

**T.C.  
MİLLÎ EĞİTİM BAKANLIĞI**

# **BİLİŞİM TEKNOLOJİLERİ**

**AĞ VERİ TABANI YÖNETİMİ**  
**481BB0099**

**Ankara, 2012**

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- **PARA İLE SATILMAZ.**

# İÇİNDEKİLER

AÇIKLAMALAR .....	ii
GİRİŞ .....	1
ÖĞRENME FAALİYETİ-1 .....	3
1. VERİ BÜTÜNLÜĞÜ .....	3
1.1. Görünümler (View).....	3
1.1.1. Yeni Görünüm Oluşturma .....	4
1.1.2. Görünümü Güncelleme.....	8
1.1.3. Görünümü Silme.....	9
1.2. Tetikleyiciler (Trigger) .....	9
1.2.1. Yeni Tetikleyici Oluşturma .....	10
1.2.2. Tetikleyiciyi Düzenleme.....	13
1.2.3. Tetikleyici Silme.....	14
1.3. Saklı Yordamlar (Stored Procedures) .....	14
1.3.1. Saklı Yordam Oluşturmak .....	15
1.3.2. Saklı Yordamlarda Değişiklik Yapmak .....	16
1.3.3. Saklı Yordamı Silmek.....	18
1.3.4. Değer Alan Alt Yordamlar .....	18
UYGULAMA FAALİYETİ .....	21
ÖLÇME VE DEĞERLENDİRME .....	25
ÖĞRENME FAALİYETİ-2 .....	26
2. VERİ TUTARLILIĞI.....	26
2.1. Kısıtlayıcı (Constraint).....	26
2.2. Kısıtlayıcı Türleri.....	27
2.2.1. Birincil Anahtar Kısıtlayıcı (Primary Key Constraint).....	27
2.2.2. Tekil Alan Kısıtlayıcı (Unique Constraint) .....	28
2.2.3. Yabancı Anahtar Kısıtlayıcı (Foreign Key Constraint) .....	28
2.2.4. Varsayılan Kısıtlayıcı (Default Constraint) .....	29
2.2.5. Kontrol Kısıtlayıcı (Check Constraint).....	30
2.3. Kısıtlayıcıları Düzenlemek .....	30
2.4. Kısıtlayıcıları Silmek .....	31
2.5. Varsayılan (Default) Nesnesi Oluşturmak .....	32
2.6. Rol (Rule) Oluşturmak.....	32
UYGULAMA FAALİYETİ .....	33
ÖLÇME VE DEĞERLENDİRME .....	37
MODÜL DEĞERLENDİRME .....	38
CEVAP ANAHTARLARI.....	40
KAYNAKÇA .....	41

# AÇIKLAMALAR

<b>KOD</b>	<b>481BB0099</b>
<b>ALAN</b>	<b>Bilişim Teknolojileri</b>
<b>DAL/MESLEK</b>	<b>Veritabanı Programcılığı</b>
<b>MODÜLÜN ADI</b>	<b>Ağ Veritabanı Yönetimi</b>
<b>MODÜLÜN TANIMI</b>	Ağ veritabanı yönetimi ile ilgili bilgilerin verildiği öğrenme materyalidir.
<b>SÜRE</b>	<b>40/32</b>
<b>ÖN KOŞUL</b>	Ağ Veri Tabanında Temel Tablo İşlemleri” modülünü tamamlamış olmak
<b>YETERLİK</b>	Ağ Veritabanı Yönetimini Sağlamak
<b>MODÜLÜN AMACI</b>	<b>Genel Amaç</b> Öğrenci bu modül ile gerekli ortam sağlandığında; görünüm, tetikleyici, alt program ve indisleri düzenleme, kullanıcı rollerini ve kullanıcı tanımlı veri tiplerini düzenleme, kural ve varsayılan değerleri düzenleme işlemlerini yapabileceksiniz. <b>Amaçlar</b> <b>1.</b> Veri bütünlüğünü sağlayabilecektir. <b>2.</b> Veri tutarlılığını sağlayabilecektir.
<b>EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI</b>	<b>Ortam:</b> Bilgisayar Laboratuvarı <b>Donanım:</b> Bilgisayar, lisanslı ağ veritabanı yazılımları
<b>ÖLÇME VE DEĞERLENDİRME</b>	Modülün içinde yer alan her faaliyetten sonra verilen ölçme araçları ile kazandığınız bilgileri ölçerek kendi kendinizi değerlendireceksiniz. Öğretmen, modülün sonunda, size ölçme aracı (test, çoktan seçmeli, doğru-yanlış, vb. ) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek değerlendirecektir.

# GİRİŞ

## **Sevgili Öğrenci,**

Son yıllarda yapılan birçok proje, çok sayıda bilgisayar tarafından kullanılabilir şekilde tasarlanmaktadır. Bu, ağ ortamında birden fazla kullanıcının aynı proje üzerinde çalışabilmesine olanak vermektedir. Bu işlemleri çok sık kullandığımız veritabanı programıyla da yapabileceğiniz gibi ağ ortamında güvenlik ve hızlı erişim açısından en iyi sonucu veren SQL Server veritabanıyla da yapabilirsiniz. Bu program, milyonlarca kaydın olduğu tablolar üzerinde işlem yaparken tüm kullanıcılara hitap edebilmekte ve istenilen sorgu sonuçlarını en hızlı şekilde elde edebilmenizi sağlamaktadır.

Bu modülle, ağ veritabanı için veri bütünlüğünü sağlamayı ve kısıtlamalar oluşturmayı, oluşturulan bu kısıtlamaları yönetmeyi, saklı prosedür tanımlamayı ve bunları sorgu içinde kullanabilmeyi öğreneceksiniz.



# ÖĞRENME FAALİYETİ-1

## AMAÇ

Bu öğrenme faaliyetinde, görünüm, tetikleyici ve saklı yordamları kullanarak veri bütünlüğünü sağlayabileceksiniz.

## ARAŞTIRMA

Veri bütünlüğü sağlanmayan bir veritabanında ne gibi sorunlarla karşılaşılabilir?  
Araştırıp maddeler halinde listeleyiniz.

## 1. VERİ BÜTÜNLÜĞÜ

Veritabanı içinde bir tabloda veri güncelleme, silme veya ekleme gibi işlemler yapılırken diğer tablo ya da tablolardaki verilerin de birbirleriyle uyum içinde olması gerekmektedir. Bu amaç doğrultusunda; veri tutarlılığının kaybolmamasının garanti altına alınması, **veri bütünlüğü** olarak adlandırılır.

Veri bütünlüğü, iki farklı yöntemle sağlanabilir:

**Tanımlanabilir veri bütünlüğü:** Tanımlanan nesnelere kendi özellikleri sayesinde sağlanabilen veri bütünlüğüdür.

**Prosedürel (programsal) veri bütünlüğü:** Bir programlama mantığıyla bütünlüğün tasarlanması gerekir. SQL’de bu yaklaşım ise trigger (tetikleyiciler), stored procedure (saklı yordamlar) veya programcı kodlarıyla yapılır.

Tanımlanabilir veri bütünlüğü, prosedürel veri bütünlüğüne göre daha kullanışlı ve denetlenebilir. Ancak, tanımlanabilir veri bütünlüğünün kullanılmadığı durumlarda prosedürel veri bütünlüğü kullanılır.

### 1.1. Görünümler (View)

Görünüm (view), sorguları basitleştirmek, erişim izinlerini düzenlemek, farklı sunuculardaki benzer verileri karşılaştırmak ve bazen sorgu süresini kısaltmak için kullanılan, aslında var olmayan, SELECT ifadesi ile tanımlanmış sanal tablolardır.

Görünümler:

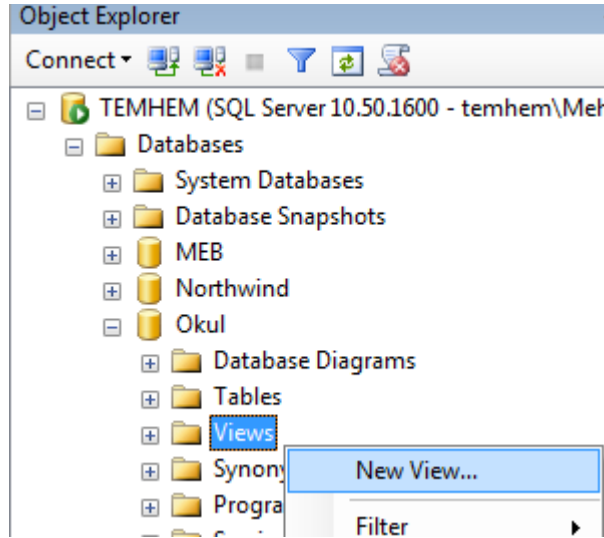
Karmaşık sorguları basitleştirmek,  
Sorgu süresini kısaltmak ve ağ üzerindeki trafiği düşürmek,  
Erişim izinlerini düzenlemek,  
Farklı sunuculardaki benzer verileri karşılaştırmak  
için kullanılır.

### 1.1.1. Yeni Görünüm Oluşturma

Görünüm (view) oluşturmak için Object Explorer penceresinden veya T-SQL komutlarından faydalanabiliriz.

**Object Explorer penceresinde New View seçeneği ile;**

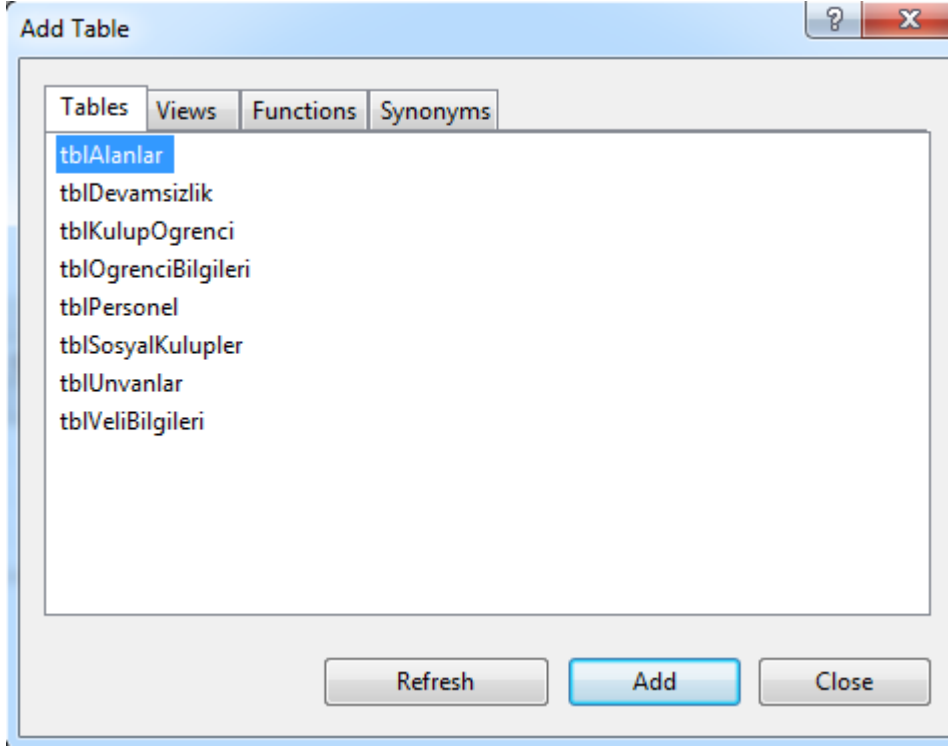
**Object Explorer** penceresinde çalışacağımız veritabanı düğümünü açıyoruz. Alt düğümlerden **Views** düğümü üzerinde sağ tuşa tıklayarak açılan menüden **New View** seçeneğini tıklıyoruz.



**Resim 1.1: Object Explorer penceresi**

Karışımıza veritabanı üzerinde tanımlanmış tabloların listesinin bulunduğu **Add Table** penceresi gelecektir. Bu pencereden hangi tablo üzerinde işlem yapacaksak onu seçerek **Add** butonuna basıyoruz.

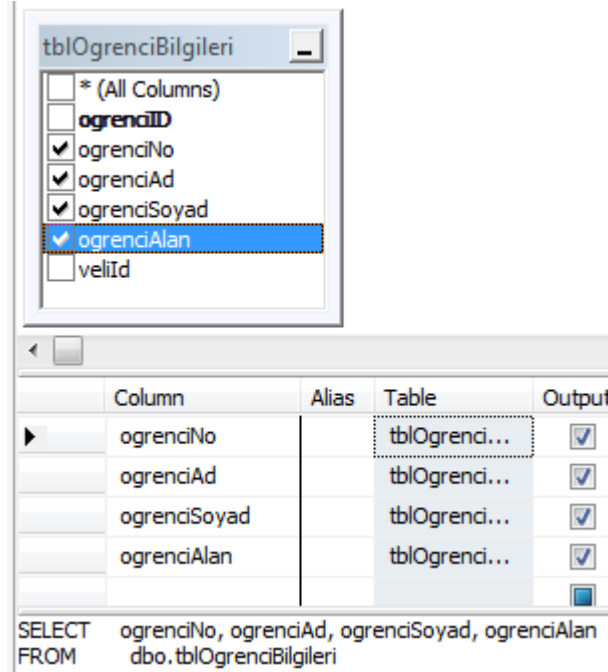




**Resim1.2: Add Table penceresi**

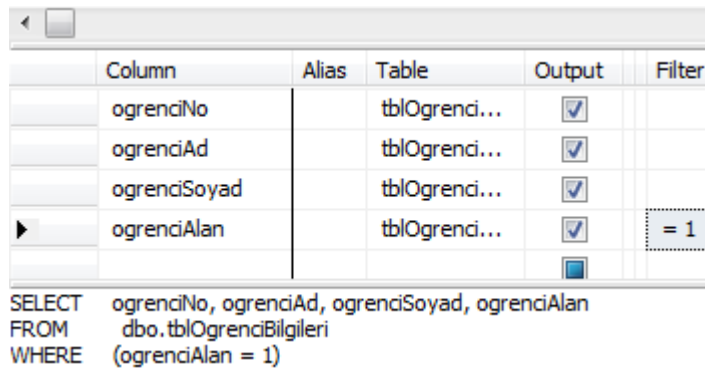
Biz Bilişim Teknolojileri alanındaki öğrencileri listeleyen bir görünüm oluşturacağımızı düşünelim. Dolayısıyla tablo listesinden tblOgrenciBilgileri tablosunu seçelim. Tabloyu seçip Add butonuna tıkladıktan sonra pencereyi kapatmak için Close butonuna tıklayalım.

Eklediğimiz tabloda varolan ve görünüm(view) içinde kullanacağımız alanları seçiyoruz. Alan adları yanındaki kutucuklara işaret koydukca **select** ifadesinin otomatik olarak oluştuğunu göreceksiniz.



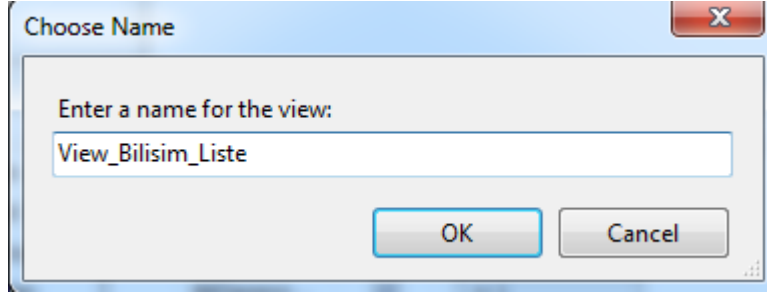
**Resim 1.3: Tablo alanları seçimi penceresi**

Bilişim Teknolojileri alanındaki öğrencileri listeleyeceğimize göre ogrenciAlan sütununun **Filter** özelliğine 1(tblAlanlar tablosunda Bilişim Teknolojileri alanına karşılık gelen alanID değeri kaç ise) değerini giriyoruz. Biz bu değeri girdikten sonra **WHERE** cümlesinin otomatik olarak oluşturulduğuna dikkat etmeliyiz.



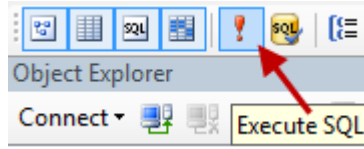
**Resim 1.4: Seçilen alanların görünüm ve filtre uygulama penceresi**

Tüm bu işlemleri tamamladıktan sonra **kaydet** butonuna tıklayıp görünümümüze bir ad vererek kayıt işlemini tamamlıyoruz.



**Resim 1.5: Görünüme ad verme penceresi**

**View Designer** araç çubuğu üzerindeki **Execute SQL** butonuna basarak oluşturduğumuz görünümün sonucunu görebilirsiniz.



**Resim 1.6: Execute SQL butonu**

## T-SQL komutları ile

Genel kullanımı aşağıdaki gibidir.

```
CREATE VIEW view_adi  
AS  
SELECT sütun_adlari  
FROM tablo_adi
```

Object Explorer ile gerçekleştirdiğimiz örneği T-SQL komutları ile de uygulayalım:

```
CREATE VIEW View_Bilisim_Liste_TSQL  
AS  
SELECT ogrenciNo, ogrenciAd, ogrenciSoyad, ogrenciAlan  
FROM dbo.tblOgrenciBilgileri  
WHERE (ogrenciAlan = 1)
```

Komut yapımızı oluşturduktan sonra Execute butonuna basarak görünümümüzün oluşturulmasını sağlıyoruz.



**Resim 1.7: Execute butonu**

Oluşturduğumuz bu görünümü çalıştırıp sonucu görmek için yeni bir sorgu penceresi açıyoruz. Sorgu penceremize aşağıdaki SQL cümlesini yazıp Execute butonuna basıyoruz.

```
SELECT * FROM View_Bilisim_Liste_TSQL
```

	ogrenciNo	ogrenciAd	ogrenciSoyad	ogrenciAlan
1	121	Ahmet	Güneş	1
2	122	Mustafa	Yıldız	1

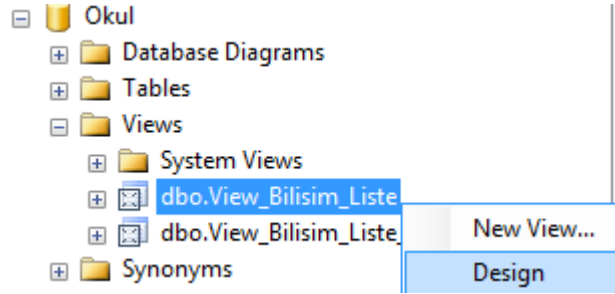
Resim 1.8: Görünüm sorugusu sonuç ekranı

### 1.1.2. Görünümü Güncelleme

Daha önce oluşturduğumuz bir görünümü güncellemek için de Object Explorer penceresinden veya T-SQL komutlarından faydalanabiliriz.

#### Object Explorer penceresinde Design seçeneği ile

**Views** düğümü altında listelenen görünümlerden güncellemek istediğimizin üstünde sağ tuşa tıklayarak açılan menüden **Design** seçeneğini seçiyoruz.



Resim 1.9: Object Explorer penceresi Görünüm listesi

Karşımıza görünümü oluştururken ki pencere gelecektir. Gerekli düzenlemeleri yaptıktan sonra güncellemenin geçerli olması için kaydet butonuna tıklayarak görünümü kayıt ediyoruz.

#### T-SQL komutları ile

Genel kullanımı aşağıdaki gibidir:

```
ALTER VIEW view_adi
WITH secenekler
AS
SELECT sütun_adlari
```

Daha önce oluşturduğumuz View\_Bilisim\_Liste\_TSQL görünümünde Bilişim Teknolojileri alanında okuyan öğrenciler listeleniyordu. Görünümümüzü öğrenci numarası 150 den küçük olanları listeleyecek şekilde değiştirelim.

```
ALTER VIEW View_Bilisim_Liste_TSQL|
AS
SELECT ogrenciNo, ogrenciAd, ogrenciSoyad, ogrenciAlan
FROM dbo.tblOgrenciBilgileri
WHERE (ogrenciNo < 150)
```

Komut yapımızı oluşturduktan sonra Execute butonuna tıklayarak görünümümüzün güncellenmesini sağlıyoruz.

### 1.1.3. Görünümü Silme

#### Object Explorer penceresinde Delete seçeneği ile

Views düğümü altında listelenen görünümlerden silmek istediğimizin üzerinde sağ tuşa tıklıyoruz. Açılan menüden **Delete** seçeneğini seçiyoruz. Karşımıza çıkan pencerede OK butonuna tıklayarak silme işlemini gerçekleştiriyoruz.

#### T-SQL komutları ile

Genel kullanımı aşağıdaki gibidir:

```
DROP VIEW view_adi
```

Daha önce oluşturduğumuz View\_Bilisim\_Liste\_TSQL görünümünü silelim.

```
DROP VIEW View_Bilisim_Liste_TSQL
```

## 1.2. Tetikleyiciler (Trigger)

Bir tabloda yapılan değişikliğin, diğer bir tabloyu etkilemesini sağlamak amacıyla kullanılan yapılara **tetikleyici** (trigger) denir. 3 çeşit tetikleyici yapısı vardır:

**Ekleme tetikleyicisi (INSERT Trigger):** Bir tabloda ekleme işlemi yapıldığında diğer tablonun bu işlemten etkilenmesi istendiğinden kullanılır.

**Silme tetikleyicisi (DELETE Trigger):** Bir tabloda silme işlemi gerçekleştirildiğinde diğer tablonun bu işlemten etkilenmesi istendiğinde kullanılır.

**Güncelleme tetikleyicisi (UPDATE Trigger):** Bir tabloda güncelleme işlemi gerçekleştirildiğinde diğer tablonun bu işlemten etkilenmesi istendiğinde kullanılır.

## 1.2.1. Yeni Tetikleyici Oluşturma

Genel yapısı aşağıdaki gibidir.

```
CREATE TRIGGER tetikleyici_adi
ON tablo_adi
FOR INSERT,DELETE,UPDATE /*Trigger Türü*/
AS
BEGIN
/*Yazmak istediğimiz sql komutları*/
END
```

### Ekleme tetikleyicisi oluşturma (INSERT Trigger)

Tetikleyici oluşturacağımız tabloda herhangi bir kayıt ekleme işlemi olursa bir başka tablonun da bu kayıttan etkilenmesini istediğimiz durumlarda kullanılır.

#### Örnek:

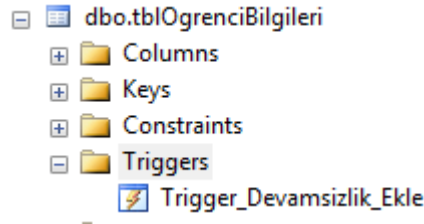
Daha önceki modüllerde oluşturduğumuz tablolar üzerinde işlemler yapacağız. Öğrenci bilgilerini tuttuğumuz tblOgrenciBilgileri tablomuzaya yeni bir öğrenci kaydı eklediğimizde öğrenci devamsızlıklarının tutulduğu tblDevamsizlik tablosunda da bu öğrenciye ait bir kaydın olması istenmektedir. Baktığımızda iki işlem gibi görünen bu işlemi tetikleyici aracılığıyla tek işleme düşürebiliriz.

```
CREATE TRIGGER Trigger_Devamsizlik_Ekle
ON tblOgrenciBilgileri
FOR INSERT
AS
BEGIN
DECLARE @ogrenciID int
SELECT @ogrenciID = ogrenciID FROM inserted
INSERT INTO tblDevamsizlik(ogrenciId, devamsizlik) VALUES (@ogrenciID, 0)
END
```

Kod	Açıklama
➤ CREATE TRIGGER Trigger_Devamsizlik_Ekle	➤ Trigger_Devamsizlik_Ekle adında yeni bir tetikleyici tanımlıyoruz.
➤ ON tblOgrenciBilgileri	➤ Tetikleyicimizin tetikleneceği tabloyu belirtiyoruz.
➤ FOR INSERT	➤ Tetikleyici türünün kayıt ekleme olduğunu belirtiyoruz. SQL Server kayıt ekleme anında <b>inserted</b> adında bir tablo oluşturur.
➤ DECLARE @ogrenciID int	➤ int tipinde bir değişken tanımlıyoruz.
➤ SELECT @ogrenciID = ogrenciID FROM inserted	➤ Kayıt ekleme esnasında tablomuzun ogrenciID sütununa atanan değeri tanımlamış olduğumuz değişken üzerine aktarıyoruz.
➤ INSERT INTO	➤ tblOgrenciBilgileri tablomuzaya yeni bir

tblDevamsizlik(ogrenciId, devamsizlik) VALUES (@ogrenciID, 0)	kayıt eklendiği zaman tblDevamsizlik tablosuna da kayıta ilgili yeni bir kayıt ekliyoruz.
--	---

Kod bloğumuzu tamamladıktan sonra Execute komutu ile çalıştırıyoruz. Tetikleyicimiz oluşturulduktan sonra tblOgrenciBilgileri tablomuzda herhangi bir öğrenci kayıt ettiğimizde otomatik olarak tblDevamsizliklar tablosunda da eklenen öğrenciye ait bir kayıt oluşturulacaktır. Bir tablo için oluşturulmuş tetikleyicilerin listesini o tablo altındaki düğümlerden Triggers düğümü altında görebilirsiniz.



**Resim 1.10: Object Explorer penceresi Tetikleyici listesi**

Şimdi yeni bir sorgu penceresi açıp tblOgrencibilgileri tablomuzda yeni bir kayıt ekleyelim ve sonucu görelim.

```
INSERT INTO tblOgrenciBilgileri (ogrenciNo,ogrenciAd,ogrenciSoyad,ogrenciAlan,veliId)
VALUES (201,'Aslan','Güneş',1,1)
```

	ogrenciID	ogrenciNo	ogrenciAd	ogrenciSoyad	ogrenciAlan	veliId
1	1	121	Ahmet	Güneş	1	1
2	2	122	Mustafa	Yıldız	1	2
3	3	123	Ayşe	Güneş	2	1
4	5	200	Fatma	Yıldız	2	2

**Resim 1.11: tblOgrenciBilgileri tablosunun kayıt eklenmeden önceki durumu**

	ogrenciID	ogrenciNo	ogrenciAd	ogrenciSoyad	ogrenciAlan	veliId
1	1	121	Ahmet	Güneş	1	1
2	2	122	Mustafa	Yıldız	1	2
3	3	123	Ayşe	Güneş	2	1
4	5	200	Fatma	Yıldız	2	2
5	6	201	Aslan	Güneş	1	1

**Resim 1.12: tblOgrenciBilgileri tablosunun kayıt eklendikten sonraki durumu**

	devamID	ogrenciid	devamsizlik
1	1	1	5
2	2	2	7
3	4	3	4
4	5	5	12

**Resim 1.13: tblDevamsizlik tablosunun kayıt eklenmeden önceki durumu**

	devamID	ogrenciid	devamsizlik
1	1	1	5
2	2	2	7
3	4	3	4
4	5	5	12
5	6	6	0

**Resim 1.14: tblDevamsizlik tablosunun kayıt eklendikten sonraki durumu**

### **Silme tetikleyicisi oluşturma (DELETE Trigger)**

Tetikleyici oluşturacağımız tabloda herhangi bir kayıt silme işlemi olursa bir başka tablonun da bu işlemten etkilenmesini istediğimiz durumlarda kullanılır.

#### **Örnek:**

tblOgrenciBilgileri tablomuzda yer alan bir öğrencinin okulla ilişkisinin kesildiğini ve bilgilerinin tablodan silindiğini düşünelim. Dolayısıyla bu öğrenciye ait devamsızlık ve sosyal kulüp bilgilerinin de silinmesi gerekmektedir. Tek tek tablolarda bu kayıtları arayıp silmek yerine tetikleyici tanımlayarak bu işleri daha kolay halledilebilir bir duruma getirebiliriz.

```
CREATE TRIGGER Trigger_Ogrenci_Sil
ON tblOgrenciBilgileri
FOR DELETE
AS
BEGIN
DECLARE @ogrenciID int
SELECT @ogrenciID = ogrenciID FROM deleted
DELETE FROM tblDevamsizlik WHERE ogrenciId = @ogrenciID
DELETE FROM tblKulupOgrenci WHERE ogrenciId = @ogrenciID
END
```

Kod bloğumuzu oluşturduktan sonra Execute butonuna basarak tetikleyicimizi oluşturuyoruz. tblOgrenciBilgileri tablomuzdan bir kayıt silindiği zaman tblDevamsizlik ve tblKulupOgrenci tablolarımızda da bu kayıtlarla ilgili kayıtlar silinecektir.



## Güncelleme tetikleyicisi oluşturma (UPDATE Trigger)

Tetikleyici oluşturacağımız tabloda herhangi bir kayıt güncelleme işlemi olursa bir başka tablonun da bu işlemten etkilenmesini istediğimiz durumlarda kullanılır.

### Örnek:

Veli bilgilerinin tutulduğu tblVeliBilgileri tablosunda bir velinin soyadının değiştirildiğini düşünelim. Öğrenci bilgilerinin tutulduğu kayıtlarda bu veliyle ilişkili öğrencilerin soyadının da değiştirilmesi gerektiğini farz edelim.

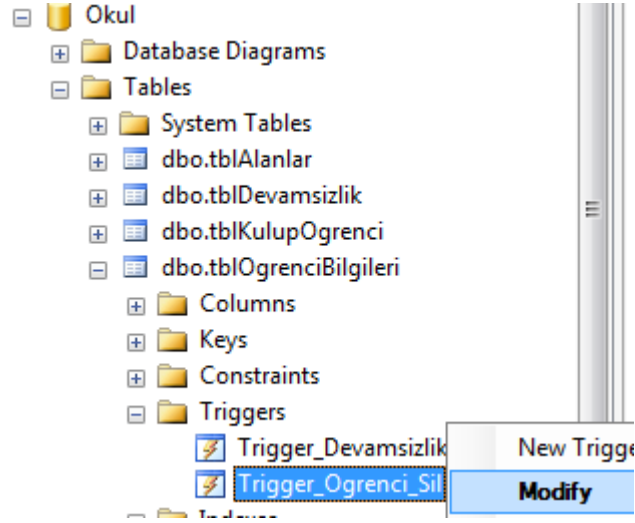
```
CREATE TRIGGER Trigger_Veli_Guncelle ON tblVeliBilgileri
FOR UPDATE
AS
BEGIN
DECLARE @degisenSoyad nvarchar(50), @veliId int
SELECT @veliId = veliID from deleted
SELECT @degisenSoyad = veliSoyad from inserted
UPDATE tblOgrenciBilgileri SET ogrenciSoyad = @degisenSoyad WHERE veliId = @veliId
END
```

Burada bilmeniz gereken update triggerin insert ve delete triggerdan biraz daha farklı çalışmasıdır. Update triggerda direk bir güncelleme işlemi olmaz. Güncelleme yapabilmek için önce güncellenen tabloyu siler ve daha sonra güncellenmiş şekli ile tabloyu tekrar ekler.

### 1.2.2. Tetikleyiciyi Düzenleme

Düzenlemek istediğimiz tetikleyici üzerinde sağ tuşa tıklıyoruz. Açılan menüden **Modify** seçeneğini seçiyoruz. Tetikleyicimize ait SQL komutların bulunduğu yeni bir sorgu penceresi açılacaktır. Burada gerekli düzenlemeleri yaptıktan sonra Execute butonuna basarak düzenlemenin geçerli olmasını sağlamalıyız.

Modify (düzenle) komutunu verdiğinizde ise tetikleyici, ALTER komutu ile açılacaktır. Daha önceden de bildiğiniz gibi ALTER, var olan tablo, tetikleyici gibi nesnelere üzerinde değişiklik yapmak için kullanılan komuttur. Daha önceden oluşturulmuş tetikleyiciyi düzenlerken yeni bir sorgu sayfası açıp Modify komutunu kullanmadan da ALTER komutu ile tetikleyiciyi yeniden yazarak düzenleyebilirsiniz. Ancak bu önerilmez.



Resim 1.15: Object Explorer penceresi Tetikleyici düzenleme

### 1.2.3. Tetikleyici Silme

#### Object Explorer penceresinde Delete seçeneği ile;

Triggers düğümü altında listelenen tetikleyicilerden silmek istediğimizin üzerinde sağ tuşa tıklıyoruz. Açılan menüden **Delete** seçeneğini seçiyoruz. Karşımıza çıkan pencerede OK butonuna tıklayarak silme işlemini gerçekleştiriyoruz.

#### T-SQL komutları ile;

Genel kullanımı aşağıdaki gibidir.

```
DROP TRIGGER tetikleyici_adi
```

Daha önce oluşturduğumuz Trigger\_Devamsizlik\_Ekle tetikleyicisini silelim.

```
DROP TRIGGER Trigger_Devamsizlik_Ekle
```

### 1.3. Saklı Yordamlar (Stored Procedures)

Stored Procedure SQL Server üzerinde önceden derlenmiş olup saklanan SQL ifadeleridir. Önceden derlenmiş olarak bulduklarından her türlü sorgulamada en iyi performansı verirler. SQL Server' da sistem tarafından "sp\_" ile başlayan isimlerle tanımlanmış bir çok yordam mevcuttur. Bunlar daha çok yönetim amaçlı olarak sistem tablolarından bilgi toplamak için kullanılırlar.

Saklı yordamlar;

- Parametre alabilirler,
- Başka altyordamları çağırabilirler,

- Kendisini çağıran bir altıordam veya toplu işleme başarılı olduğunu ya da olmadığını, hata oluşması durumunda hatanın nedenini bir durum değeri olarak döndürebilirler.

Parametrelerin değerlerini kendisini çağıran bir altıordam döndürebilirler.

Altıordamların oluşturulmasının nedeni, sıkça yapılan işlemlerin bir defa yazılarak program akışına göre tekrar tekrar kullanılmasını sağlamaktır. Böylece, kod yazımı ve programlama kolaylaştırılmış olur.

Saklı yordamlar diğer programlama dillerindeki fonksiyonlara (function) denk gelmektedir. Oluşturulan bir saklı yordama ana programdan bir komut ile ulaşılabilir.

### 1.3.1. Saklı Yordam Oluşturmak

Stored Procedure'ün oluşturulma şekli aşağıdaki gibidir.

```
CREATE PROC [ EDURE ] prosedür_adı  
AS  
T-SQL ifadeleri  
GO
```

Oluşturacağınız saklı yordamlar ile sistemin saklı yordamlarının karışmamasına dikkat ediniz.

Saklı yordam oluşturabilmek için **sysadmin**, **db\_owner** veya **dll\_admin** rolüne sahip olmanız gerekir.

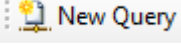
Bir saklı yordam; CREATE DEFAULT, CREATE PROCEDURE, CREATE RULE, CREATE TRIGGER ve CREATE VIEW ifadelerini içeremez. Ancak, her nesneden veri alabilir.

#### Örnek

- tblOgrenciBilgileri tablosunda veliId'si 1 olan öğrencileri listeleyen bir store procedure oluşturalım.

	ogrenciID	ogrenciNo	ogrenciAd	ogrenciSoyad	ogrenciAlan	veliId
1	1	121	Ahmet	Güneş	1	1
2	2	122	Mustafa	Yıldız	1	2
3	3	123	Ayşe	Güneş	2	1
4	5	200	Fatma	Yıldız	2	2

Resim 1.16: tblOgrenciBilgileri tablosunda bulunan kayıtların listesi

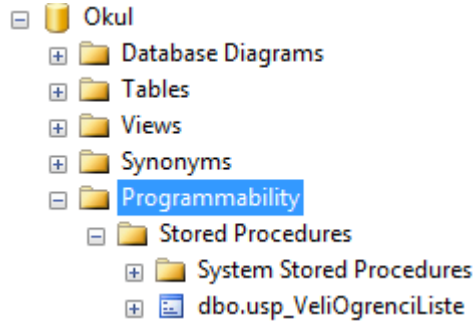
 butonuna tıklayarak yeni bir sorgu penceresi açalım. Sorgu penceresinde SQL kod bloğumuzu aşağıdaki gibi oluşturalım.

```
CREATE PROC usp_VeliOgrenciListe
AS
    SELECT * FROM tblOgrenciBilgileri WHERE veliId = 1
GO
```

Şimdi Execute butonuna tıklayarak veya klavyeden F5 tuşuna basarak sorgumuzu çalıştıralım.

Çalıştırdıktan sonra sonuç penceresinde herhangi bir liste oluşmayacaktır. Biz burada sadece ağ veritabanı yazılımımıza yeni bir store procedure tanımladık.

Tanımlamış olduğumuz store procedure listesine **Programmability** düğümü altında yer alan **Stored Procedures** düğümünden erişebiliriz.



**Resim 1.17: Object Explorer penceresi Saklı Yordam listesi**

Tanımlamış olduğumuz saklı yordamı çalıştırmak için **EXEC** komutunu kullanırız. Şimdi tanımladığımız usp\_VeliOgrenciListe saklı yordamını çağıralım ve sonucunu görelim.

Yeni bir sorgu penceresi açalım. Sorgu penceremize aşağıdaki komutu yazalım.

```
EXEC usp_VeliOgrenciListe
```

Execute butonuna tıklayarak veya klavyeden F5 tuşuna basarak sorgumuzu çalıştıralım. Sorgumuzu çalıştırdıktan sonra sonuç penceresinde aşağıdaki gibi bir liste oluşacaktır.

	ogrenciID	ogrenciNo	ogrenciAd	ogrenciSoyad	ogrenciAlan	veliId
1	1	121	Ahmet	Güneş	1	1
2	3	123	Ayşe	Güneş	2	1

**Resim 1.18: usp\_VeliOgrenciListe Saklı Yordamı sonuç ekranı**

### 1.3.2. Saklı Yordamlarda Değişiklik Yapmak

Saklı yordamlarda değişiklik yapmak için **ALTER** komutu kullanılır.

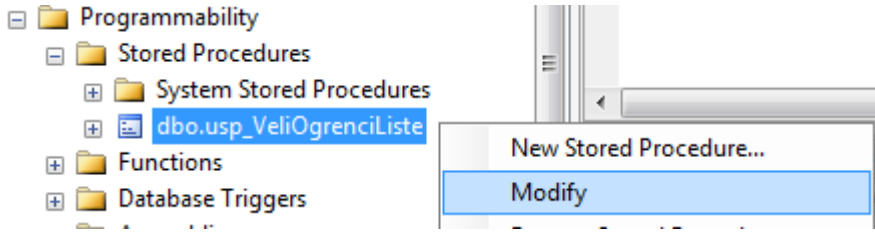
## Genel Kullanımı

```
ALTER PROC [EDURE] prosedür_adi  
AS  
    T-SQL ifadeleri  
GO
```

Bir Stored Procedure’de değişiklik yapabilmek için önce kaynak kodunun alınması ve bir Query ekranına kopyalanıp düzenlenmesi gerekir.

### Örnek:

- usp\_VeliOgrenciListe adı prosedürümüzde veliId’si 1 olan öğrencileri listelemiştik. Bu prosedürü veliId’si 2 olan öğrencileri listeleyecek şekilde değiştirelim.
- Oluşturduğunuz “usp\_VeliOgrenciListe” adındaki Stored Procedure üzerinde fareyle sağ tıklıyoruz ve açılan menüden Modify seçeneğini seçiyoruz.

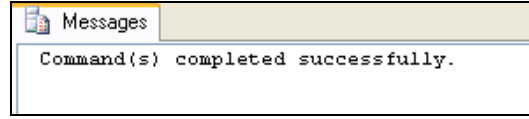


Resim 1.19: Object Explorer penceresi saklı yordam düzenleme

- Modify komutunu verince kodlar bir Query sayfası şeklinde ekrana gelecektir. Üzerinde gerekli değişiklikleri yapınız ve Execute (F5) ediniz.

```
USE [Okul]  
GO  
/***** Object: StoredProcedure [dbo].[usp_VeliOgrenciList  
SET ANSI_NULLS ON  
GO  
SET QUOTED_IDENTIFIER ON  
GO  
ALTER PROC [dbo].[usp_VeliOgrenciListe]  
AS  
    SELECT * FROM tblOgrenciBilgileri WHERE veliId = 2  
|
```

- Execute işleminin sorunsuz bir şekilde gerçekleştiğini Messages penceresinde görebilirsiniz.



**Resim 1.20: Mesaj penceresi**

- Yeni bir Query ekranı açınız ve Stored Procedure'ün çalışması için gerekli kodu yazınız.

```
EXEC usp_VeliOgrenciListe
```

- usp\_VeliOgrenciListe stored procedure'ü işletilecek ve sonuçlar Results penceresinde size gösterilecektir.

	ogrenciD	ogrenciNo	ogrenciAd	ogrenciSoyad	ogrenciAlan	veliId
1	2	122	Mustafa	Yildiz	1	2
2	5	200	Fatma	Yildiz	2	2

**Resim 1.21: usp\_VeliOgrenciListe saklı yordam sonuç ekranı**

### 1.3.3. Saklı Yordamı Silmek

Var olan bir saklı yordamı silmek için **DROP** komutunu kullanmak gereklidir. DROP komutundan sonra saklı yordamın sahibinin adı ve saklı yordamın adı yazılmalıdır.

#### Genel Kullanımı

DROP PROC sahip.prosedür\_adi

#### Örnek

DROP PROC dbo.usp\_VeliOgrenciBilgileri

### 1.3.4. Değer Alan Alt Yordamlar

Stored Procedure'lerin daha etkin kullanılabilmesi ve işlevsel bir hale gelebilmesi için dışarıdan değer almalarına ihtiyaç duyulur. Bu nedenle girdi parametreleri (Input Parameter) kullanılır.

Stored Procedure'nin aldığı değer Query'den gelen değerdir. Gönderilen değeri karşılayacak bir değişken Stored Procedure'de tanımlanmalıdır.

#### Örnek

- Daha önce oluşturduğumuz “usp\_VeliOgrenciListe” stored procedure'ünde veliId bilgisini kendimiz yazıyorduk. Şimdi ise veliId bilgisini dışarıdan veri olarak göndereyim.

```

CREATE PROC usp_VeliOgrenciListeInputValue
@veliId int
AS
SELECT * FROM tblOgrenciBilgileri WHERE veliId = @veliId
GO

```

- veliId adında bir değişken tanımladık. Sorgumuzda karşılaştırma yaparken bu değeri kullandık. “usp\_VeliOgrenciListeInputValue” stored procedure’ümüzü çağıralım.

veliId değeri 1 verildiğinde;

```
EXEC usp_VeliOgrenciListeInputValue 1
```

Sonuç;

	ogrenciID	ogrenciNo	ogrenciAd	ogrenciSoyad	ogrenciAlan	veliId
1	1	121	Ahmet	Güneş	1	1
2	3	123	Ayşe	Güneş	2	1

**Resim 1.22: usp\_VeliOgrenciListeInputValue Saklı Yordam sonuç ekranı**

veliId değeri 2 verildiğinde;

```
EXEC usp_VeliOgrenciListeInputValue 2
```

Sonuç;

	ogrenciID	ogrenciNo	ogrenciAd	ogrenciSoyad	ogrenciAlan	veliId
1	2	122	Mustafa	Yıldız	1	2
2	5	200	Fatma	Yıldız	2	2

**Resim 1.23: usp\_VeliOgrenciListeInputValue saklı yordam sonuç ekranı**

Stored Procedure’e gelen parametrelerin isteğe bağlı olması istenebilir. Bu gibi durumlarda Stored Procedure’de tanımlanan değişkene default değer atanması yapılır. Stored Procedure’e gelen parametreye değer atanmazsa, tanımlanan değişkene atanan default değer işleme tabi tutulur. Stored Procedure’de bir default değer atayacaksanız bu değer, bir sabit olması gerekir.

## Örnek

- Bir sınıftaki öğrencilerin bilgilerinin tutulduğu bir tabloda öğrenci adlarının içerisinde “a” harfi geçen öğrencileri gösteren bir Stored Procedure şöyle yazılmalıdır.

	ogrenciID	ogrenciNo	ogrenciAd	ogrenciSoyad	ogrenciAlan	velid
1	1	121	Ahmet	Güneş	1	1
2	2	122	Mustafa	Yıldız	1	2
3	3	123	Ayşe	Güneş	2	1
4	5	200	Fatma	Yıldız	2	2

Resim 1.24: tblOgrenciBilgileri tablosundaki kayıtların listesi

```
CREATE PROC usp_OgrenciAra
@ara nvarchar(50)=NULL
AS
    IF @ara IS NOT NULL
    SELECT * FROM tblOgrenciBilgileri WHERE
    ogrenciAd = '%' + @ara + '%'
GO
```

- Stored Procedure’de tanımlanan “ara” değişkeni Query’den bir değer gelmese de NULL değerini alacak ve işlem gerçekleşecektir.
- Query’den öğrenci adlarının içerisinde “a” harfi olan öğrencileri görmek için Reaşağıdaki gibi kod satırını yazınız ve çalıştırınız.

Ogr\_Ara 'a'

- **Ogr\_Ara ‘a’** şeklindeki yazım ile de Stored Procedure çalışacaktır.
- Sonuçları Results penceresinde görebilirsiniz.

	Ogr_No	Ad	Soyad	Cinsiyet	Yas
1	101	Hasan	KAYA	E	17
2	104	Ayşe	KOCAER	K	17
3	106	Fatma	YILMAZ	K	18

Resim 1.25: Sonuçların gösterilmesi

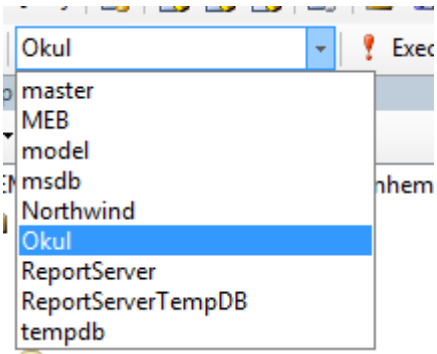
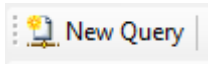
- Stored Procedure’e değer gönderilmeseydi Messages penceresinde “Command(s) completed successfully.” mesajını görecektiniz.

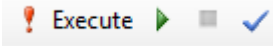
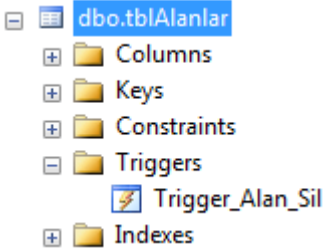
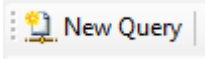
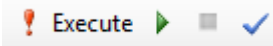


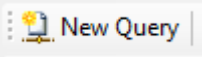
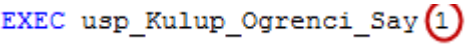
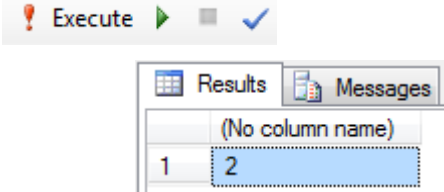
## UYGULAMA FAALİYETİ

Aşağıdaki sorguları oluşturunuz.

- tblAlanlar tablosundan bir alan silindiği zaman o alana ait öğrencileri de silen tetikleyiciyi oluşturunuz.
- kulupId değeri gönderilen kulübe ait kaç öğrenci kayıtlı olduğunu döndüren saklı yordamı oluşturunuz.

İşlem Basamakları	Açıklama
➤ Ağ veritabanı sunucumuza bağlanınız.	
➤ Veritabanı seçim kutusundan Okul veritabanımızı seçiniz.	
➤ Yeni bir sorgu penceresi açınız.	
➤ Tetikleyiciyi oluşturacak komutumuzu yazıp tetikleyici adını belirleyiniz.	➤ <b>CREATE TRIGGER</b> Trigger_Alan_Sil
➤ Tetikleyicimizin hangi tablo üzerinde tetikleneceğini belirleyiniz.	➤ <b>ON</b> tblAlanlar
➤ Tetikleyici türümüzü belirleyiniz.	➤ <b>FOR DELETE</b>
➤ Silinen kaydın ID değerini tutacak değişkenimizi ve veri tipini tanımlayınız.	➤ <b>DECLARE</b> @d_AlanID int
➤ Silinen kaydın ID değerini değişkenimiz üzerine aktarınız.	➤ <b>SELECT</b> @d_AlanID = alanID <b>FROM</b> deleted
➤ Silinen alana ait öğrencileri tblOgrenciBilgileri tablosundan da silen sorgumuzu yazınız.	➤ <b>DELETE FROM</b> tblOgrenciBilgileri <b>WHERE</b> ogrenciAlan = @d_AlanID

<pre>CREATE TRIGGER Trigger_Alan_Sil ON tblAlanlar FOR DELETE AS BEGIN DECLARE @d_AlanID int SELECT @d_AlanID = alanID FROM deleted DELETE FROM tblOgrenciBilgileri WHERE ogrenciAlan = @d_AlanID END</pre>	
<p>➤ Execute butonuna tıklayarak sorgumuzu çalıştırıp tetikleyicimizi oluşturunuz.</p>	
<p>➤ Tablo üzerinde sağ tuşa tıklayarak açılan menüden Refresh seçeneğini seçiniz. Tetikleyicimizin Triggers düğümü altında oluşturulduğunu görürüz.</p>	
<p>➤ Yeni bir sorgu penceresi açınız.</p>	
<p>➤ Saklı yordamımızı oluşturacak komutu yazıp saklı yordam adını belirleyiniz.</p>	<p>➤ <b>CREATE PROC</b> usp_Kulup_Ogrenci_Say</p>
<p>➤ Parametre olarak gönderilen kulupId değerini tutacak değişkeni yazınız.</p>	<p>( ➤ @f_KulupId int )</p>
<p>➤ Gelen değere karşılık tblKulupOgrenci tablosundaki öğrenci sayısını bulacak sorgumuzu yazınız.</p>	<p>➤ <b>SELECT</b> <b>COUNT(*)</b> <b>FROM</b> tblKulupOgrenci <b>WHERE</b> kulupId = @f_KulupId</p>
<pre>CREATE PROC usp_Kulup_Ogrenci_Say ( @f_KulupId int ) AS SELECT COUNT(*) FROM tblKulupOgrenci WHERE kulupId = @f_KulupId GO</pre>	
<p>➤ Execute butonuna tıklayarak sorgumuzu çalıştırıp saklı yordamımızı oluşturunuz.</p>	

<p>➤ Yeni bir sorgu penceresi açınız.</p>	
<p>➤ Saklı yordamımızı çağırarak sorgumuzu yazıp herhangi bir kulupId değeri gönderiniz.</p>	
<p>➤ Execute butonuna tıklayarak sorgumuzu çalıştırıp sonucunu görünüz.</p>	

## KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Ağ veritabanı sunucusuna bağlanabildiniz mi?		
2. Veritabanı seçim kutusundan üzerinde sorgu oluşturulacak veritabanını seçebildiniz mi?		
3. Yeni bir sorgu penceresi açabildiniz mi?		
4. Tetikleyiciyi oluşturacak komutu yazabildiniz mi?		
5. Tetikleyicinin tetikleneceği tabloyu belirleyebildiniz mi?		
6. Tetikleyici türünü belirleyebildiniz mi?		
7. Değişken tanımlayabildiniz mi?		
8. Silinen kaydın ID değerini değişken üzerine aktarabildiniz mi?		
9. Tetiklenecek sorguyu oluşturabildiniz mi?		
10.Sorguyu çalıştırıp hatasız bir şekilde tetikleyiciyi oluşturabildiniz mi?		
11.Saklı yordamı oluşturacak komutu yazabildiniz mi?		
12.Saklı yordama gönderilecek parametre değerini tutacak değişkeni ve türünü yazabildiniz mi?		
13.Sonucu oluşturacak sorguyu oluşturabildiniz mi?		
14.Soruguyu çalıştırıp hatasız bir şekilde saklı yordamı oluşturabildiniz mi?		
15.Saklı yordamı çağırabildiniz mi?		

## DEĞERLENDİRME

Değerlendirme sonunda “**Hayır**” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “**Evet**” ise “Ölçme ve Değerlendirme”ye geçiniz.

## ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi SELECT ifadesi ile tanımlanmış sanal tablolardır?  
A) Trigger  
B) Table  
C) Defaults  
D) View
2. Görünümleri düzenlemek için sağ tuş menüsünden hangisi seçilmelidir?  
A) Design  
B) Edit  
C) Modify  
D) Repair
3. Bir tabloda ekleme işlemi yapıldığında diğer bir tablonun etkilenmesi istendiği durumlarda kullanılan tetikleyici türü aşağıdakilerden hangisidir?  
A) DELETE TRIGGER  
B) UPDATE TRIGGER  
C) INSERT TRIGGER  
D) ALTER TRIGGER
4. Yeni bir kayıt eklenirken ağ veritabanı yazılımı tarafından otomatik oluşturulan tablo aşağıdakilerden hangisidir?  
A) deleted  
B) inserted  
C) insert  
D) edited
5. Önceden tanımlanmış bir tetikleyiciyi silmek için aşağıdaki komutlardan hangisi kullanılmalıdır?  
A) DELETE TRIGGER  
B) DROP TRIGGER  
C) ALTER TRIGGER  
D) SIL TRIGGER

## DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

# ÖĞRENME FAALİYETİ-2

## AMAÇ

Bu öğrenme faaliyetinde, kısıtlayıcıları kullanarak ve rol oluşturarak veri tutarlılığını sağlayabileceksiniz.

## ARAŞTIRMA

- Veri tutarlılığının neden önemli olduğunu araştırınız.
- Veri tutarlılığının sağlanması için yapılabilecek veritabanı işlemlerini araştırınız.

## 2. VERİ TUTARLILIĞI

### 2.1. Kısıtlayıcı (Constraint)

Veri üzerindeki mantıksal sınırlamalara **kısıt** adı verilir.

Kısıtlar, veri modellerinde bütünlük sağlamak için kullanılır. Kısıtlamalar, tabloların tanımlanmasıyla beraber oluşan öğelerdir. Kısıtlamalar ile Rule (kural) ve Default'ların (varsayılan) yapabileceği işler yapılabilir.

Kısıtlamalar, tablo oluştururken yani CREATE TABLE komutuyla birlikte tanımlanabilir. Tablo oluşturulmuşsa ALTER TABLE komutuyla bu işlem gerçekleşir. ALTER TABLE komutuyla kullanıldığında sütunlara girilen bilgilerin dikkate alınması gerekir.

**sp\_helpconstraint** saklı yordamını kullanarak istenilen nesnenin kısıtlama bilgileri elde edilebilir.

"**sp\_helpconstraint object\_name**" şeklinde kullanılır.

sp\_helpconstraint tblOgrenciBilgileri

Object Name	constraint_type	constraint_name	delete_a...	update_...	status_...	status_for_replicat...	constraint_keys
1	DEFAULT on column ve...	DF_tblOgrenciBilgileri_veliId	(n/a)	(n/a)	(n/a)	(n/a)	((0))
2	FOREIGN KEY	FK_tblOgrenciBilgileri_tblAlanlar	No Action	No Action	Enabled	Is_For_Replication	ogrenciAlan
3							REFERENCES Okul.dbo.tblAlanlar (alanID)
4	FOREIGN KEY	FK_tblOgrenciBilgileri_tblVeliBilgileri	No Action	No Action	Enabled	Is_For_Replication	veliId
5							REFERENCES Okul.dbo.tblVeliBilgileri (veliID)
6	PRIMARY KEY (clustered)	PK_tblOgrenciBilgileri	(n/a)	(n/a)	(n/a)	(n/a)	ogrenciID

Table is referenced by foreign key	
1	Okul.dbo.tblDevamsizlik: FK_tblDevamsizlik_tblOgrenciBilgileri
2	Okul.dbo.tblKulupOgrenci: FK_tblKulupOgrenci_tblOgrenciBilgileri

Resim 2.1: sp\_helpconstraint Saklı Yordam sonuç ekranı

## 2.2. Kısıtlayıcı Türleri

Kısıtlayıcı türleri aşağıda maddeler halinde sıralanmıştır.

### 2.2.1. Birincil Anahtar Kısıtlayıcı (Primary Key Constraint)

Birincil anahtar kısıtlayıcı anlamındadır. Aynı olmayan değerler girilmesini sağlar. Bu da her kaydın farklı olması demektir. Her tablonun en fazla 1 adet Primary Key Constraint'i olabilir.

#### ➤ Yeni bir tablo oluştururken kullanımı:

```
CREATE TABLE tablo_adi (
    Sütun_adları,
    CONSTRAINT const_adi PRIMARY KEY (sütun_adi)
)
```

#### Örnek:

```
CREATE TABLE tblDuyurular(
    duyuruID int not null,
    duyuruBaslik nvarchar(150) not null,
    duyuruIcerik nvarchar(max),
    duyuruTarih date,
    CONSTRAINT PKC_uyuruId PRIMARY KEY(duyuruID)
)
```

#### ➤ Oluşturulmuş olan tabloda kullanımı:

```
ALTER TABLE tablo_adi
    ADD CONSTRAINT cons_adi PRIMARY KEY (sütun_adi)
```

### Örnek:

```
ALTER TABLE tblDuyurular
ADD CONSTRAINT PKC_duyuruId PRIMARY KEY(duyuruID)
```

## 2.2.2. Tekil Alan Kısıtlayıcı (Unique Constraint)

Tekil alan kısıtlayıcı anlamındadır. Birincil anahtar olan ve tablodaki diğer alanlar içinde aynı içeriğe sahip verilerin olmaması için Unique Constraint tanımlanır. TcKimlikNo primary key ve OkulNo Unique şeklinde bir tanımlama Unique Constraint'e bir örnektir.

## 2.2.3. Yabancı Anahtar Kısıtlayıcı (Foreign Key Constraint)

Yabancı anahtar kısıtlayıcı anlamındadır. Bir tablodaki bir sütuna ait verilerin başka bir tablonun belirli bir sütunundan gelmesini denetler.

Bir tabloya girilebilecek değerleri başka bir tablonun belli bir alanında yer alabilecek veri grubu ile sınırlandırmaya ve en önemlisi de ilişkilendirmeye yarar.

### ➤ Yeni bir tablo oluştururken kullanımı:

```
CREATE TABLE tablo_adi (
    Sütun_adları,
    CONSTRAINT const_adi FOREIGN KEY (sütun_adi)
    REFERENCES diğertablo_adi (sütun_adi)
)
```

### Örnek:

tblOgrenciBilgileri tablomuzu yeni oluşturuyormuş gibi düşünelim.

```
CREATE TABLE tblOgrenciBilgileri(
    ogrenciID int not null,
    ogrenciNo int not null,
    ogrenciAd nvarchar(50) not null,
    ogrenciSoyad nvarchar(50) not null,
    ogrenciAlan int not null,
    veliId int not null,
    CONSTRAINT FKC_ogrenciAlan FOREIGN KEY(ogrenciAlan)
    REFERENCES tblAlanlar(alanID),
    CONSTRAINT FCK_veliId FOREIGN KEY(veliId)
    REFERENCES tblVeliBilgileri(veliID)
)
```

### ➤ Oluşturulmuş olan tabloda kullanımı:

```
ALTER TABLE tablo_adi
```



```
ADD CONSTRAINT cons_adi FOREIGN KEY (sütun_adi)
REFERENCES diğertablo_adi (sütun_adi)
```

**Örnek:**

```
ALTER TABLE tblOgrenciBilgileri
ADD CONSTRAINT FKC_ogrenciAlan FOREIGN KEY(ogrenciAlan)
REFERENCES tblAlanlar(alanID),
CONSTRAINT FCK_veliId FOREIGN KEY(veliId)
REFERENCES tblVeliBilgileri(veliID)
```

#### 2.2.4. Varsayılan Kısıtlayıcı (Default Constraint)

Varsayılan kısıtlayıcı anlamındadır. Tablodaki herhangi bir alan için girilmesi gereken bir değerin atanmasıdır. INSERT komutu için geçerlidir. Örneğin, öğrenci devamsızlıklarının tutulduğu tblDevamsizlik tablosuna yeni bir öğrenci eklendiğinde devamsizlik sütununa varsayılan değer olarak 0 atanabilir.

➤ **Yeni bir tablo oluşturulurken kullanımı:**

```
CREATE TABLE Tabloadi (
    Sütunadi,
    Sütunadi CONSTRAINT Constraint_Adi DEFAULT (değer)
)
```

**Örnek:**

```
CREATE TABLE tblDevamsizlik(
    devamID int not null,
    ogrenciId int not null,
    devamsizlik int CONSTRAINT DC_devamsizlik DEFAULT(0)
)
```

➤ **Oluşturulmuş olan tabloda kullanımı:**

```
ALTER TABLE Tabloadi
ADD CONSTRAINT Constraint_Adi DEFAULT ifade veya değer FOR
Sütunadi
```

**Örnek:**

```
ALTER TABLE tblDevamsizlik
ADD CONSTRAINT cons_devamsizlik DEFAULT 0 FOR devamsizlik
```

## 2.2.5. Kontrol Kısıtlayıcı (Check Constraint)

Kontrol kısıtlayıcı anlamındadır. Belirtilen formata göre verilerin girilmesini sağlar. Örneğin, TcKimlikNo alanına 11 karakterin girilmesi Check Constraint ile sağlanabilir.

Check Constraint'in kullanım amaçları şöyledir:

- Sütuna girilebilecek verileri bir sınır ile kısıtlamak,
  - Sütuna girilebilecek verilerin belli bir formatta girilmesini sağlamak,
  - Sütuna girilebilecek verileri başka sütun formatlarına göre karşılaştırarak denetlemektir.
- **Yeni bir tablo oluşturulurken kullanımı:**

```
CREATE TABLE Tablo_Adi (  
    Sütun_Adi,  
    Sütun_Adi,  
    .....  
    CONSTRAINT Constraint_Adi CHECK (ifade)  
)
```

**Örnek:**

```
CREATE TABLE tblNotlar(  
    notID int not null,  
    dersId int not null,  
    not1 int,  
    CONSTRAINT CC_not1 CHECK (not1 > 0 AND not1 < 100)  
)
```

- **Oluşturulmuş olan tabloda kullanımı:**

```
ALTER TABLE Tabloadi  
    ADD CONSTRAINT Constraint_Adi CHECK (ifade)
```

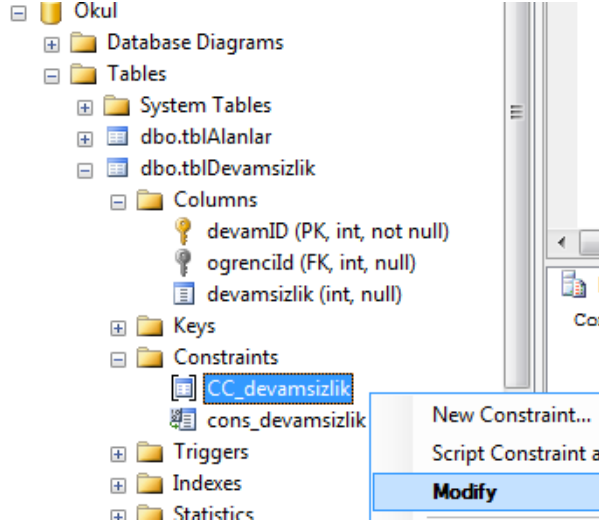
**Örnek:**

```
ALTER TABLE tblDevamsizlik  
    ADD CONSTRAINT CC_devamsizlik CHECK (devamsizlik >= 0)
```

## 2.3. Kısıtlayıcıları Düzenlemek

Kısıtlayıcıları düzenlemenin en uygun yolu Managemet Studio'yu kullanmaktır. Düzenleme işlemi T-SQL kodlarıyla da gerçekleştirilebilir ancak, bu işlem için birkaç aşamayı gerçekleştirmeniz gerekir.

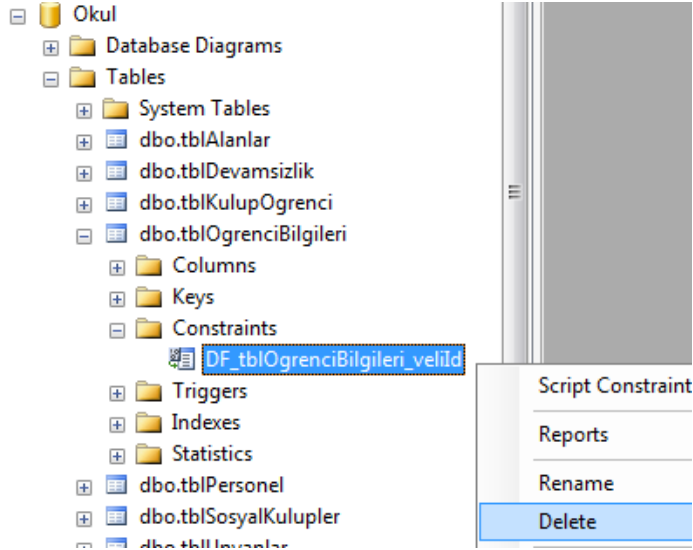
Management Studio’da oluşturulan kısıtlayıcıları düzenlemek için üzerinde fareyle sağ tıklayınız. Açılan yardımcı menü ile yeni bir kısıtlayıcı oluşturabilir, düzenleyebilir, adını değiştirebilir veya silebilirsiniz.



Resim 2.2: Object Explorer kısıtlayıcı düzenleme

## 2.4. Kısıtlayıcıları Silmek

Tanımlanan bir Constraint’i Management Studio’da mevcut kısıtlayıcı üzerinde fareyle sağ tıklayıp açılan menüden Delete komutunu vererek silebilirsiniz.



Resim 2.3: Object Explorer kısıtlayıcı silme

Bu işlemi T-SQL kodu yazarak da yapabilirsiniz.

```
ALTER TABLE tablo_adi  
DROP CONSTRAINT const_adi
```

## Örnek

```
ALTER TABLE tblDevamsizlik  
DROP CONSTRAINT CC_devamsizlik
```

## 2.5. Varsayılan (Default) Nesnesi Oluşturmak

Default nesnesi, Default Constraint ile aynı işleve sahiptir ve ayrı bir nesne olarak tanımlanır. Bir tabloda bir alan için sadece bir adet Default nesnesi tanımlanabilir. Ama, nesne olarak tanımlanmayan Default veya Check Constraint'ler birden fazla tanımlanabilir.

Genel kullanımı şu şekildedir.

```
CREATE DEFAULT Default_Adi AS değer veya ifade
```

Son olarak, oluşturulan Default nesnesinin `sp_bindefault` isimli sistem Stored Procedure'ü kullanılarak sütunla ilişkilendirilmesi gerekir.

```
sp_bindefault Default_Adi, 'tablo.sütun_adi'
```

Eğer, default nesnesini artık kullanmayacaksanız DROP ile silmeniz gerekir.

```
DROP DEFAULT Default_Adi
```

## 2.6. Rol (Rule) Oluşturmak

Rule nesnesi de ayrı bir nesne olarak tanımlanmaktadır. Check Constraint'lerle aynı işleri yapabilir. Rule oluşturduktan sonra `sp_bindrule` isimli sistem Stored Procedure'üyle ilişkilendirilmesi gerekir.

Genel kullanımı şu şekildedir.

```
CREATE RULE Rule_Adi AS ifadeler
```

Bağlantı şekli de aşağıdaki gibidir.

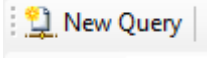
```
sp_bindrule Rule_Adi, Tablo.Sütun_adi
```


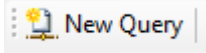
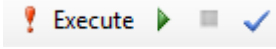
Rule'i silmek için de DROP komutunu kullanmalısınız.

```
DROP RULE Rule_Adi
```

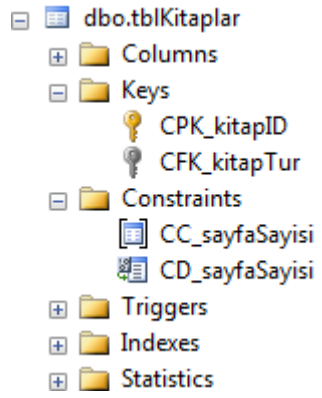
## UYGULAMA FAALİYETİ

Kütüphanede bulunan kitapların bilgilerini tutacak bir tablo oluşturup üzerinde gerekli kısıtları uygulayınız.

İşlem Basamakları	Açıklama
➤ Çalışacağımız veri tabanı üzerinde yeni bir sorgu penceresi açınız.	
➤ Tablo oluşturma komutumuzu yazalım ve tablomuz için bir isim belirleyiniz.	<code>CREATE TABLE tblKitaplar(</code>
➤ Şimdi kitap bilgilerini tutacak sütunlarımızı belirleyiniz.	
➤ Öncelikle kitapların birbiri ile karışmaması için tekil değer tutacak bir alan ve veri türünü belirleyiniz.	<code>kitapID int not null,</code>
➤ Kitap adını tutacak alanımızı ve veri türünü belirleyiniz.	<code>kitapAdi nvarchar(100) not null,</code>
➤ Kitaplarımızı bir türe göre gruplandırmak için bir alan oluşturunuz. Bu alana girilecek değer kitap türlerini listelediğimiz bir tablodan alınacağını düşününüz.	<code>kitapTur int not null,</code>
➤ Kitapların sayfa sayısını tutacak alanımızı ve veri türünü belirleyiniz. Bu alana bir değer girilmediği takdirde varsayılan olarak sıfır atanmasını sağlayınız.	<code>sayfaSayisi int null CONSTRAINT CD_sayfaSayisi DEFAULT(0),</code>
➤ Kitap özetini tutacak bir alan ve veri türünü belirleyiniz.	<code>kitapOzeti nvarchar(4000),</code>
➤ Tekil değer tutacak alanımızı birincil alan olacak şekilde kısıtımızı oluşturunuz.	<code>CONSTRAINT CPK_kitapID PRIMARY KEY (kitapID),</code>
➤ Sayfa sayısını tutan alanımıza sıfırdan küçük değer girilmesini engelleyecek kısıtımızı oluşturunuz.	<code>CONSTRAINT CC_sayfaSayisi CHECK(sayfaSayisi &gt;= 0),</code>
➤ Kitap türü değerini tutacak alanımıza kitapTur tablosundaki turId alanındaki değer haricinde değer girmesini engelleyecek kısıtımızı oluşturunuz.	<code>CONSTRAINT CFK_kitapTur FOREIGN KEY(kitapTur) REFERENCES tblKitapTur(turID)</code>
➤ Sorgumuzun son hali aşağıdaki gibi olacaktır:	

<pre>CREATE TABLE tblKitaplar(     kitapID int not null,     kitapAdi nvarchar(100) not null,     kitapTur int not null,     sayfaSayisi int null CONSTRAINT CD_sayfaSayisi DEFAULT(0),     kitapOzeti nvarchar(4000),     CONSTRAINT CPK_kitapID PRIMARY KEY (kitapID),     CONSTRAINT CC_sayfaSayisi CHECK(sayfaSayisi &gt;= 0),     CONSTRAINT CFK_kitapTur FOREIGN KEY(kitapTur)     REFERENCES <u>tblKitapTur</u>(turID) )</pre>	
➤ Execute butonumuza basarak sorgumuzu çalıştırınız.	
➤ tblKitapTur tablomuz olmadığı için sorgumuz hata verecektir.	
➤ Yeni bir sorgu penceresi açınız.	
➤ Tablo oluşturma komutumuzu yazalım ve tablomuz için bir isim belirleyiniz.	<pre>CREATE TABLE tblKitapTur(</pre>
➤ Kitap türlerinin birbiri ile karışmaması için tekil değer tutacak bir alan ve veri türünü belirleyiniz.	<pre>turID int not null,</pre>
➤ Kitap türlerini tutacak alanımızı ve veri türünü belirleyiniz.	<pre>tur nvarchar(50),</pre>
➤ Tekil değer tutacak alanımızı birincil alan olacak şekilde kısıtımızı oluşturunuz.	<pre>CONSTRAINT CPK_turID PRIMARY KEY(turID)</pre>
➤ Sorgumuzun son hali aşağıdaki gibi olacaktır:	
<pre>CREATE TABLE tblKitapTur(     turID int not null,     tur nvarchar(50),     CONSTRAINT CPK_turID PRIMARY KEY(turID) )</pre>	
➤ Sorgumuzu çalıştırınız.	
➤ Diğer sorgu penceremize geçip sorgumuzu çalıştırınız.	

➤ Sorgumuzun sonucunu inceleyiniz.



## KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız beceriler için **Evet**, kazanamadığınız beceriler için **Hayır** kutucuğuna (X) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. Yeni bir sorgu penceresi açabildiniz mi?		
2. Tablo oluşturma komutunu yazabildiniz mi?		
3. Gerekli olan alan isimlerini belirleyebildiniz mi?		
4. Alan türlerini belirleyebildiniz mi?		
5. Varsayılan değer kısıtlayıcısını oluşturabildiniz mi?		
6. Birincil anahtar kısıtlayıcısını oluşturabildiniz mi?		
7. Şart kısıtlayıcısını oluşturabildiniz mi?		
8. Yabancı anahtar kısıtlayıcısını oluşturabildiniz mi?		
9. Sorguyu hatasız bir şekilde oluşturup çalıştırabildiniz mi?		

## DEĞERLENDİRME

Değerlendirme sonunda “**Hayır**” şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetini tekrar ediniz. Bütün cevaplarınız “**Evet**” ise “Ölçme ve Değerlendirme”ye geçiniz.



## ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. Veri üzerindeki mantıksal sınırlamalara ..... adı verilir.
2. "....." saklı yordamını kullanarak istenilen nesnenin kısıtlama bilgileri elde edilebilir.
3. ....., belirtilen formata göre verilerin girilmesini sağlayan kısıtlayıcıdır.

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

4. Aşağıdakilerden hangisi kısıtlayıcı türlerinden değildir?  
A) Primary Key Constraint  
B) View Constraint  
C) Unique Constraint  
D) Default Constraint
5. Default nesnesinin sütunla ilişkilendirmek için hangi store procedure kullanılır?  
A) sp\_bindefault  
B) sp\_insertdefault  
C) sp\_addefault  
D) sp\_storedefault

## DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise “Modül Değerlendirme”ye geçiniz.

# MODÜL DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Yeni bir görünüm oluşturmak için hangi SQL komutu kullanılır?  
A) ALTER VIEW  
B) DROP VIEW  
C) CREATE VIEW  
D) INSERT VIEW
2. Sorgu penceresine yazdığımız SQL kod bloğunu çalıştırmak için hangi komut butonu kullanılmalıdır?  
A) Execute  
B) Debug  
C) Parse  
D) Build
3. Daha önceden tanımlanmış bir tetikleyiciyi güncellemek için hangi SQL komutu kullanılır?  
A) EDIT TRIGGER  
B) ALTER TRIGGER  
C) UPDATE TRIGGER  
D) DROP TRIGGER
4. Daha önceden tanımlanmış bir store procedure'yi çağırmak için hangi SQL komutu kullanılır?  
A) Run  
B) Debug  
C) Build  
D) Exec
5. Bir tablodaki bir sütuna ait verilerin başka bir tablonun belirli bir sütunundan gelmesini denetleyen kısıtlayıcı türü aşağıdakilerden hangisidir?  
A) Unique Constraint  
B) Default Constraint  
C) Foreign Key Constraint  
D) Check Constraint
6. Aşağıdakilerden hangisi store procedure'ün özelliklerinden değildir?  
A) Parametre alabilirler.  
B) Başka prosedürleri çağırabilirler.  
C) Önceden derlendiklerinden performansları yüksektir.  
D) Hata nedenini bir durum değeri olarak döndürebilirler.

7. Bir kayıt silindiğinde ağ veritabanı yazılımı tarafından otomatik oluşturulan tablo aşağıdakilerden hangisidir?  
A) deleted  
B) inserted  
C) delete  
D) edited

**Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.**

8. ...., bir tabloda güncelleme işlemi gerçekleştirildiğinde diğer tablonun etkilenmesi istendiği durumlarda kullanılan tetikleyici türüdür.
9. Birincil anahtar olan ve tablodaki diğer alanlar içinde aynı içeriğe sahip verilerin olmaması için ..... tanımlanır.
10. Rule oluşturduktan sonra ..... isimli sistem saklı yordamıyla ilişkilendirilmesi gerekir.

## **DEĞERLENDİRME**

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

# CEVAP ANAHTARLARI

## ÖĞRENME FAALİYETİ 1'İN CEVAP ANAHTARI

1	D
2	A
3	C
4	B
5	B

## ÖĞRENME FAALİYETİ 2'NİN CEVAP ANAHTARI

1	Kısıt
2	sp_helpconstraint
3	Kontrol Kısıtlayıcı / Check Constraint
4	B
5	A

## MODÜL DEĞERLENDİRMENİN CEVAP ANAHTARI

1	C
2	A
3	B
4	D
5	C
6	B
7	A
8	Güncelleme Tetikleyicisi / Update Trigger
9	Tekil Alan Kısıtlayıcı / Unique Constraint
10	Sp_bindrule

## KAYNAKÇA

- <http://www.webmastersitesi.com/mssql/116702-veri-butunlugu.htm>  
(25.06.2012 – 09:25)