 - Veri tipleri**

<https://www.kodlamamerkezi.com/veritabani-sql/sql-server-2012-veri-tipleri/>

- **DİE 03 - Veri tipleri**

<https://www.kodlamamerkezi.com/veritabani-sql/sql-server-2012-veri-tipleri/>

- **DİE 04 - DateTime DBCC CHECKIDENT**

<https://netsisnedir.wordpress.com/2011/05/06/sql-server-dbcc-komutlari/>

<http://www.muratoner.net/sql/sql-datetime-fonksiyonlarinin-kullanimi>

- **DİE 05 - Şema ,index , view**

http://belgeler.gen.tr/man/man7/man7-create_schema.html

<https://mcansozeri.wordpress.com/2011/03/30/mssqlde-view-kullanimi-create-view-alter-view-drop-view/>

<http://www.ismailgursoy.com.tr/index-nedir-ne-ise-yarar/>

- **ODEV 05 - Saklı yordam , Tetikleyici**

<http://blog.ehocam.com/2010/09/sql-server-uzerinde-stored-procedure-anlamak-ve-kullanmak.html>

<http://www.yazilimciblog.com/trigger>

- **DİE 06 - Merge , Transaction Mimarisi**

<http://www.cihanozhan.com/sql-serverda-transaction-kullanimi-yonetimi/>
<http://www.hayriali.com/transaction-mimarisi/>

- **DİE 08 - "Veri tabanı kullanıcı, rolleri ve yetkileri nelerdir?**

<http://www.cihanozhan.com/sql-serverda-transaction-kullanimi-yonetimi/>
<https://social.technet.microsoft.com/wiki/contents/articles/34711.sql-server-merge-komutu-kullanm-ve-performans-onerileri-tr-tr.aspx>
<http://www.hayriali.com/transaction-mimarisi/>

- **DİE09 – SQL İnjeksiyon & Video Raporlama**

<https://www.kodevreni.com/526-sql-injection-nedir-ve-sql-injection-nas%C4%B1-%C3%B6nlenir/>

- **DİE10 – İstanbul Bilişim Kongresi**

<http://anahtar.sanayi.gov.tr/tr/news/dijital-donusum/9628>

<http://blog.euromsg.com/data-mining-veri-madenciliği-nedir/>

<https://nenedir.com.tr/bilisim-otobani-information-superhighway-nedir/>

