STORED PROCEDURE'LER (SAKLI YORDAMLAR)

Eskiden yazılımlar, sadece prosedür denilen kod parçalarından oluşurdu. Her prosedür, belli bir işlevi yerine getirmek için yazılmış kod parçalarıdır.

Mesela, 2 sayı alıp, bunların toplamını hesaplayan bir kod parçasını "toplayıcı" adında bir prosedür içinde paketleyebiliriz.

Bir prosedür, başka bir prosedür içinden çağrılabilir. Bu da sık yapılan işlemler için yazılmış kodların bir defa yazılıp, çok defa kullanılmasını, böylelikle programlamayı kolaylaştırmayı amaçlar.

Birden fazla işlemi, paketlenmiş bir halde, tek bir komut ile çalıştırılması gerektiğinde Stored Procedure'ler kullanılır. Burada işlemlerden kastedilen, Transact SQL ile yapılan her şeydir.

Bir stored procedure oluşturulduktan sonra veritabanı sunucusunda saklanır. Her ihtiyaç duyulduğunda defalarca çağrılabilir.

Stored procedure'ler ağ trafiğini azaltır. Sistem performasını arttırır. Sunucu bazlı yazılımları kolaylaştırır.

Neler yapılabilir ? yapısı nasıldır?
Örnek c++ prosedürü

```
void Hesap (int a, int b)
{
      İşlemler
}
```

Burada Hesap, prosedürün adı,
a ve b ise prosedürün parametresidir.
Çalıştırsak

Hesap(5,7)

- parametreli olabilir.
- Parametresiz olabilir.
- İstendiğinde parametreden veri geri dönebilir.
- İstendiğinde, parametreye varsayılan değer atanabilir.
- Her türlü sql deyimi çalıştırılabilir.
- Prosedür içinde prosedür çalıştırılabilir.
- Sunucu tarafında çalışması istenen uygulamalar yapılabilir.

Genel Oluşturma ifadesi

CREATE : oluşturma
ALTER : değiştirme
DROP : silme

```
CREATE PROCEDURE prosedürAdı
varsaParametreler
AS
BEGIN
```

…. SQL deyimlerini yazıyoruz.
END

Örnek prosedür

```
CREATE PROCEDURE spPersonel
AS
SELECT * FROM Personel
go

--Çalıştırma
EXEC spPersonel
```

Yukarıdaki prosedürü değiştirelim. Sicilno verilen personelin bilgisi bize versin

```
ALTER PROCEDURE spPersonel
@sicilno varchar(20)
AS
BEGIN
    SELECT * FROM Personel WHERE
SicilNo = @sicilno
END
```

```
-- Çalıştırma
EXEC spPersonel '2543'

EXEC spPersonel '0415'
```

Parametreler :

Parametre isimlerinin başında değişkenler gibi @ sembolü vardır. Aynen değişkenler gibi tanımlanır.

@parametre1 vertipi1, @parametre2 veritipi2, ….

Örnek parametre
@ad varchar(20), @soyad varchar(20), @para money

Eğer parametrede varsayılan değer var ise varsayılanlı olarak aşağıdaki gibi yazılır.

@parametre vertipi = varsayılanDeğer

Örnek :
@ad varchar(20)='Kenan'

Prosedürü çalışırırken eğer @ad parametresine değer girmez isek, @ad paramatresinin değeri 'Kenan' dır.

Eğer parametreden veri dönmesi istenirse OUTPUT kelimesi kullanılır.

@parametre veritipi OUTPUT
Örnek:
@Basarili int OUTPUT

@Basarili parametresi ile amacımız, veri göndermek değil, prosedürden bilgi almaktır.

Örnek :

Tablo: Hesaplar

HesapNo	Adi	Soyadi	Bakiye
2053	İbrahim	Öz	12500
6804	Hasan	Bülbül	5000
2543	Neşe	Şen	4150
0415	Ahmet	Uzun	-1000
2749	Ayşe	Doğru	14000
0324	Halil	Yaman	12500

Soru: hesaplar arası para transferi yapan bir prosedür olüşturunuz. Ancak, bakiyenin sıfırın altına düşmesine izin verilmesin.

Çözüm

Kullanacağımız parametreler.

AlacaklıHesapNo,

BorçluHesapNo,

ParaMiktarı

ProsedürAdı : spParaTransfer

```
CREATE PROCEDURE spParaTransfer
@AlacaklıHesapNo char(10),
@BorcluHesapNo char(10),
```

```
@ParaMiktari money
AS
BEGIN
…..  SQL DEYİMLERİ
END
```

Prosedür içindeki sql deyimlerini yazalım.

Borçlu hesabın sıfırın altına düşüp düşmediğinin belirlenmesi. Bunun için IF deyimi içine bir alt sorgu yazacağız. Bu alt sorgu parası eksilecek kişinin bakiyesini verir.

```
IF (select bakiye from hesaplar where
HesapNo = @BorcluHesapNo) >=
@ParaMiktarı
BEGIN
    …. TRANFER ET
END
ELSE
    ..  bakiye yetmedi mesajı ver
------
```

… TRANSFET ET kısmını yazalım.

```
UPDATE Hesaplar SET Bakiye=Bakiye-
@ParaMiktar WHERE HesapNo=
@BorcluHesapNo
Update hesaplar set Bakiye=Bakiye+
@ParaMiktar WHERE
HesapNo=@AlacakliHesapNo
---- Bakiye yetmedi mesajı
PRINT 'Bakiye Yetersiz'
```

Tümünü yeniden yazalım.

```
CREATE PROCEDURE spParaTransfer
@AlacakliHesapNo char(10),
@BorcluHesapNo char(10),
@ParaMiktari money
AS
BEGIN
IF (select bakiye from hesaplar where
HesapNo = @BorcluHesapNo) >=
@ParaMiktari
BEGIN
```

```
UPDATE Hesaplar SET Bakiye=Bakiye-
@ParaMiktari WHERE HesapNo=
@BorcluHesapNo
Update hesaplar set Bakiye=Bakiye+
@ParaMiktari WHERE
HesapNo=@AlacakliHesapNo
END
ELSE
    PRINT 'Bakiye Yetersiz.'
END
go


-- 2053 den 6804 'e 1000  tranfer edelim.
exec spParaTransfer '6804','2053',1000

-- 2053 den 6804 'e 13000 yollayalım
exec spParaTransfer '6804','2053',13000
```