VERİ TABANI VE YÖNETİMİ DERSİ – UYGULAMA FÖYLERİ

Föy-3

Konu: Veri Bütünlüğü, Tanımlamalı ve Programsal Veri Bütünlüğü, Veri Bütünlüğü Türleri, SQL Server ile gerçekleme

GENEL BİLGİLER (Ayrıntılı Bilgi için Kitabın 7. Ünitesi incelenebilir.)

Bir tabloya veri eklenirken, tablodan veri silinirken ya da tablodaki veriler güncellenirken, ayın varlığa ait diğer tablolardaki verilerin de birbirleri ile uyumluluğunun devam ettirilmesi gerekmektedir. Bu tür durumlarda veri tutarlılığının garanti altına alınması için VTYS tarafından sağlanan veri bütünlüğü öğelerini tanımlamak ya da programlamak gerekir.

Veri Bütünlüğünü Sağlamak

VTYS'lerde veri bütünlüğü iki şekilde sağlanabilir

→ Tanımlamalı Veri Bütünlüğü: Tanımlanan nesnelerin kendi özellikleri ile…

→ Programsal (Prosedürel) Veri Bütünlüğü: Bir programlama yaklaşımı ile…

Kaynak tüketimi, hız ve esneklik ve performans açısından bu iki türü kıyaslayınız.

Tanımlamalı Veri Bütünlüğü

Tanımlamalı veri bütünlüğü öğeleri, Constraint'ler, Rule'lar ve Default'lardır.

Constraint (Kısıtlayıcı) türleri:

1- Primary key constraint
2- Unique Constraint
3- Check Constraint
4- Default Constraint
5- Foreing Key Constraint

Tanımlamalı veri bütünlüğü üç temel esasa dayanır: *Varlık bütünlüğü, sütun bütünlüğü, referanssal bütünlük*

Varlık bütünlüğü türleri: *Primary key constraint, unique constraint*

Sütun bütünlüğü türleri: *Check constraint, default constraint, primary ve foreign key constraint çifti*

Referans bütünlüğü türleri: *Check constraint, foreign key constraint*

Programsal (Procesürel) Veri Bütünlüğü

Trigger ve stored procedure'ler ile VTYS dışında program yazarak sağlanan veri bütünlüğüne verilen addır.

Trigger ve stored procedure kavramlarını araştırınız.

Veri Bütünlüğü Tekniğine Karar Vermek

Veri bütünlüğü tekniklerini seçerken birçok parametre göz önünde bulundurulabilir. Bunlardan en önemlisi, tekniğin transaction'dan önce mi sonra mı devreye girmesi ile ilgilidir. Constraint'ler sisteme yük bindirme açısından en başarılı olanıdır.

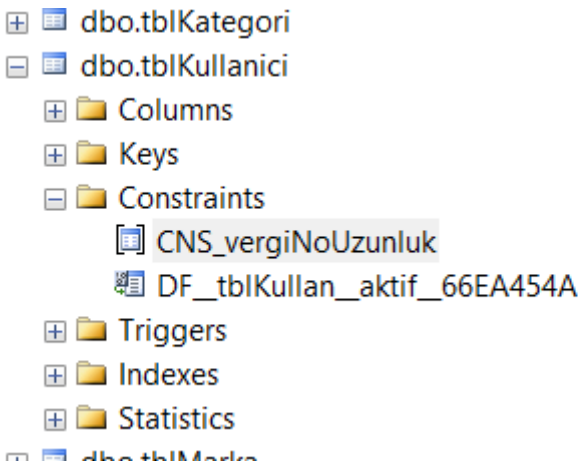
Constraintler, tablonun yapısında yer alırlar, transaction'dan önce devreye girerler. Ancak düşük işlevsellik sunarlar.

Programlamalı veri bütünlüğü araçları örn. Trigger işlevsellik açısından gelişmiştir. Olumsuz yanı transaction'dan sonra devreye girmesidir.

NOT: Bir tablo üzerinde daha sonra constraint tanımlanırken dikkat edilmesi gereken ana husus, tablodaki verilerin o constraint'i bozmaması gerektiği – ona uygun olması – dir.

UYGULAMA

Bir tablo üzerinde önceden tanımlanmış constraintler aşağıdaki şekildeki gibi görülebilir, ilgili sekmeye sağ tıklanarak istenilen constraint tanımlanabilir.



Veri Bütünlüğünü Gerçeklemek

Constraint'ler Default'lar ve Rule'ların nasıl programlandığını bu bölümde öğreneceğiz.

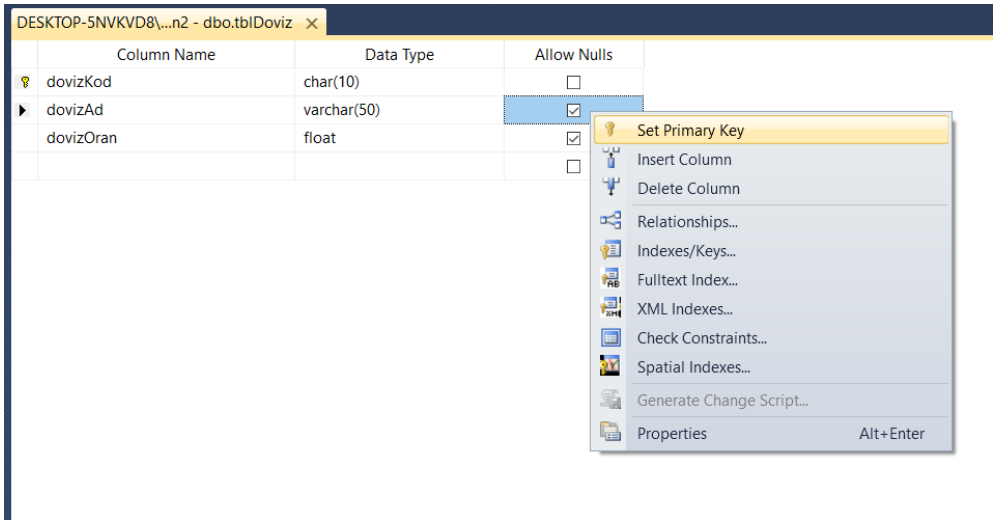
Constraint'ler tabloların tapısında yer alan tanımlamalardır. Bu nedenle constraint'ler tablolar oluşturulurken CREATE TABLE komutu ile birlikte tanımlanabilir. Tablolar daha önce tanımlanmışsa ALTER TABLE deyimi içerisinde constraint'ler tanımlanabilir. Ancak tabloya önceden girilmiş olan veriler var ise, tablo üstünde constraint'ler tanımlayabilmek için bu verilerin Constraint'leri ihlal etmiyor olması gerekir.

Primary Key Constraint Oluşturmak

Normalizasyon çerçevesinde tablolarınızı tasarladıysanız, her bir tablonun en az bir aday anahtar alan ile tanımlanmış olması gerekir. Birincil anahtar, aday anahtardan birinin seçilmesi ile mantıksal tasarım aşamasında elde edilir. Primary Key Constraint türünden bir kısıtlayıcı ile veri tabanı yönetim sistemine, bu alanı Primary Key olarak denetlenmesi gerektiğini deklare etmiş oluruz.

Bir tabloda sadece bir tane Primary Key Constraint tanımlanabilir. Daha önceden tablo üstünde bir CLUSTERED Indeks alan tanımlanmış değil ise ve özel olarak NONCLUSTERED deyimi kullanılmazsa, Primary Key Constraint olarak tanımlanan sütun ya da sütunlar SQL Server tarafından Unique CLUSTERED Indeks olarak ele alınır.

Aşağıdaki gibi birincil anahtar kısıtlayıcı tanımlanabilir.



Var olan bir tablo üzerinde sonradan kodla birincil anahtar kısıtlayıcı oluşturulabilir.

ALTER TABLE tablo_ismi

ADD CONSTRAINT constraint_ismi PRIMARY KEY(sutun_ismi)

[CLUSTERED|NONCLUSTERED],

Örnek:

tblDoviz1 tablosunu, primary key tanımı ile oluşturalım:

SQLQuery1.sql - (l...NVKVD8\orcun (54))*

```
CREATE TABLE tblDoviz (
    dovizKod CHAR(5),
    dovizAd VARCHAR(50),
    dovizOran DECIMAL(18,4),
    CONSTRAINT PKC_dovizKod PRIMARY KEY(dovizKod)
)
```

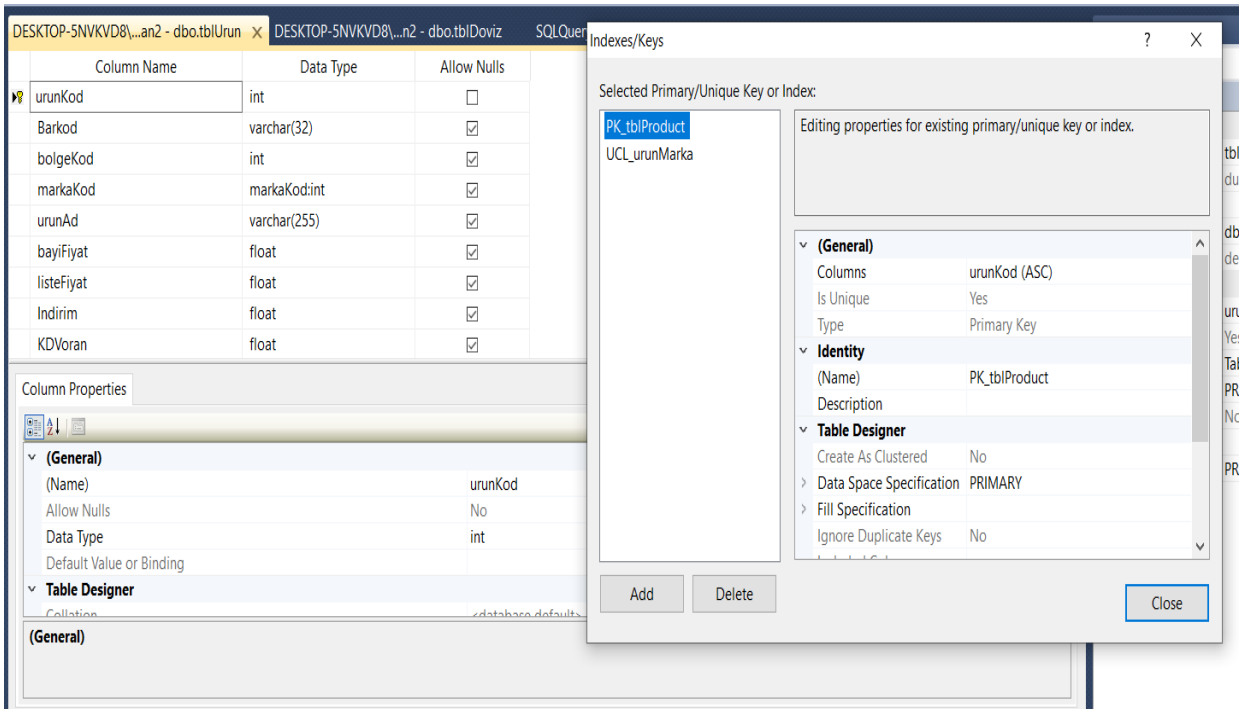
DESKTOP-5NVKVD8\...n2 - dbo.tblDoviz | SQLQuery1.sql - (l...NVKVD8\orcun (54))*

Column Name	Data Type	Allow Nulls
dovizKod	char(10)	☐
dovizAd	varchar(50)	☑
dovizOran	float	☑
		☐

NOT: Daha önce tablo üzerinde böyle bir tanımlama olabilir, onun için ilk önce onun silinmesi gerekir ya da farkı isimde bir tablo oluşturarak bunu test edebilirsiniz.

Unique Constraint Oluşturmak

Unique Key Constraint'i Primary Key Constraint'ten ayıran en önemli iki özellik, Unique Key, bir tabloda birden fazla olabilir. Ancak Primary Key tablo başında sadece bir adet alan olabilir. Unique Key Constraint tanımlı bir alan NULL değerler kabul edebilir. Ancak NULL kalmayacaksa mutlaka önceden girilmiş değerlerden farklı olmak zorundadır.



Bir tablonun oluşturulması esnasında, üstünde bazı sütunlar için Unique Constraint tanımlamada şu genel ifade kullanılabilir:

CREATE TABLE tablo_ismi (

Sütün tanımları…,

CONSTRAINT constraint_ismi UNIQUE [CLUSTERED|NONCLUSTERED]

(sutun ismi)

)

Daha önceden tanımlı bir tablo üstünde Unique Constraint oluşturmak için, genel ifade şu şekildedir:

ALTER TABLE tablo_ismi

ADD CONSTRAINT constraint_ismi UNIQEU [CLUSTERED|NONCLUSTERED]

(sutun ismi)

```
CREATE TABLE tblkullanici1(
kullanicikod VARCHAR(10),
kullaniciad VARCHAR(30),
email VARCHAR(30),
vergino VARCHAR(10),
aktifmi VARCHAR(1),
tel VARCHAR(1)
)
```

Yukarıdaki kodu çalıştırarak ilgili tabloyu oluşturun.

```
ALTER TABLE tblKullanici1
ADD CONSTRAINT PK_tblKullanici PRIMARY KEY CLUSTERED (kullaniciKod),
CONSTRAINT UCkullaniciAd_tblKullanici UNIQUE (kullaniciAd),
CONSTRAINT UCemail_tblKullanici UNIQUE (email),
CONSTRAINT UCvergiNo_tblKullanici UNIQUE (vergiNo)
```

Yukarıdaki kodu çalıştırın. Ardından bu tabloya veri girmeye çalışın, bazı alanlara aynı değerleri girmeyi deneyin (Management Studio üzerinden el ile girmeyi deneyebilirsiniz veya insert into ile deneyebilirsiniz). Sonucu gözlemleyin.

Insert into tablo adı (alanlar) values (deger1,değer2, …, degern)

Not: Bu tablo üzerinde daha önceden tanımlamalar varsa onları görsel araçları kullanarak silin.

Default Constraint Oluşturmak

Default Constraint'ler tabloya veri eklenirken bir alana veri girilmemesi halinde bu alanın alacağı standart bir değer tanımlamada kullanılır. Genellikle tarih alanı için değer girilmediğinde o günün tarihini kabul etmek üzere SQL Server'da GETDATE() fonksiyonu, kaydı gerçekleştiren geçerli kullanıcıyı kayda geçirmek için SUSER_NAME() fonksiyonu kullanılabilir.

Örnek:

TblKullanici tablosunda yer alan aktifMi sütununda bir Constraint tanımlayalım: Şayet bu sütunun değeri belirtilmedi ise, kullanıcı aktif olarak işaretlensin.

```
ALTER TABLE tblKullanici1
ADD CONSTRAINT DC_Aktif DEFAULT 1 FOR aktifMi
```

Yukarıdaki kodları çalıştırın. Sonucu gözlemleyin. İlgili tablolara constraint'e uygun ve uygun olmayan veriler girmeye çalışın. Sonucu gözlemleyin.

Check Constraint Oluşturmak

Bir sütuna girebilecek değerleri, belli kıyaslara karşı kontrol eden kısıtlayıcı türüdür. Farklı amaçlar için kullanılabilir. Bir sütün için, gerek olduğunda birden fazla Check Constraint tanımlanabilir.

Check Constraint'ler 3 farklı amaç için kullanılabilir:

1. Bir sütuna girebilecek değerleri belli bir küme ile kısıtlamak için
2. Bir sütuna girebilecek değerleri belli bir formata uygunluğunu denetlemek için
3. Bir sütuna girebilecek değerleri başka sütunlar üstünden bir kıyas ile denetlemek için

Bir tablo ilk oluşturulurken bir sütunu üstünde Check Constraint tanımlamak için şu genel yapı kullanılır:

```
CREATE TABLE tablo_ismi (
Sütün tanımlamaları…,
CONSTRAINT constraint_ismi CHECK(ifade) FOR sutun_ismi
)
```

Bir test tablosu oluşturalım. Bu tabloya girilecek isimlerin 4 karakterden daha uzun olmasını istiyorsak:

```
CREATE TABLE test
(
```

İsim VARCHAR(12),

CONSTRAINT CHC_LEN CHECK(LEN(isim)>4)

)

GO

INSERT INTO test VALUES('Ali)

Yukarıdaki kodu çalıştırın ve sonucu gözlemleyin.

Daha önceden oluşturulmuş bir tablonun sütunu üstünde Check Constraint tanımlamak için:

ALTER TABLE tablo_ismi

ADD CONSTRAINT constraint_ismi CHECK(ifade) FOR sutun_ismi

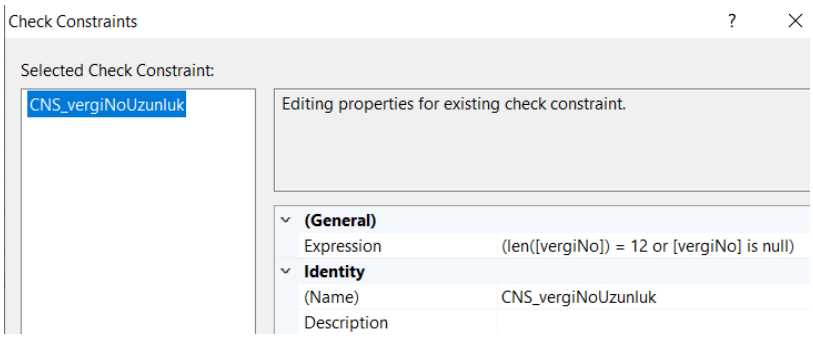
Örnek:

tblKullanici tablosunda, vergiNo sütununa 10 haneden daha az veri girilmemeli. Çünkü kişilerin vergi numaraları 10 hanelidir veya hiç girilmemiştir. Yani NULL'dur.

ALTER TABLE tblKullanici1

ADD CONSTRAINT CNS_vergiNoUzunluk CHECK(LEN(vergiNo)=10 OR vergiNo IS NULL)

Yukarıdaki kodu çalıştırın. Aşağıdaki gibi kısıtlayıcıyı gözlemleyin.



Örnek:

İçinde @ geçmiyorsa e-mail adresini kabul etmemek üzere bir Check Constraint ekleyelim.

```
ALTER TABLE tblKullanici1
ADD CONSTRAINT CHK_Mail CHECK(CHARINDEX('@',email)>0 OR email IS NULL)
```

Örnek:

Telefon numarasının şu formatta girildiğinden emin olmak istiyoruz: 0XXXYYYYYYY

Burada sadece ilk hanenin sıfır olmasını ve uzunluğun 11'den az olmamasını denetlemek üzere bir constraint tanımlayalım:

```
ALTER TABLE tblKullanici1
ADD CONSTRAINT CHK_Tel CHECK(
tel IS NULL OR
(tel LIKE "[0] [0-8] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]") AND LEN(tel) = 11)
```

Yukarıdaki kodları çalıştırdıktan sonra ilgili tablolara kayıt eklemeyi deneyerek sonucu gözlemleyiniz. Kısıtlayıcıyı bozan ve bozmayan kayıtlar ekleyerek sonucu gözlemleyin.

Tablo Seviyeli Veri Bütünlüğü Sağlamak

Veri tabanındaki verilerin tablo içerisinde ve tablolar arasında birbiri ile uyumlu olması sağlanabilir.

Sütunlar Arası Check Constraint Oluşturmak

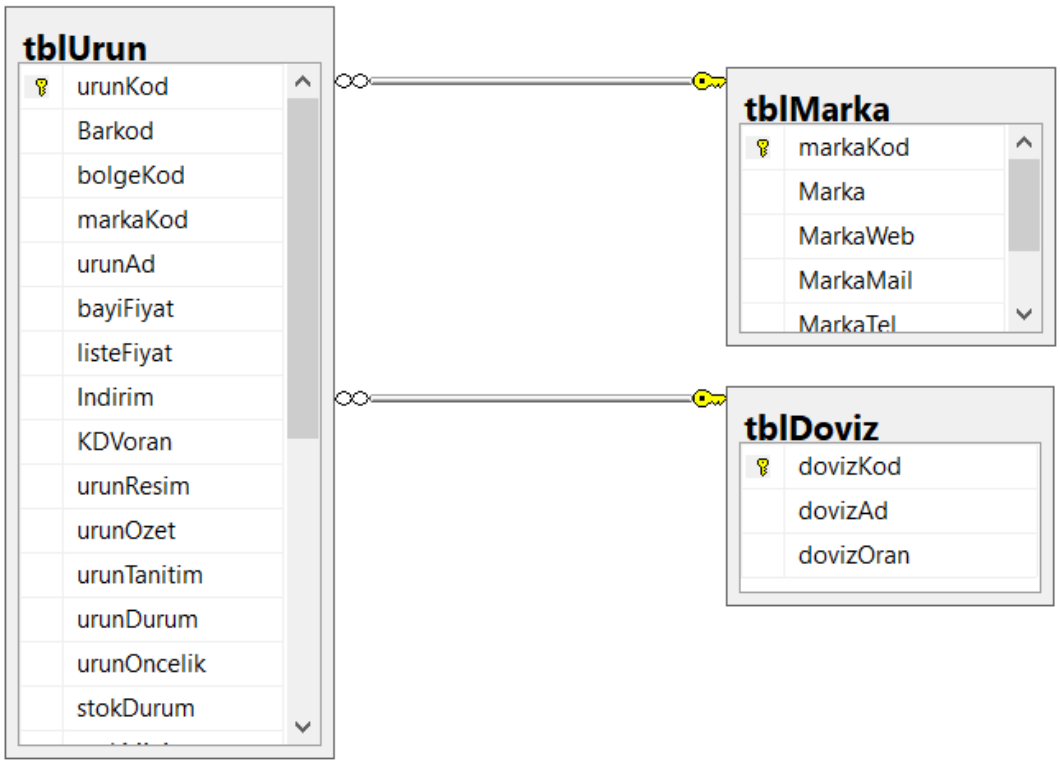
Bir demirbaş programı yazdığımızı varsayalım. tblDemirbas adında bir tabloda demirbaşlara ait bir çok bilgilerin tutulduğunu varsayalım. AlisTarihi, bir demirbaşın ne zaman alındığını TerkinTarihi de ne zaman hurdaya ayrıldığını tutan sütünlar olarak tanımlanmış olsun. Bu durumda, demirbaş terkin tarihinin her zaman boş veya alım tarihinden büyük olduğunu garanti edecek bir CHECK constraint tablo seviyeli olarak şu şekilde tanımlanmalıdır:

```
CREATE TABLE tblDemirbas(
demirbasNo VARCHAR(10),
demirbasAd VARCHAR(250),
alisTarihi DATETIME,
terkinTarihi DATETIME NULL,
CONSTRAINT chkTerkinTarih CHECK (terkinTarih IS NULL OR terkinTarih >= alisTarih) – FOR terkinTarih
)
```

Yukarıdaki kodu çalıştırın ve ilgili tabloya örnek kayıtlar girmeye çalışın. Sonucu gözlemleyin.

Foreign Key Constraint Oluşturmak

Bir tablonun belli sütununa girilecek değerleri başka bir tablonun indekslerle tekilleştirilmiş bir sütunundaki değer kümesi ile sınırlandırmak için kullanılır. Böylece ilişkilendirilen tablo (Foreign Key Table)'nun Foreign Key alanına girilecek değerlerin ilişkilenen tablo (Primary Key Table)'nun birincil alanından geliyorsa kabul edilmesi burada yer almayan değerlerin kabul edilmemesi sağlanmış olur.



T-SQL ile bir tablo oluşturulurken, bu tablonun belli sütunlarını üstünde Foreign key Constraint tanımlamak için şu genel yapı kullanılır:

CRATE TABLE tablo_ismi (

Sütun tanımlamaları…,

CONSTRAINT constraint_ismi

FOREIGN KEY(sutun_ismi)

REFERENCES tablo_ismi(sutun_ismi)

)

Daha önceden oluşturulmuş bir tablonun belli sütunlarını Foreign Key Constraint ile denetlemek için şu yapı kullanılır:

ALTER TABLE tablo_ismi

ADD CONSTRAINT constraint_ismi

FOREIGN KEY(sutun_ismi)

REFERENCES tablo_ismi(sutun_ismi)

Örnek:

Ürünler tablosunda, markaKod sütununa, tblMarka tablosunun markaKod sütunu dışından gelen verilerin, dovizKod sütununa da tblDoviz tablosunun dovizKod sütunu dışından gelecek verilerin kabul edilmemesi istenmektedir. İlgili değişikliği tablo üstünde tanımlayalım:

ALTER TABLE tblUrun

ADD CONSTRAINT FK_tblProduct_tblDoviz FOREIGN KEY (dovizKod) REFERENCES tblDoviz (dovizKod),

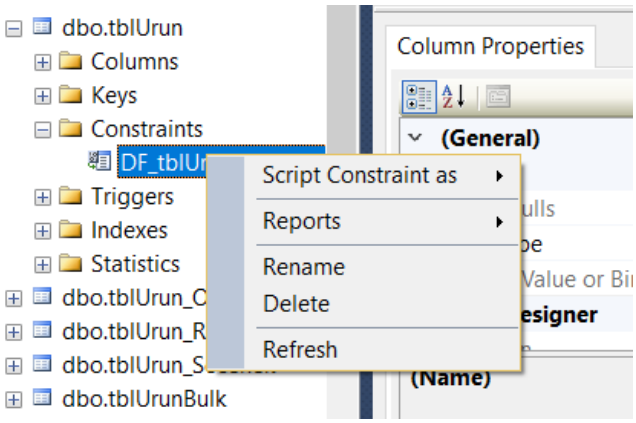
CONSTRAINTS FK_tblProduct_tblMarka FOREIGN KEY (markaKod) REFERENCES tblMarka(markKod)

Ardışık Bütünlük (Cascading Integrity) – dış anahtarın referansta bulunan değerler değiştikçe (silme güncelleme) değişmesi gerekliliği - kavramını araştırın. Kitabın 180-182. Sayfaları

Contraintleri Yönetmek

Constraint'ler üstünde değişiklik yapmak için en uygun yer, Management Studio'dur. Çünkü Transact SQL ile bir constraint silip yeniden oluşturmak, üstüne değişiklik yapmaktan daha kolaydır.

Her bir constraint'in Management Studio ile nasıl oluşturulduğunu öğrendik. Değiştileceği ekranlara da aynı şekilde erişebiliriz.



Constraintleri Silmek

Constraintler istenildiği zaman Management studio üzerinden ya da kod ile silinebilir.

ALTER TABLE tablo_ismi

DROP CONSTRAINT constraint_ismi

Constraintleri Denetime Kapatmak ve Açmak

Constraint'lerin veritabanını bazen anlık olarak denetlemesi istenmez. Bir süre için Constraint'leri susturmak gerekebilir. Buna genellikle bir dış kaynaktan veri alma işlemi boyunca veya bazı kayıtlar üstünde elle düzeltme yaparken ihtiyaç duyulabilir. Bu türden durumlarda şu genel ifadelerden faydalanacağız:

ALTER TABLE tablo_ismi

NOCHECK CONSTRAINT ALL | constraint_ismi

İlgili işlemler yapıldıktan sonra tekrardan Constraint'ler açılabilmek için primary key, unique, foreign key ve check constraint'lerce tanımlı kurallara aykırı veri bulunmuyor olması gerekir. Bu şart sağlanıyorsa, tekrardan constraint'ler şu ifade ile aktif yapılır:

ALTER TABLE tablo_ismi

CHECK CONSTRAINT ALL | constraint_ismi