VERİ TABANI VE YÖNETİMİ DERSİ – UYGULAMA FÖYLERİ

Föy-5

Konu: Stored Prosedürler (SP), Parametreli SP'ler, SP'lerin seçeneklerini ayarlamak, yönetmek ve değiştirmek

GENEL BİLGİLER (Ayrıntılı Bilgi için Kitabın 15. Ünitesi incelenebilir.)

Stored Prosedürlerle (Saklı Yordam) Çalışmak

Nesneye dayalı programlama bu kadar popüler değilken, programlar sadece prosedür ya da fonksiyon denilen parçacıklardan oluşurdu. Her prosedür, başka bir işlevi yerine getirmek için özellikle yapılandırılmış program parçacığıdır. Bir prosedür, başka bir prosedür içinden çağrılabilir. Bu da sık kullanılan işlemler için yazılmış kodların bir defa yazılıp çok defa kullanılmasını böylelikle de programlamayı kolaylaştırmayı amaçlar. Ancak bunun dışında, Stored Prosedür (SP)'ler veritabanı programlama için güvenlik unsuru olarak da kullanılmaktadır. Örneğin bir tabloya seçme (SELECT) yetkisi vermek yerine tablodan sadece belli bir seçme işlemi yapan prosedüre yetki verilerek daha küçük bir yetki ile verilerin okunması sağlanabilir.

Stored prosedürler, birçok gelişmiş programlama dilindeki fonksiyon yapılarına karşılık gelir.

En büyük özelliği sorguların önceden hazırlanması(derlenmesi) ile VTYS ile aynı uzayda çalışmasından dolayı daha hızlı sonuç vermesidir. Bir SP oluşturulduktan sonra, veritabanı sunucusunda saklanır. Her ihtiyaç duyulduğunda aynı SP defalarca çağrılabilir. Ancak SQL Server içerisinde en fazla 32 seviyeye kadar SP'lerin bir başka Sp çağırmasına müsaade edilir.

<u>SP'ler oluşturuluş şekline göre dört tiptir:</u>

→ ***Extended Stored Prosedürler:***

Genellikle *.dll şeklinde prosedürlerdir. C,C++,VB gibi T-SQL dışındaki dillerle yazılıp dll'e derlenirler.

→ ***CLR Stored Prosedürler:***

SQL Server 2005 sonrasında CLR ortamında herhangi bir dili kullanarak kodlanan SP'lerdir.

→ ***Sistem Stored Prosedürleri***

Genellikle sp_ ön ekiyle başlarlar ve hepsi master veri tabanında tutulan SP'lerdir.

→ *Kullanıcı Tanımlı Stored Prosedürler:*

Programcının yani bizim programladığımız SP'lerdir. 3 tip SP vardır.

Geçici: SP özellikle SQL Server'in eski sürümlerinde kullanılabilen bir tür olup, her oturumda derlenmesi gereken SP'ler için kullanılırdı.

Yerel: SP'lere verilen genel addır.

Uzak: SP, RPC desteğinde bir dağıtık modelde, uzaktaki sunucuda yer alan sp için kullanılır.

Bir SP oluşturulduktan sonraki ilk çalıştırmasında şu aşamalara tabi tutulur:

Aşama	Yapılar
Ayrıştırma (Parsing)	İfadelerin geçerliliği denetlenir. Bu aşamada geciktirilmiş isim çözümleme desteklenir. Yani olmayan bir tablodan seçme yapılıyorsa bu aşamada bir hata oluşmaz. Sorgu Ağacı (query tree) ya da Sıra Ağacı (sequence) denilen yapı ortaya çıkarılır.
Derleme (Compiling)	Bir önceki aşamada oluşturulan sıra ağacı ele alınarak çalıştırma planı (execution plan) çıkartılır. Çalıştırma planı güvenlik, hak ve yetkiler dahilinde denetlenir. Çalıştırma planı, hangi aşamada hangi kontrollerin, contraint'lerin veya indekslerin kullanılacağını veya tablo taraması yapılacağı gibi tanımlamaları da içerir
Çalıştırma (Executing)	Bir önceki aşamada oluşturulan çalıştırma planı ele alınarak işlemler gerçekleştirilir. Örneğin bir SELECT ifadesi yer alıyorsa, sorgu Veri İşlemeden sorumlu DML yöneticisine iletilir.

Ancak SP normal şartlarda ikinci defa çalıştırılırken, aynı aşamaları takip etmez. Çünkü derlenme işleminden sonra elde edilen çalıştırma planı, Prosedür Hafızası (procedure cash) denilen bir hafızada saklanır. Takip eden çağırma istekleri, prosedür hafızasındaki çalıştırma planı kullanılarak cevaplanır ve ilk iki aşama atlandığı için normal şartlarda daha hızlı neticeye ulaşılır. Bir script bloğu çalıştırmaya göre stored prosedür çalıştırmayı avantajlı kılan nokta burasıdır.

Stored Prosedür kullanmanın faydaları:

Her seferinde iş-sunum katmanı arasında oluşabilecek yoğun veri-alış verişi bazı hallerde, stored prosedür ile veri katmanındaki işlemlere dönüştürülebilir.

Stored Prosedür Oluşturmak

CREATE PROC[EDURE] prosedür adi

[WITH seçenekler]

AS

Prosedur-adi, prosedüre verilecek adı gösterir. Her prosedür dışarıdan parametre almayabilir. Bazı prosedürler çıkış parametresi de alabilir. Bunu belirtmek için OUTPUT deyimi kullanılır. OUTPUT parametreler, konu içerisinde ele alınacaktır. WITH deyimini kullanarak bir stored prosedür'ün kaynak kodunu gizlemek gibi ek seçenekler belirtebiliriz. Stored Prosedür'ün sonuna GO koymak bir zorluluk değildir ancak ciddi hatalardan sizi kurtarabilir.

Örnek_1:

```
CREATE PROC SP$gunlukSatis
AS
        SELECT COUNT(*)
        FROM tblSiparisDetay SD INNER JOIN tblSiparis S
ON SD.faturaKod=S.faturaKod
WHERE S.siparisTarih > GETDATE()-1
AND S.siparisTarih < GETDATE()+1
GO
```

Yukarıdaki SP'yi oluşturunuz. Görevini irdeleyiniz.

Var olan (daha önceden yazılmış) bir stored prosedürün kaynak koduna erişmek için

Sp_helptext 'sp_adi'

Kodu kullanılır.

Bir önceki örnekte yazmış olduğunuz stored prosedürün kaynak kodunu görüntüleyin.

Stored Prosedür Çalıştırmak

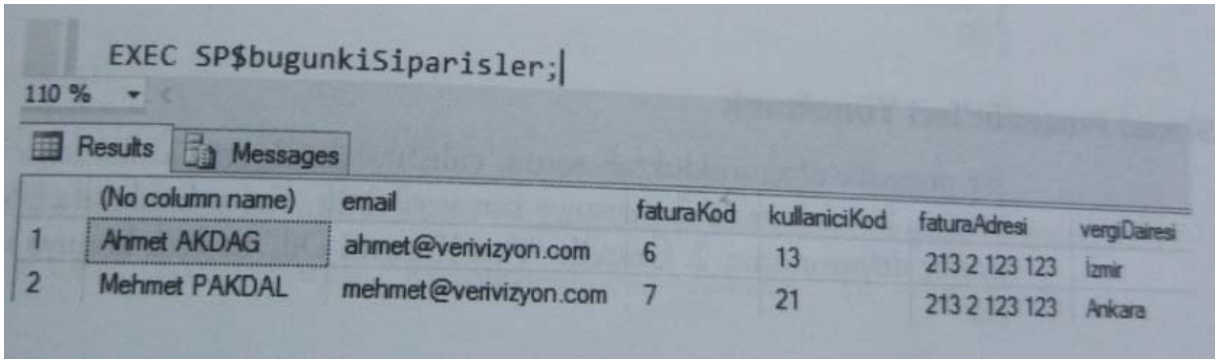
Bir stored prosedür yazıldıktan sonra sadece parse işlemi gerçekleştirilir. Ancak ilk çalışma anında diğer iki işlem olan compile ve execute işlemleri gerçekleştirilir. Bir SP'de gerçekte olmayan bir tablo adı geçiyorsa bununla ilgili hatayı ilk çalıştırma anında alırız.

Bir stored prosedür, bulunduğu batch'in ilk ifadesi ise sadece adını yazmak çalıştırmak için yeterlidir. Ama batch'in ilk ifadesi değil ise, EXECUTE veya EXEC deyiminden sonra adını vermek gerekir.

Örnek_2:

Son bir gün içerisinde verilen siparişlerle ilgili genel bilgileri döndüren bir SP yazalım:

```
CREATE PROC SP$bugunkisiparisler
AS
SELECT K.isim + ' ' + K.soyad, K.email, S.*
FROM tblSiparis S JOIN tblKullanici K
ON K.kullaniciKod = S.KullaniciKod
WHERE siparisTarih < GETDATE() – 1 AND siparisTarih > GETDATE() + 1
```



	(No column name)	email	faturaKod	kullaniciKod	faturaAdresi	vergiDairesi
1	Ahmet AKDAG	ahmet@verivizyon.com	6	13	213 2 123 123	İzmir
2	Mehmet PAKDAL	mehmet@verivizyon.com	7	21	213 2 123 123	Ankara

SP'yi çalıştırın ve yukarıdaki gibi bir çıktı almak için ilgili tablolara gerekli verileri girin.

NoCount Oturum Parametresinin Kullanımı

Bir oturum parametresi olan NOCOUNT'un açılış ve kapanışı için kullanılması gereken genel ifade şu şekildedir:

SET NOCOUNT {ON | OFF}

NOCOUNT Oturum parametresi, SQL Server'da bir performans artış unsuru olarak kullanılabilir. Çünkü SQL Server her işlemden sonra etkilenen kayıt sayısını mesaj olarak döndürmek üzere fazladan bir döngü çalıştıracak şekilde ayarlı iken, bu oturum parametresi açıldığında kaç kaydın etkilendiğinin sonuç olarak döndürülmesi ile ilgilenmez.

SQL Server kaç kaydın etkilendiğini özellikle bir istemci yazılımdan gelen sorgunun sonucu olarak, istemci yazılıma geri bildiriyorsa, bu süreç daha fazla kaynak tüketen bir işlem halini alır. Bu nedenle performans arttırıcı bir yöntem olarak, kaç kaydın etkilendiği ile ilgilenmiyorsanız sorgudan önce oturum parametresi şu şekilde açılabilir:

```
SET NOCOUNT ON
```

Örnek_3:

Az önceki NOCOUNT oturum parametresini performans arttırıcı unsur olarak göz önüne alarak şu şekilde değiştirebiliriz:

```
CREATE PROC SP$bugunkiSiparisler

AS

SET NOCOUNT ON ---- kaç kayıt eklendiğini sonuç olarak döndürme

SELECT K.isim + ' ' + K.soyad , K.email, S.*

FROM tblSiparis S JOIN tblKullanici K

ON K.kullaniciKod = S.KullaniciKod

WHERE siparisTarih < GETDATE()-1 AND siparisTarih > GETDATE()+1
```

Yukarıdaki SP'yi oluşturunuz ve çıktısını gözlemleyiniz.

Stored Prosedürlerde Değişiklik Yapmak

Stored Prosedürleri değiştirmek için ALTER deyimi kullanılır. Genel kullanımı şu şekildedir:

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

Örnek_4:

İlk olarak 'SP$GunlukSatis' SP'sinin kodlarını görüntüleyin.

İpucu: Kaynak kodları görüntülemek için hangi sistem sp'in kullanıldığını hatırlayın!

Daha sonra bu kodu sonuç penceresinden kopyalayıp Query ekranına yapıştırın ve aşağıdaki hale getirin:

```
ALTER PROC SP$GunlukSatis
WITH ENCRYPTION
AS
SET NOCOUNT ON
        SELECT COUNT(*)
        FROM tblSiparisDetay SD JOIIN tblSiparis S
        ON SD.faturaKOD = S.faturaKOD
        WHERE S.siparisTarih > GETDATE()-1
        AND S.siparisTarih < GETDATE()+1
SET NOCOUNT OFF
GO
```

Kodu çalıştırın ve bu sp'nin kaynak kodunu tekrar görüntülemeye çalışın. Sonucu gözlemleyin.

WITH ENCRYPTION parametresinin görevini tartışın.

Stored Procedürlerde Parametrelerle Çalışmak

Prosedürler, dışarıdan girdi parametreleri ile değer alabilir ya da dışarıya değer döndürmek üzere çıkış parametreleri alabilirler.

Girdi Parametreleri

Prosedürleri daha işlevsel bir hale getirebilmek için çoğu zaman dışarıdan parametre almalarına ihtiyaç duyarız. Bu nedenle girdi parametreler kullanılır. Girdi parametreler, stored prosedürlerin işlevselliğini arttırır.

Girdi parametre ile stored prosedür tamınlamanın genel hali şu şekildedir:

```
CREATE PROC[EDURE] prosedür-ismi
        @parametre-ismi veri-tipi [=default-deger]
        [,…n]
[WITH seçenekler]
AS
        T-SQL ifadeleri
GO
```

Örnek_6:

Bir kullanıcı kodunu girdi parametresi olarak aldığında, bu kullanıcıya ait tblSepet'de yer alan ürünlerin kodunu, adını ve adetini döndüren bir SP tanımlayalım:

```
CREATE PROC SP$sepetim(
        @kullaniciKod INT
)
AS SET NOCOUNT ON;
        SELECT S.UrunKod, U.UrunAd, S.Adet
        FROM tblSepet S INNER JOIN tblUrun U  ON U.UrunKod = S.UrunKod
        WHERE S.KullaniciKod = @KullaniciKod;
SET NOCOUNT OFF;
GO
```

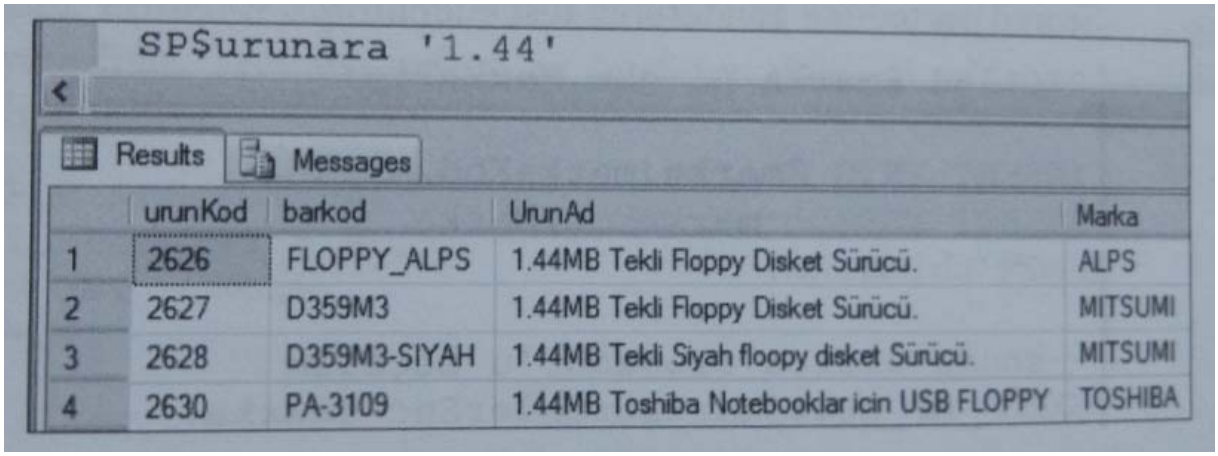
Her iki şekilde de SP$sepetim prosedürünü çağıralım:

SP$sepetim @kullaniciKod=2;

EXEC sp$sepetim 2;

Yukarıdaki kodları çalıştırın ve sonucu gözlemleyin.

Aşağıdaki çıktıyı üretecek SP$urunara girdi parametreli SP'yi kodlayın.



	urunKod	barkod	UrunAd	Marka
1	2626	FLOPPY_ALPS	1.44MB Tekli Floppy Disket Sürücü.	ALPS
2	2627	D359M3	1.44MB Tekli Floppy Disket Sürücü.	MITSUMI
3	2628	D359M3-SIYAH	1.44MB Tekli Siyah floopy disket Sürücü.	MITSUMI
4	2630	PA-3109	1.44MB Toshiba Notebooklar icin USB FLOPPY	TOSHIBA

Çıktı Parametreleri

OUTPUT opsiyonu dışarıdan gelen bir değişkenin içerisinin doldurularak dışarıya tekrar gönderilmesi esasına dayalı olarak dışarıya parametre göndermede kullanılır. Yani bir prosedür kendisini çağıran başka bir prosedüre veya çalıştırıldığı yere kendi ürettiği bir çıktı değerini yanıt olarak verebilir.

```
CREATE PROC[EDURE] presedur_adi
        @parametre ismi veri_tipi [=standart_deger] OUTPUT
        [ ,…n ]
 [WITH seçenekler]
AS
        T-SQL ifadeleri..
GO
```

Örnek_7:

```
CREATE PROC ortaliyici(
@sayi1 smallint,
@sayi2 smallint,
@sayi3 smallint,
@ortalama decimal(18,2) OUTPUT
)
AS
SET NOCOUNT ON
SELECT @ortalama = (@sayi1 + @sayi2 + @sayi3)/CAST(3 AS DECIMAL)
SET NOCOUNT OFF
GO
```

Yukarıdaki sp'yi tanımlayın ve ilgili parametrelerle çağırın.

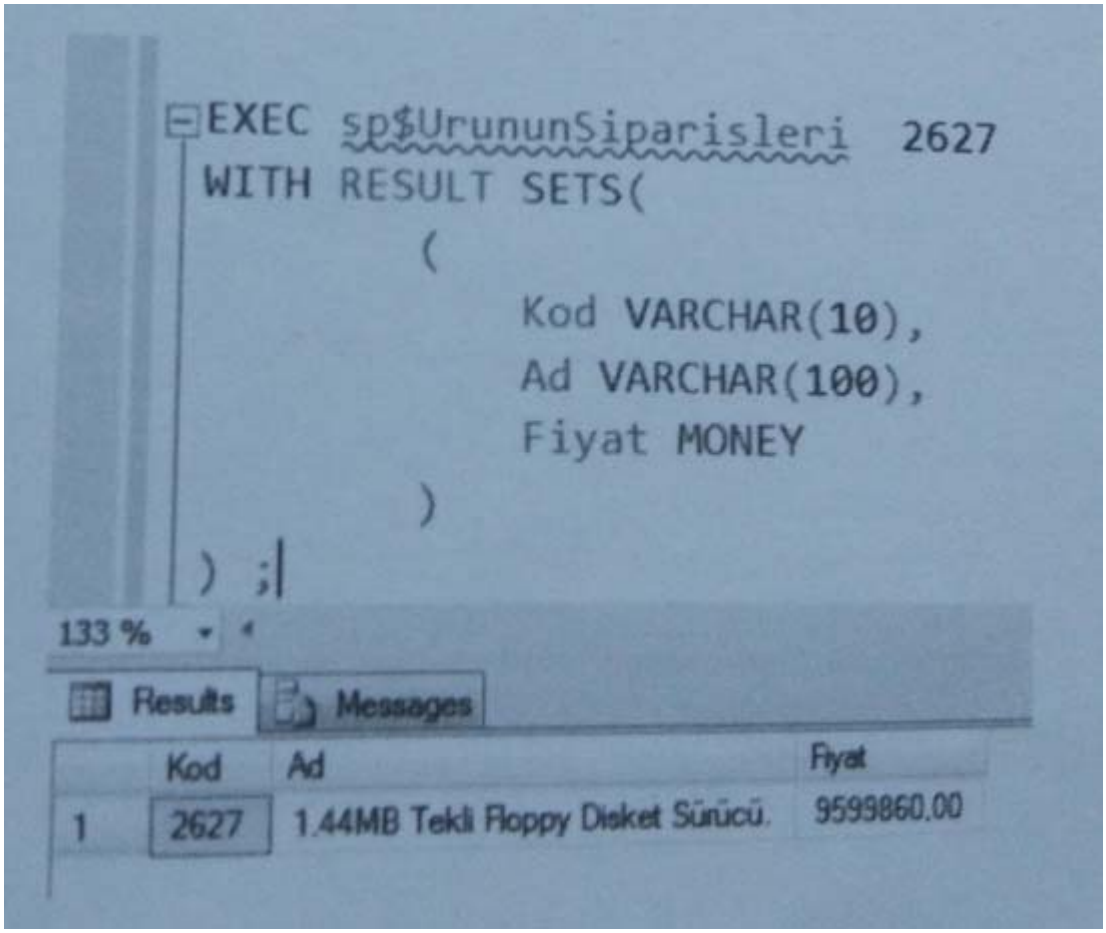
Stored Procedürleri "With Result Sets" ile çağırmak

Bazen stored prosedürlerden dönen ilk sonucun farklı bir sütun adı ile veya farklı bir veri tipinde döndürülmesini isteyebilirsiniz. Bu durumlarda WITH RESULT SETS yan cümlesi ile prosedür çağırılabiliriz. SQL Server 2012'den itibaren "WITH RESULTS SETS" ile prosedür çağırma desteği eklenmiştir.

Örnek_8:

```
ALTER PROC sp$UrununSiparisleri(
@urunKod INT
)
AS
    SELECT U.urunKod, U.urunAd, SD.fiyat
    FROM tblUrun U JOIN tblSiparisDetay SD ON SD.urunKod=U.urunKod
    WHERE SD.urunKod=@urunKod
GO
```

Not: Bu sp varsa Alter ile, yoksa CREATE ile çalıştırın.



BÖLÜM SONU ETKİNLİĞİ

1. Herhangi bir tablo üzerinde veri ekleme, silme ve güncelleme için gerekli stored procedürü tanımlayınız.
2. Dışardan parametre olarak ürün adını alan ve o ürünü arayan stored prosedürü yazınız.