SQL Server Views

"View nedir? Ne değildir?" sorularına geçmeden önce View'ler hakkında kulaktan kalma bilgilerimizin ne kadarı doğru, ne kadarı yanlış bunları bir gözden geçirelim. View'ler hakkında en çok söylenen söz "Veritabanında ihtiyacın olmasa da istediğin kadar kullan hiç zararı yok, sonra üzerinden işlem yapmak daha kolay oluyor" sözüdür. Bu söz bir açıdan doğru olmasına rağmen yanlış bilgi içerdiği de su götürmez. View'ler, ihtiyaç duyulduğunda yaratılması ve doğru şekilde kullanılması halinde bizi birçok yükten kurtaracak olan veritabanı elemanlarıdır. Eğer yerinde kullanılırlarsa veritabanı üzerinde daha verimli çalışmamıza olanak sağlarlar.

Peki, nedir bu View'ler? **View'ler bir veya birden fazla tablodan istenilen verilerin bir arada sunulmasını sağlayan tanımlanmış sorgulardır**.

Sanal bir tablo olarak da düşünebiliriz.

Aynı tablolar gibi satırlar ve sütunları içerir.

Bir ya da birkaç tablodan seçtiğimiz verileri okuyabiliriz hatta bazı durumlarda veri girişi bile yapabiliriz. Peki neden "sanal" kelimesini kullanıyoruz?

Çünkü **View'ler veri saklamazlar** sadece istenen veriye ulaşılacak yolu kullanarak verileri kullanıcıya sunarlar.

View'leri nasıl yaratacağımız konusuna girmeden, bize ne avantajlar sağlıyorlar biraz bahsedelim çünkü sağladığı avantajları bilirsek ne amaçlar için kullanacağımıza da daha rahat karar veririz. Daha önce gereksiz yere View yaratıp kullanmanın hem bizim için hem de veritabanı için verimsiz olacağından zaten bahsetmiştik.

-Birden çok tablo ile çalışırken gereksiz karmaşadan(özellikle her seferinde uzun bir SQL sorgu cümlesi yazmakla uğraşmaktan) kurtulmak.

-Veri ulaşım performansını arttırmak.

-Veri erişimini sınırlamak ve kontrol altında tutmak.

Buraya kadar kod yazmamamızın en önemli nedeni View hakkında öğreneceğiniz bilgilerin "SQL'de nasıl View yaratılır?" sorusunun cevabından daha yararlı ve karmaşık olmasıdır. Bir diğer neden ise eğer yeterince SQL bilgisine sahipsek zaten bir View'in nasıl yazılması gerektiğini az çok biliyor olmamızdır.

Northwind örnek veritabanında bir View yaratarak devam edelim. Bu veritabanını indirmek için link:

```
USE [Northwind]
GO

CREATE VIEW [dbo].[Order_Customer]
AS
SELECT  Orders.OrderID, Orders.OrderDate, Orders.ShipName, Customers.
CompanyName, Customers.City, Customers.Country
FROM dbo.Customers
INNER JOIN dbo.Orders ON Customers.CustomerID = Orders.CustomerID
```

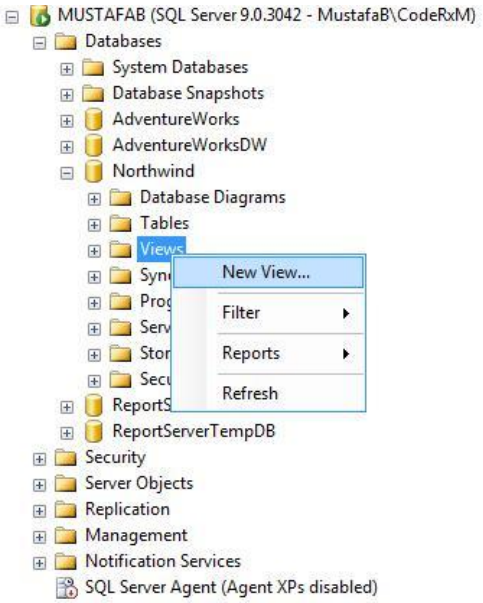
Örnek-1: View oluşturma, Order_Customer ismiyle bir view oluşturuyoruz

```
CREATE VIEW vw_ogrenciNotlari
AS
SELECT o.ad,o.soyad,d.dersAd,n.notu
FROM tbl_ogrenci o
join tbl_ogrenciNot n ON o.ogr_id=n.ogr_id
join tbl_ders d ON n.ders_id=d.ders_id
```

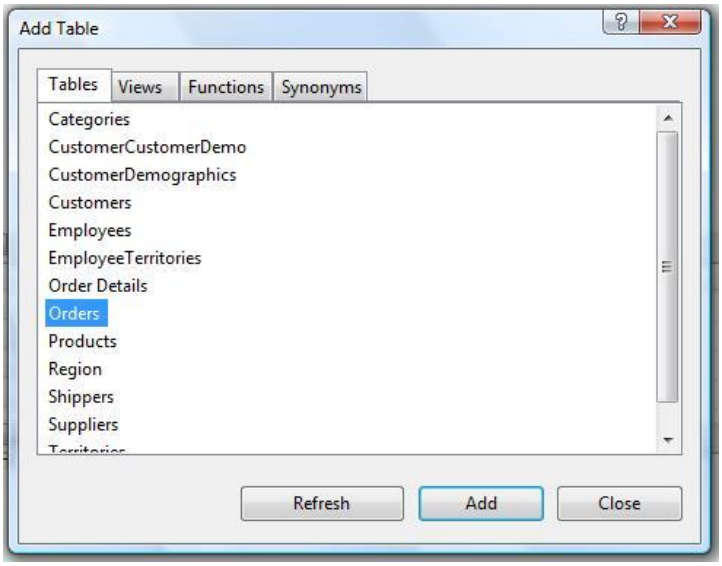
```
select ad,notu
from vw_ogrenciNotlari
where ad='ali'
order by notu
```

SQL bilgisine sahip olmasak bile işlem yapacağımız veritabanı içindeki Views klasörüne sağ tuşla tıklayıp "New View…" seçeneğini seçtikten sonra karşımıza gelen ekrandan tablolar ekleyerek kolaylıkla yaratmak istediğimiz View'i hazırlayabiliriz. Tek yapmamız gereken tablolardan istediğimiz sütunları belirleyip Select cümlemizin otomatik olarak yazılmasını sağlamaktır.

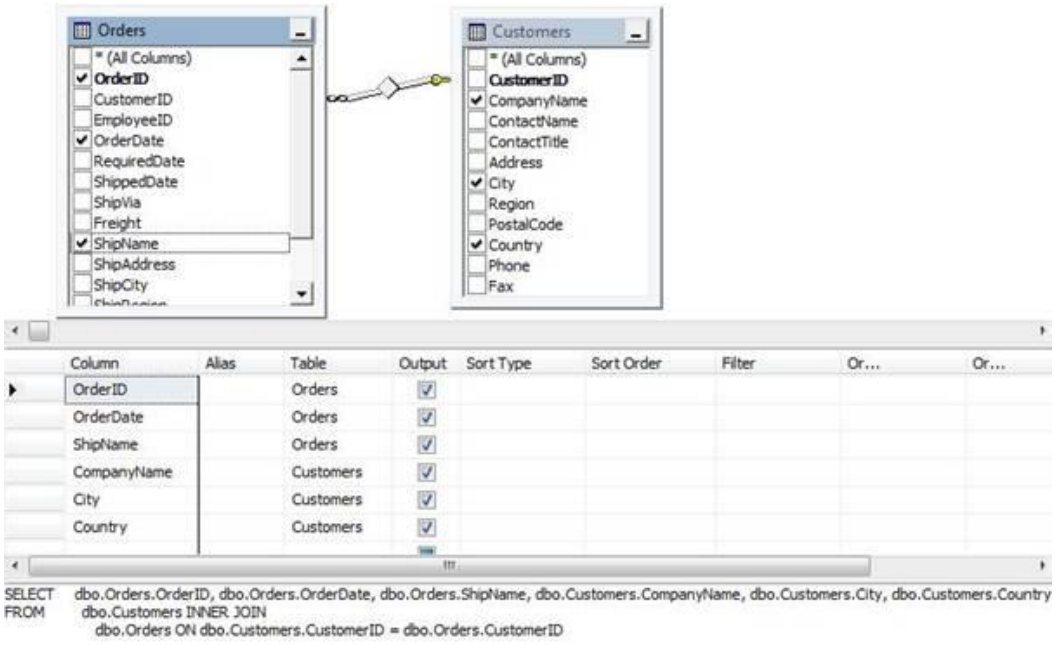
Ekran-1: Yeni View ekleme



Ekran-2: Tablo ekleme



Ekran-3: View kaydetme, Order_Customer ismiyle kaydedelim



View oluştururken dikkat etmemiz gereken hususlar:

Oluşturulan View de;

-Compute ya da Compute By cümleciği,

-Into anahtar sözcüğü,

-Option cümleciği

kullanılamaz.

-View'ler temporary tablo dediğimiz geçici olarak oluşturulan tablolara referans verilemez. Bu tablolardan veri istenemez.

-Order By cümleciği sadece TOP operatörü ile birlikte kullanıldığında kullanılabilir.

-Bir View farklı tablolardan gelen aynı isimlere sahip farklı sütunları içerebilir.

-View'lerdeki sütunlar aritmetik ifadelere sahip olabilir.

Yukarıdaki Örnek-1'de bir View nasıl yaratılır bunu örnekledik. İlk örneğimizi henüz incelemişken gelin bir View'in SYNTAX'i (sentaksı) nasıl olur ona bakalım ve daha sonra ayrıntılı olarak inceleyeceğimiz View'in teknik kısımlarını, yüzeysel olarak görelim. Aşağıdaki sentaksta [ ] arasında yazdıklarımız isteğe ve ihtiyaca göre yazılabilen kısımlardır.

Sentaks(Syntax):

CREATE VIEW *view_ismi* *[(görünen sütun ismi,…)]*

[WITH {ENCRYPTION | SCHEMABINDING}]

AS

select_cümlesi

[WITH CHECK OPTION]

Görüldüğü gibi sadece –

Create View *view_ismi*

As

select_cümlesi

- ile bir view yaratabiliriz. Bizi asıl zorlayacak kısım yazacağımız Select cümlesi olsa gerek, başka bir içerikte kalmadı gibi.

Create View *[sahip_ismi].view_ismi* kısmındaki *[sahip_ismi]* alanını doldurmak zorunda değiliz yani isteğe bağlı olarak doldurulabiliriz. Yalnız daha sonra karışıklık olmaması için o View'in kim tarafından yönetileceğini belirtmek yararımıza olacaktır. Hiçbir isim girilmemesi durumunda o sıradaki kullanıcıyı View'in sahibi olarak atayacaktır.

Create View view_ismi başlangıcından sonra gelen *[(görünen sütun ismi,…)]* yazan bölümde *Select* sorgusu ile sonuçta gösterilecek sütunların isimlendirmesini yapabiliriz. Birden fazla sütun ismi yazmak için sütunların arasına yukarıdaki gibi virgül koyulur.

Bunu test etmek için gelin ilk yaptığımız örneğin sütunlarının Türkçeleştirilmiş halini, yeni bir View olarak oluşturalım. İki örneği kıyaslayıp sonuçlarını görüntülediğimizde farkı görebiliriz.

Örnek-2: View oluşturma, Siparis_Musteri ismiyle yeni bir view oluşturalım

```
USE [Northwind]
GO

Create VIEW [dbo].[Siparis_Musteri] ([Sipariş Numarası], [Sipariş Tarihi], [Gemi İsmi],
[Alıcı Şirketin Adı], [Alıcının Şehri], [Alıcının Ülkesi])
AS
SELECT  Orders.OrderID, Orders.OrderDate, Orders.ShipName, Customers.
CompanyName, Customers.City, Customers.Country
FROM dbo.Customers
INNER JOIN dbo.Orders ON Customers.CustomerID = Orders.CustomerID
```

```
CREATE VIEW vw_ogrenciDersleriYeni
(ogrenciNo,ogrenciAd,ogrenciSoyad,bolum,dersAd,harfNot)
AS

SELECT o.ogrNo, o.ad, o.soyad, o.bolum, d.dersAd, n.harfNot
FROM Ogrenci o
JOIN Notlar n ON o.ogrNo=n.ogrNo
JOIN Dersler d ON n.dersKod=d.dersKod
```

Ekran-4: Siparis_Musteri View'inin sonucu(üstteki sütun isimlerine dikkat!)

Sipariş Numarası	Sipariş Tarihi	Gemi İsmi	Alıcı Şirketin Adı	Alıcının Şehri	Alıcının Ülkesi
10248	04.07.1996 00:...	Vins et alcools C...	Vins et alcools C...	Reims	France
10249	05.07.1996 00:...	Toms Spezialitäten	Toms Spezialitäten	Münster	Germany
10250	08.07.1996 00:...	Hanari Carnes	Hanari Carnes	Rio de Janeiro	Brazil
10251	08.07.1996 00:...	Victuailles en stock	Victuailles en stock	Lyon	France
10252	09.07.1996 00:...	Suprêmes délices	Suprêmes délices	Charleroi	Belgium
10253	10.07.1996 00:...	Hanari Carnes	Hanari Carnes	Rio de Janeiro	Brazil
10254	11.07.1996 00:...	Chop-suey Chin...	Chop-suey Chin...	Bern	Switzerland
10255	12.07.1996 00:...	Richter Supermarkt	Richter Supermarkt	Genève	Switzerland
10256	15.07.1996 00:...	Wellington Impo...	Wellington Impo...	Resende	Brazil
10257	16.07.1996 00:...	HILARION-Abastos	HILARION-Abastos	San Cristóbal	Venezuela

Geldik View kullanmayı kulaktan dolma öğrenmiş ve View oluşturmayı bildiğini iddia eden bazı insanların kullanmaya aşina olmadığı bölümlere.

[WITH {ENCRYPTION | SCHEMABINDING}]

Encryption:

Öncelikle *With Encryption* cümlesinin ne işe yaradığından bahsedelim biraz. Aşağıda yaptığımız örneği direk alıp kopyalarsak oluşturduğumuz View'den sonuçta ne olduğunu büyük olasılıkla anlayamayacağız.

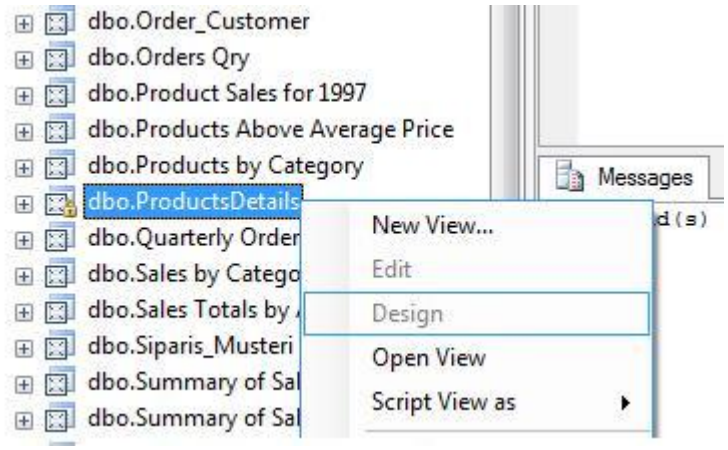
Encryption yapısını kullanmamızın nedeni o veritabanında oluşturduğumuz View'in kod içeriğinin yani View'i oluşturmamızı sağlayan kaynak kodun görüntülenemez şekilde şifrelenmesidir. Görüntülenemez çünkü Encryption yaptığımız bir View'i Decryption gibi bir işlemle geri alamıyoruz. Şifresiz haline döndüremiyoruz. Bunu yapabilmek için View'in kaynak koduna ihtiyaç duyuyoruz. Kaynak kodunu bildiğimiz bir View'in şifrelenmiş kaynak kodunu niye açmaya çalışalım ki sonuçta. Bu yüzden Encryption(şifreleme) yapmayı düşündüğümüz View cümlesinin yedeğini kaydetmeyi unutmazsak bizim için daha iyi olur. Tekrar bu cümleyi oluşturmak zorunda kalacağımız durumlarda oradan alıp kullanabiliriz.

Örnek-3: With Encryption yapısı

```
USE [Northwind]
GO

Create VIEW [dbo].[ProductsDetails]
With Encryption
AS
SELECT    Prod.ProductName, Cat.CategoryName, Prod.QuantityPerUnit, Prod.UnitPrice
FROM      dbo.Products as Prod INNER JOIN
              dbo.Categories as Cat ON Prod.CategoryID = Cat.CategoryID
```

Ekran-5: Encryption özelliği aktif bir View'in ekrandaki görüntüsü



Daha önce bahsettiğimiz gibi Encryption yaptığımız View'in kaynak kodunu görmemiz mümkün değil. Örnekte oluşturduğumuz View'e sağ tuşla tıkladığımızda karşımıza çıkan *Modify veya Design* seçeneği artık inaktif durumda. View'lerin içeriğini görüntülemenin bir başka yolu da bir stored procedure(saklı yordam) kullanmak, **sp_hepltext.**

sp_helptext 'ÖrnekViewinİsmi'

Bu stored procedure'e Encryption yaptığımız örneğimiz üzerinde işlem yapmasını söylersek daha önce Encryption yapıldığı için sonuçta bir hata mesajı almamız kaçınılmaz. Fakat biz bunu öncelikle Encryption yapılmamış bir View üzeride deneyelim. Ben ilk yaptığımız View örneğini kullanarak *sp_helptext* stored procedure'ünü çalıştıracağım.

Örnek-4: sp_helptext Order_Customer

```
sp_helptext Order_Customer
```

Results | Messages

	Text
1	
2	CREATE VIEW [dbo].[Order_Customer]
3	AS
4	SELECT Orders.OrderID, Orders.OrderDate, Orders.ShipName, Customers.CompanyName, Customers.City, Customers.Country
5	FROM dbo.Customers
6	INNER JOIN dbo.Orders ON Customers.CustomerID = Orders.CustomerID

Örnek-5: sp_helptext ProductsDetails(Encryption özelliği açık bir view'in kaynak içeriği görüntülenemez)

```
sp_helptext ProductsDetails
```

Messages

```
The text for object 'ProductsDetails' is encrypted.
```

SchemaBinding:

Encryption kelimesi kadar olmasa da Schemabinding kelimesinin tercümesi, zaten bize ne amaçlar için kullanabileceğimizi anlatmaya yeter gibi gözüküyor. Eğer oluşturacağımız View'de *With Schemabinding* kullanırsak ya da daha önce oluşturduğumuz bir View'e *ALTER View* ile *With Schemabinding* özelliği verirsek bu View'de görmek istediğimiz verilerin tutulduğu tablolardaki veri çektiğimiz sütunları kilitlemiş oluruz.

Bir View'e Schemabinding özelliği eklemek istiyorsak hatırlamamız gereken iki durum var.

-Schemabinding'li View'imizde * kullanamayız. Yani *select * from ...* diye bir cümle kullanmaktansa bütün sütunların isimlerini tek tek girmeliyiz. Eğer aşağıdaki gibi kullanırsak Schemabinding'in neden olduğu bir hata mesajı ile karşılaşırız.

Örnek-6: Schemabinding ve *

```
USE [Northwind]
GO

Create VIEW [dbo].[Kategoriler]
With Schemabinding
AS
SELECT     dbo.Categories.*
FROM       dbo.Categories
```

```
Messages
Msg 1054, Level 15, State 7, Procedure Kategoriler, Line 5
Syntax '*' is not allowed in schema-bound objects.
```

-Schemabinding kullanıyorsak View'imizin Select cümlesinde referans ettiği tabloların isimleri *tablosahibininismi.tabloismi* halinde olmalıdır. İlk olarak hatalı örnek, sorun halinde karşımıza çıkan hata ekranı ve sorunsuz çalışan halinin örneğini aşağıda da görebilirsiniz.

Örnek-7: Schemabinding ve tablosahibininismi.tabloismi yanlış kullanım

```
USE [Northwind]
GO

Create VIEW [dbo].[Kategoriler]
With Schemabinding
AS
SELECT     CategoryID, CategoryName, [Description], Picture
FROM       Categories
```

```
Messages
Msg 4512, Level 16, State 3, Procedure Kategoriler, Line 5
Cannot schema bind view 'dbo.Kategoriler' because name 'Categories' is invalid for schema bindin
```

Örnek-8: Schemabinding ve tablosahibininismi.tabloismi doğru kullanım

```
USE [Northwind]
GO

Create VIEW [dbo].[Kategoriler]
With Schemabinding
AS
SELECT     CategoryID, CategoryName, [Description], Picture
FROM       dbo.Categories
```

```
Results
Command(s) completed successfully.
```

Yukarıdaki örnekte View'de görüntüleyeceğimiz verilerin bulunduğu tablo ya da tablolardaki sütunların yapısında yapılacak herhangi bir değişikliği bu View'i silmedikçe ya da Schemabinding özelliğini View'imizden kaldırmadıkça yapamayacağız. Oldu da böyle bir işlemi Schemabinding özelliği varken yapmaya kalkıştık, SQL bize bu işlemin yapılamayacağını belirten bir hata mesajı verecektir. Kategoriler isimli View için kullandığımız dbo.Categories tablosundan CategoryName sütununu kaldırmaya çalışalım.

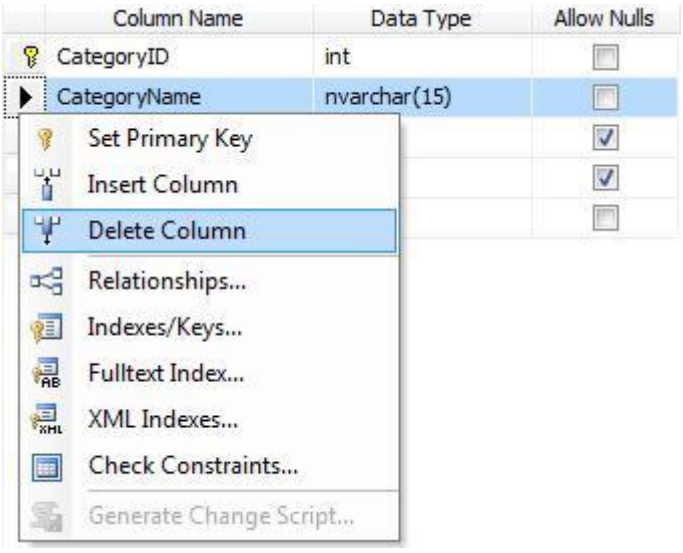
Örnek-9: Schemabinding ve tablo yapısında değişiklik yapma uyarısı

```
Alter Table Categories Drop Column CategoryName
```

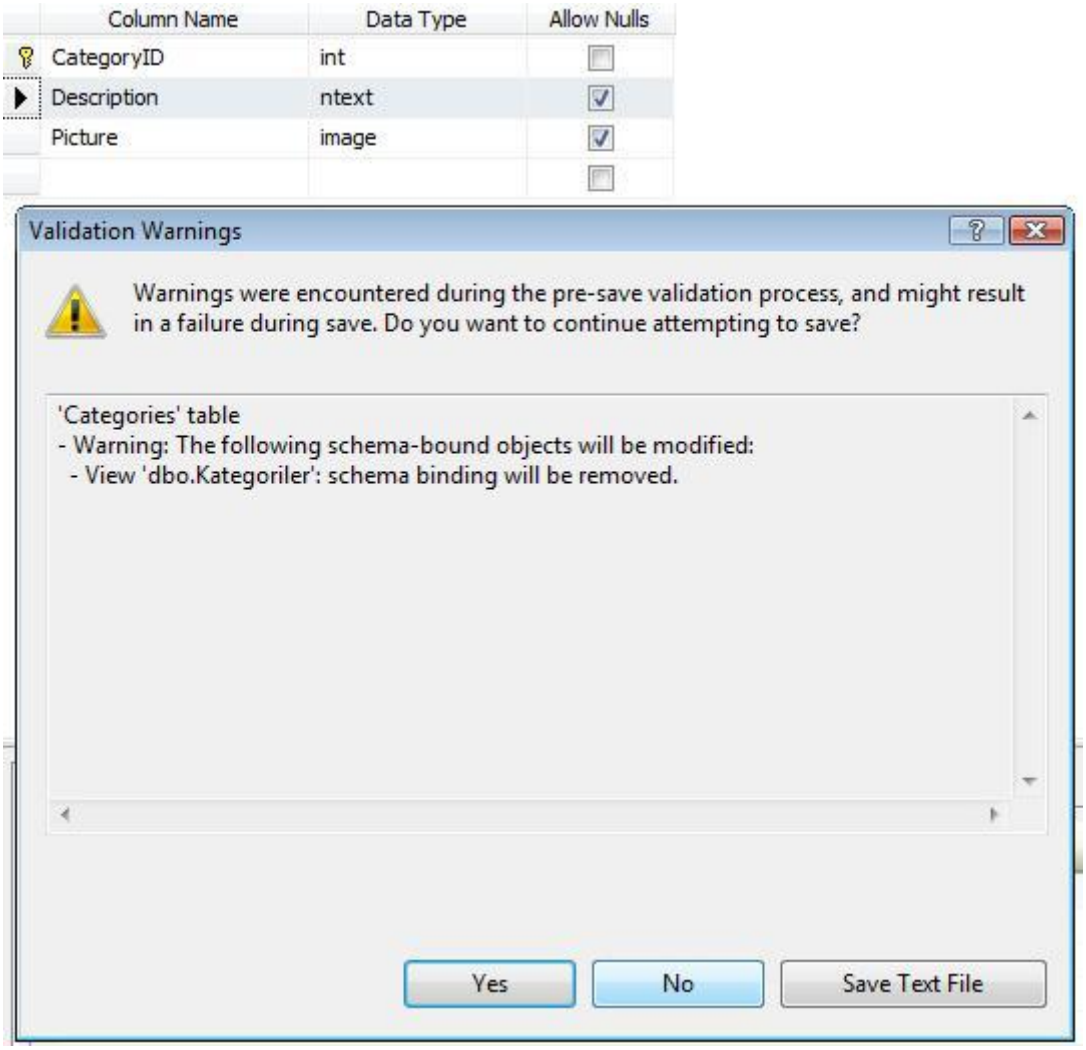
```
Messages
Msg 5074, Level 16, State 1, Line 1
The object 'Kategoriler' is dependent on column 'CategoryName'.
Msg 4922, Level 16, State 9, Line 1
ALTER TABLE DROP COLUMN CategoryName failed because one or more objects access this column.
```

Bu sütunların yapısında değişiklik yapmayı kod tarafından değil de dizayn taraflı yönetim ekranından yapacak olursak ve başka hiçbir engel söz konusu değilse bize var olan değişikliği yapmak için Schemabinding'i kaldırmak isteyip istemediğimizi soracaktır. Eğer Schemabinding'i kaldırma isteğini onaylarsak Schemabinding'i kaldırıp istediğimiz yapısal değişiklikleri de yapacaktır.

Ekran-6: Schemabinding varken tablo yapısında değişiklik yapma



Ekran-7: Schemabinding kaldırma onay ekranı



Schemabinding özelliği View'de görüntülediğimiz verilerin sütun yapısına kilit koruması koyduğundan özellikle çok büyük veritabanları tasarlayan insanlar oluşturdukları her tablo için her sütunu içeren ve Schemabinding özelliği etkin bir view oluşturmaktan çekinmezler. Bunun nedeni daha çok yanlışlıkla geri dönüşü uzun süre tutacak bir yapısal değişiklik yapmaktan korunmaktır. Bu açıdan düşünecek olursak veritabanını veri girişi vb. yapısal olmayan değişikliler için inceleyen birinin tabloların yapısal durumunda değişiklik yapmasını engelleyebiliriz.

With Check Option:

Makaleye başlarken bahsettiğimiz gibi birçok insan View'leri sanal tablo olarak isimlendirmektedir. Bu sadece sözde kalan bir benzetme de değildir. View'leri kullanarak tablolar üzerinde DML(Data Manipulation Language) SQL komutları çalıştırabiliriz; *Delete, Insert, Select, Update.*

View'imiz üzerinden verilerde değişiklik yapmaya çalışırken göz önünde bulundurmamız gereken şartlar;

-Distinct yapısı kullanılmış View'lerin üzerinden Insert, Update ya da Delete gibi işlemler yapamayız. Aynı durum Group By kullandığımız View'ler içinde geçerlidir.

-View kullanarak birden fazla tabloda veri değişikliği yapamayız.

Gelin bu komutların View ile nasıl kullanıldığına dair bir örnek yapalım.

Örnek-10: Önce MusteriListele isminde yeni bir View oluşturalım

```
USE [Northwind]
GO

Create VIEW [dbo].[MusteriListele]
AS
SELECT    CustomerID, CompanyName, ContactName, ContactTitle, [Address],  City
FROM       dbo.Customers
WHERE    City = 'Ankara'
```

Örnek-11: Şimdi ise bu View'i kullanıp yeni bir müşteri ekleyelim

```
Use [Northwind]
GO
Insert Into MusteriListele Values('MMMMM','Bodur Makale','Mustafa Bodur','Müdür',
'Beşiktaş','Istanbul')
```

Messages

```
(1 row(s) affected)
```

Ekran-8: Girilen verinin görüntüsü

MEREP	Mere Paillarde	Jean Fresniere	Marketing Assist...	43 rue St. Laurent	Montreal	Quebec
MMMMM	Bodur Makale	Mustafa Bodur	Müdür	Besiktas	Istanbul	NULL
MORGK	Morgenstern Ge...	Alexander Feuer	Marketing Assist...	Heerstr. 22	Leipzig	NULL

Eğer View'imizi bizim yaptığımız gibi belirli bir WHERE cümlesi kullanarak yaratıp az sonra örneğini inceleyeceğimiz gibi With Check Option özelliğini de bu View'imize eklediysek artık View üzerinden DML kullanarak yaptığımız işlemlerde View cümlesinin şartlarına uymayan veri değişikliğini, veri girişini engellemiş oluruz. Yani az önceki işlem bu koşullar altında hata verecektir.

Örnek-12: MusteriListele View'ine - With Check Option özelliği verelim

```
USE [Northwind]
GO
Alter VIEW [dbo].[MusteriListele]
AS
SELECT    CustomerID, CompanyName, ContactName, ContactTitle, [Address],  City
FROM       dbo.Customers
WHERE    City = 'Ankara'

With Check Option
```

Örnek-13: Şehri İstanbul olan bir kaydı tekrar girmeyi deneyelim

```
Use [Northwind]
GO
Insert Into MusteriListele Values('SSSSS','İlk Bodur Company','Serdar Bodur',
'Müdür','Beşiktaş','Istanbul')
```

CREATE VIEW cümlesinin sentaksını bitirdikten sonra artık derlediğimiz View'i değiştirmek için *ALTER VIEW* ve oluşturduğumuz View'i silmek için *DROP VIEW* cümlelerine kısaca geçebiliriz. Makalenin başlangıcından sonuna kadar veritabanı konusunda az çok ilgisi olan bizlerin DDL(Data Definition Language) ile ilgili başlangıç bilgilerini bildiğini varsaydığım için DDL anlatmadım. Yalnız *ALTER VIEW* ve *DROP VIEW*'den örnekler vermeden de bu makale View hakkında tam bir makale olamayacaktır.

Örnek-14: Kategoriler View'inden Schemabinding özelliğini kaldıralım

```
USE [Northwind]
GO

ALTER VIEW [dbo].[Kategoriler]
AS
SELECT     CategoryID, CategoryName, [Description], Picture
FROM       dbo.Categories
```

Örnek-15: Son olarak Kategoriler View'ini silelim

```
USE [Northwind]
GO

DROP VIEW [dbo].[Kategoriler]
```

Alıntı : Yaşar GÖzüdeli