VERİ TABANI VE YÖNETİMİ DERSİ – UYGULAMA FÖYLERİ
Föy-6

Konu: Triggerlar ,Trigger oluşturmak, INSERT ,UPDATE,DELETE Triggerları , Trigger Yönetimi

GENEL BİLGİLER (Ayrıntılı Bilgi için Kitabın 22. Ünitesi incelenebilir.)

Trigger (Tetikleyici), en genel anlamda kayıt ekleme, silme ve güncelleme sonucunda otomatik olarak devreye giren özel bir tür Stored Procedure olarak kabul edilir.

Trigger'ları Kavramak

Klasik manada Trigger, veri değişiminin hemen ardından loglar üzerinden otomatik devreye giren özel bir tür Stored Procedure'dür. Sadece UPDATE,INSERT ve DELETE işlemlerinden sonra devreye girebilmek üzere programlanabilirler.

Trigger'lar Ne Zaman Kullanılmalıdır?

DML Trigger'lar (Klasik Trigger'lar)

- Değişiklikleri takip etmek için
- Birincil anahtar üretmek için
- Komplike iş kurallarını gerçekleştirmek için
- Mail atmak gibi işlevleri otomatikleştirmek için

DDL Trigger'lar

- Veritabanı erişimlerini takip etmek için
- Yeni nesne oluşturma/nesne değişikliklerini takip/engel için

Trigger Ne Zaman Kullanılmamalıdır?

- Aynı veritabanı içerisinde birincil anahtar-yabancıl anahtar uyumluluğunu test etmek için
- Bir sorgudan kaç kaydın etkilendiğini bulmak için

Klasik Trigger'ların Özellikleri Ve Kısıtlamaları

Bir trigger, birden fazla olayı gerçekleştirebilir. Örneğin bir tabloya kayıt eklendiğinde devreye giren bir trigger bu kayıtları başka yerlere de ekledikten sonra, başka bir tablo üstünde de güncellemeler yapabilir. Öte yandan bir trigger, birden fazla olay tarafından tetiklenebilir. Yani hem INSERT hem de UPDATE işlemi için veya her üçü ya da herhangi biri için devreye girmek üzere programlanabilir. Trigger, geçici tablo ya da sistem tablosu oluşturamaz ama bu türden tablolara erişebilir.

Temel olarak iki farklı trigger çeşidi vardır: AFTER ve INSTEAD OF.

AFTER trigger'i sadece tablolar üstünde tanımlanabilir. INSTEAD OF trigger'ı ise hem tablolar hem viewler üstünde tanımlanabilir.

İşlev	INSTEAD OF	AFTER
Tanımlama	View'ler veya tablolar için	Sadece tablolar için
Adet Kısıtlaması	Her view'de işlem başına sadece bir adet bulunabilir. Birden fazlası için yeni bir view tanımlayıp onun üstünde birer adet daha tanımlanabilir.	Her tabloda birden fazla tanımlanabilir.
Çalışma Sırası	-(Her olay için tek trigger)	İlk ve son devreye girmesi istenen trigger'lar sp_settring-gerorder sistem stored procedure'ü ile ayarlanabilirler. Bunun dışındakiler karışık sırada çalışırlar.

Trigger'lar Nasıl Çalışır?

Triggerlar çalıştığı zaman Inserted ve Deleted tablolarını kullanırlar. Gerçek tabloya bir kayıt eklendiği zaman, trigger çalışırsa bu kayıt aynı zamanda Inserted tablosuna da eklenir. Tablodan bir kayıt silindiğinde silinen kayıt deleted tablosunda saklanır. Update işlemi ise delete ve hemen ardından yapılmış bir insert işlemi olarak ele alınır. Bir kayıt update edildiğinde orijinal kayıt deleted tablosuna işaretlenir, değişen kayıt da inserted tablosunda (ve gerçek tabloda da) saklanır.

Trigger Oluşturmak

CREATE TRIGGER trigger_adi

ON tablo_adi

[WITH seçenekler]

{ FOR | AFTER |INSTEAD OF } { INSERT | UPDATE | DELETE [,UPDATE , [,DELETE]] }

AS

Otomatik çalışacak SQL İfadeleri

SQL Server'da iki farklı türden trigger tanımlanabilir: Aynı işleve sahip FOR veya AFTER ile ya da INSTEAD OF ile

Üç farklı olay için trigger yazılabilir: INSERT, UPDATE, DELETE

- *Bir tablo üstünde hangi triggerlar olduğunu öğrenmek için ,*

sp_helptrigger tablo_ismi

- *Bir triggerın hangi tabloların hangi alanlarına etki ettiğini görmek için ,sp_depends sistem procedure'ünü kullanabiliriz*

sp_depend trigger_ismi

Örnek: Yeni bir sipariş verildiği zaman, a@b.com adresine mail atacak bir trigger programlayalım:

CREATE TRIGGER siparisGeldi

ON tblSiparis

```
FOR INSERT
AS
BEGIN
DECLARE @FaturaKod BIGINT, @msg VARCHAR(255)
SELECT @FaturaKod=inserted.faturaKod
FROM inserted
declare @body varchar(500) = 'Yeni siparis alindi: Fatura Kodu: ' + CAST(@faturakod AS
VARCHAR(5))
EXEC msdb.dbo.sp_send_dbmail
            @profile_name = 'Deneme'
           ,@recipients = 'oozyurt61@gmail.com'
           ,@subject = 'Yeni kayıt alındı'
           ,@body = @body
           ,@importance ='HIGH'
END
```

Yukarıdaki trigger'i yazın, ilgili dosyaya kayıt girin ve sonucu gözlemleyin. !

https://www.cozumpark.com/sql-server-database-mail-ve-gmail-ile-kullanimi/

https://www.mssqltips.com/sqlservertip/2578/setup-sql-server-database-mail-to-use-a-gmail-hotmail-or-outlook-account/

NOT: Sunucu mail ayarlarınızı kontrol ediniz…

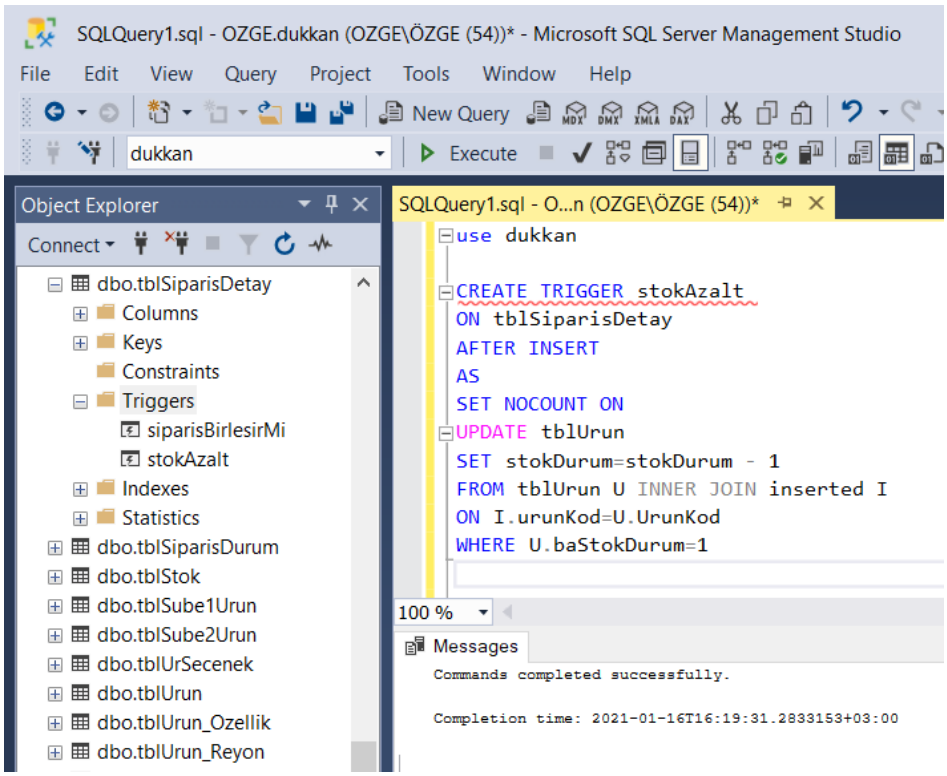
INSERT TRIGGER'I

➔ Bir tablo üstünde yeni kayıtlar eklenince devreye girer.

Insert trigger'ı devreye girdikten sonra Inserted tablosunda, yeni eklenen kayıtların bir kopyası tutulmaya başlanır. Bu tablo ,kayıt eklenen tablonun yapısal bir kopyasıdır ve trigger sona erinceye kadar saklanır.

Örnek: Sipariş verildiğinde, sipariş edilen ürünün stok takibi sistem tarafından yapılıyorsa stok miktarını bir azaltacak bir trigger yazalım. Burada baStokDurum=1 olan ürünlerin stok takiplerini yapmak, mağaza yazılımına bırakılmış demek ve stokDurum da ürünün stokta kaç adet bulunduğunu gösteriyor.

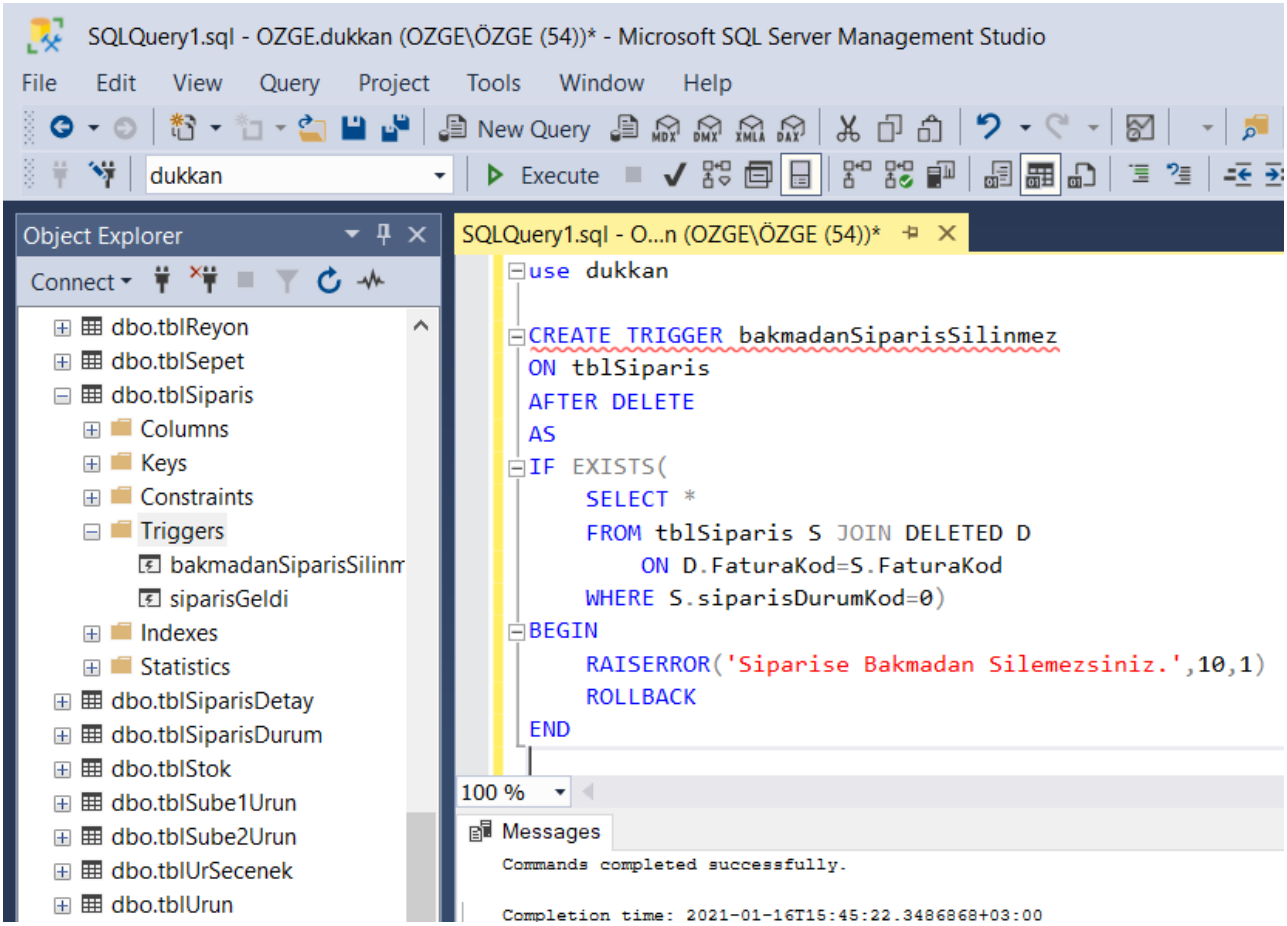
```
CREATE TRIGGER stokazalt
ON tblSiparisDetay
AFTER INSERT
AS
SET NOCOUNT ON
    UPDATE tblurun
     SET stokdurum = stokdurum-1
     FROM tblurun U INNER JOIN inserted I
      ON I.urunkod = U.urunkod
      WHERE U.bastokdurum = 1
```



DELETE TRIGGER'I

Deleted tablosunun Inserted'den bir farkı vardır, silinmiş kayıtlar sadece Deleted'de bulunur. Gerçek tablodan artık silinmiştir. Ancak transaction henüz kapatılmadığı için istenmeyen bir durumu trigger'ın ROLLBACK yapması sağlanabilir.

Örnek: Sipariş tablosunda bir silme işlemi gerçekleştirildiğinde, silinen siparişler içerisinde, henüz kontrol edilmemiş siparişler (siparisDurumKod=0) varsa,silme işlemini iptal edecek bir trigger yazalım:

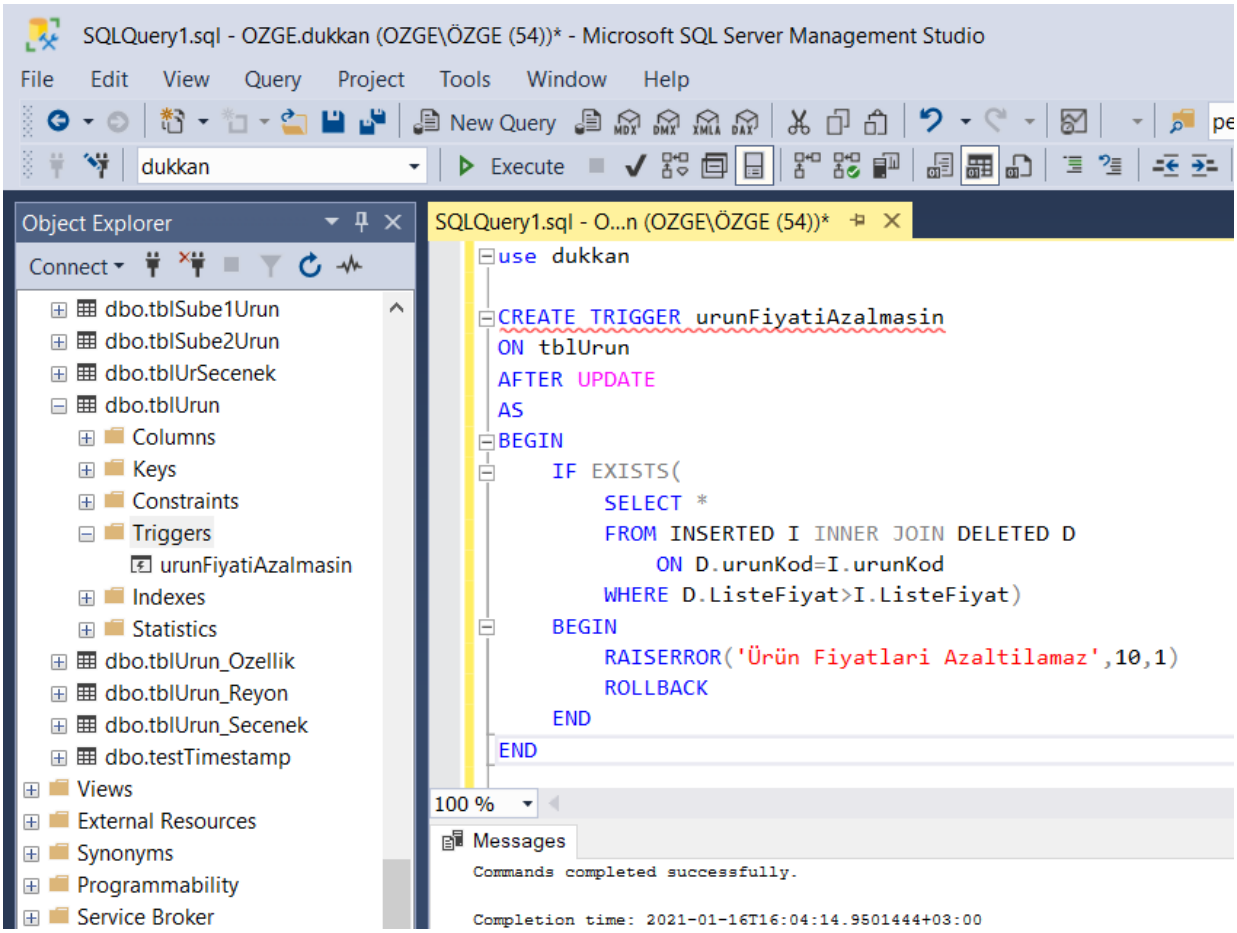


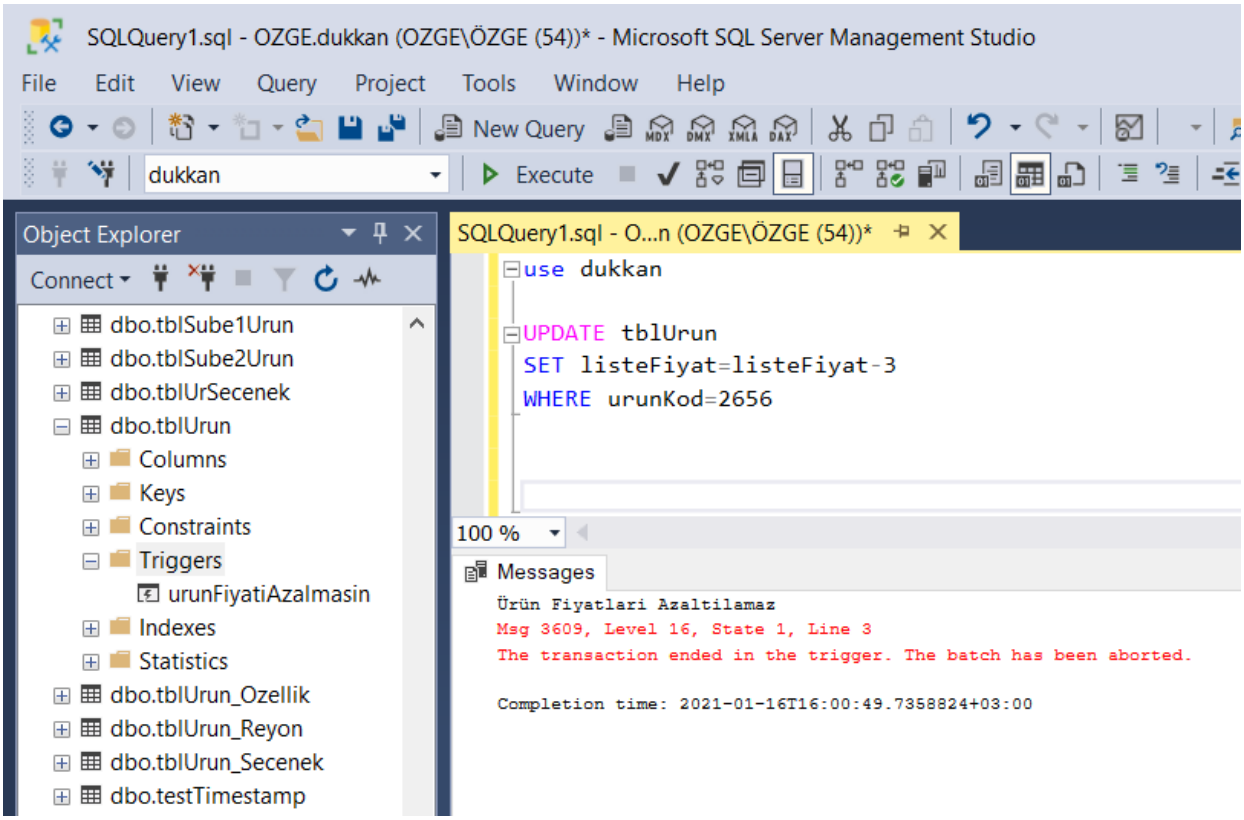
UPDATE TRIGGER'I

UPDATE trigger'ı devreye girdiği zaman iki sözde tablo oluşur (Inserted ve Deleted).Bu tablolardan Inserted, ana tablodaki kayıtlardan düzeltilenlerin yeni hallerinin kopyasını, Deleted ise aynı kayıtların güncellenmeden önceki hallerini tutar.

Örnek: Ürünlerin fiyatlarının sadece arttırılabileceği ama asla azaltılamayacağı bir trigger yazalım.

```
CREATE TRIGGER urunFiyatiAzalmasin
ON tblUrun
AFTER UPDATE
AS
BEGIN
	IF EXISTS(
		SELECT *
		FROM INSERTED I INNER JOIN DELETED D
			ON D.urunKod=I.urunKod
		WHERE D.ListeFiyat>I.ListeFiyat)
	BEGIN
		RAISERROR('Ürün Fiyatlari Azaltilamaz',10,1)
		ROLLBACK
	END
END
```





INSTEAD Of Trigger'ı

Buraya kadar yaptığımız bütün örneklerde dikkatinizi çekmiş olabilir, trigger'lar hep veri uygun değil ise işlemleri geri aldı. Oysa değişim başlamadan hemen önce verilerin uygunluğunun denetleyecek bir trigger türü çok daha kullanışlı olabilir diye düşündünüzse, işte bu tür, INSTEAD OF tip triggerlardır.

Instead of triggerlar AFTER triggerların aksine, ana tabloda bir değişiklik yapılmadan hatta constraint'ler bile devreye girmeden önce ama sözde tablolara ilgili işlem (INSERT, UPDATE VEYA DELETE) yansıtıldıktan sonra devreye girerler.

Bu diğer önemli nokta, INSTEAD OF triggerları her bir tablo ya da view üstünde her bir olay için (INSERT,UPDATE VEYA DELETE) sadece bir adet tanımlanabilir.

İpucu : INSTEAD OF triggerlar tabloların yanı sıra viewler üstünde de tanımlanabilirler.

```
CREATE TRIGGER adi
ON tablo_adi
[WITH seçenekler]
INSTEAD OF {INSTERT| UPDATE | DELETE}
AS
        OTOMATİK ÇALIŞACAK SQL İFADELERİ
```

INSTEAD OF triggerlar genellikle birden fazla tablodan veri gösterip view'ler üstünden UPDATE,INSERT,VE DELETE işlemlerini gerçeklemek için tanımlanırlar

SELECT OBJECTPROPERTY (OBJECT_ID('trigger_ismi'),'ExecIsInsteadOfTrigger')

İpucu : Bir trigger'ın INSTEAD OF trigger olup olmadığını anlamak için , OBJECTPROPERTY fonksiyonu aşağıdaki şekilde kullanılır.

DDL Trigger'lar

Bir klasik trigger, tablo uzayında tanımlanır. Yani bir trigger, sadece üstünde tanımlandığı tablodaki veri değişimine izin duyarlıdır .DDL triggerlar bundan farklı olarak iki farklı uzayda tanımlanabilirler ; veritabanı seviyesinde veya sunucu seviyesinde.

DDL Triggerlar veri ile ilgili işlemlere bakmadıklarından, INSERTED VE DELETED sözde tablolarına erişimleri söz konusu olamaz . Her türden trigger içerisinden eventdata() fonskiyonu çağrılarak; triggeri tetikleyen olay ve gerçekleşme ani gibi çok ayrıntılı bilgiler içeren XML veriye erişmek mümkündür. Bütün DDL Triggerlar için eventdata() fonksiyonu şu verileri döndürür:

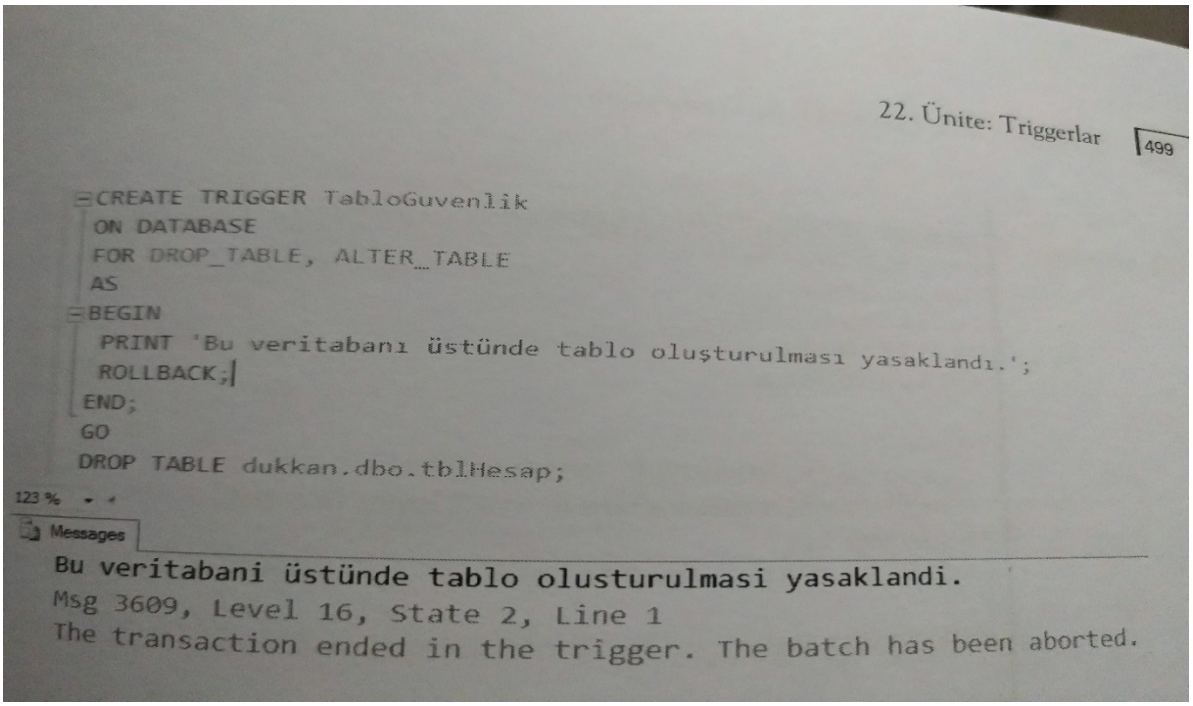
- Olay Ne zaman gerçekleşti
- Triggeri tetikleyen bağlantıya ait tekil SPID tekil sistem sunucu proses kodu
- İfadeyi çalıştıran kullanıcı adı ve ifadenin hangi kullanıcı adına çalıştırıldığı
- Olayın tipi
- Triggerları tetikleyen Sql kodu

DDL Trigger oluşturmak için genel ifade :

```
CREATE TRİGGER trigger_ismi
ON {DATABASE | ALL SERVER}
FOR { veritabanı-seviye-olaylar | server-seviye-olaylar }
AS
--- trigger programsal mantığı
```

Veritabanı Seviye DDL Triggerla Çalışmak

İfade hangi veritabanı üstünde çalıştırılırsa, trigger o veritabanı seviyeli olarak oluşturulur.



Yukarıdaki kodları yazın ve sonucu gözlemleyin.

Üstünde çalıştığınız veri tabanındaki, veritabanı seviye triggerların genel bir listesini görmek için

SELECT * FROM sys.triggers

WHERE parent_id=0

Sunucu Seviyeli DDL Triggerlar

Sunucu seviyeli DDL triggerlar, veritabanı seviyeli triggerlardan farklı olarak sunucunun bütününe özgü işlemlere karşı tepkime verir. Veritabanı oluşturulması, yeni bir rol eklenmesi vb işlemler bu gruptandır.

Sunucu seviyeli DDL Triggerlar oluşturmak için trigger sahası olarak ALL SERVER vermek gerekir

Örnek:
Bütün veritabanındaki login oluşturma , silme ve değiştirme olaylarına karşı tepkime verecek bir trigger oluşturalım :

```
CREATE TRIGGER SunucuDegisikligiKaydedici
ON ALL SERVER
FOR DDL_LOGİN_EVENTS
AS
PRINT 'Login olayı yakalandı.'
SELECT EVENTDATA().value('(/EVENT_INSTANCE/TSQLCommand/CommandText)[1] ',' nvarchar(max)')
GO
```

Arkasından triggerı tetikleyecek bir olay olarak şu ifadeyi çalıştıralım:

```
CREATE LOGIN YakalaBeni WITH PASSWORD='123'
```

Logon Triggerlar

SQL Serverda sunucuya giriş izni olan kullanıcı tanımlarına login denir. Bir login SQL Servera giriş yapmak istediğinde yetkilendirme işleminden hemen sonra, bağlantı açılma işleminden hemen önce LOGON olayı tetiklenir. LOGON olayı tetiklendikten sonra bir Logon Triggerı varsa devreye girer.

ÖRNEK :
WebRapor adındaki bir login hesabının sadece hafta içinde giriş yapabilmesi için bir Logon Trigger kodlayalım
(Öncelikle WebRapor kullanıcısını oluşturmuş olmak gerekir)

```
USE master;
GO
CREATE LOGIN WebRapor WITH PASSWORD =”ezbırcıme”;

USE dukkan;
GO
CREATE USER WebRapor FOR LOGIN WebRapor;
ALTER ROLE [db owner] ADD MEMBER [WebRapor];

--Arkasından Login triggerımızı kodlayalım
CREATE TRIGGER [TrgSadeceIsGunleri]
ON ALL SERVER
FOR LOGON
AS

 BEGIN
DECLARE @ErrorText[varchar](128)
SET @ErrorText =’ “ WebRapor” kullanıcısı sadece haftaıcı bağlanabilir ‘
SET @ErrorText  =@ErrorText + ‘lutfen hafta ıcı gunlerınde deneyiniz ‘

IF ORIGINAL LOGIN() = ‘WebRapor’ AND (
DATEPART (WEEKDAY, GETDATE()) < 2 OR (DATEPART ( WEEKDAY, GETDATE()) > 6 )
)
        BEGIN
                PRINT @ErrorText
                ROOLBACK;
        END
END;
```

TRIGGER YONETİMİ

Triggerların etkilendiği olayları değiştirebiliriz. Örneğin bir trigger öncesinde UPDATE işlemine tepki veriyorken, biz bunu hem UPDATE hem INSERT işlemine tepki verecek hale getirebiliriz.

Bir Triggerın sadece adını değiştirmek için sp_rename sistem stored procedure' ünü kullanmak yeterlidir. Şu şekilde kullanılır,

sp_rename @objname=eski_isim , @newname= yeni_isim

Parametre	**İşlevi**
@objname	Nesnenin halihazırda geçerli olan ismi
@newname	Nesnenin değiştirilme işleminden sonraki ismi

ALTER Trigger ifadesi genel olarak şu şekildedir:

```
ALTER TRIGGER trigger_adi
ON { tablo_adi | DATABASE | ALL SERVER }
[ WITH secenkeler ]
{FOR olay_tanimlari}
AS
Otomatik çalışacak SQL ifadeleri
```

Triggerların Silinmesi

DROP TRIGGER trigger_ismi

Veritabanı Seviyeli DDL Triggerları silmek

DROP TRİGGER trigger_ismi ON DATABASE

Sunucu Seviyeli DDL Triggerları Silmek

DROP TRIGGER trigger_ismi

ON DATABASE

Triggerları Tepkimeye Kapatmak

```
ALTER TABLE tablo_ismi
DISABLE TRIGGER trigger_ismi | ALL

tablo_ismi paramatresinin yerine, INSTEAD OF triggerları için view ismide gelebilir
```

Sadece o Triggerın aktif hale geçmesini sağlamak için:

```
ALTER TABLE tabl_ismi
ENABLE TRIGGER trigger_ismi | ALL
```

BÖLÜM SONU ETKİNLİĞİ

1- Ürünlerde %5 indirim yapmayı deneyiniz. Sonucu gözlemleyiniz, trigger'ları kontrol edip bu işlemi gerçekleştiriniz.
2- Dükkan veri tabanında yeni bir tablo oluşturulmasını engelleyecek triggeri kodlayınız.