STORED PROCEDURE LER
(Saklı 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 kod parçasını "toplayıcı" adında bir prosedür içinde paketleyebilir.

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, bir tek komut ile çalıştırmamız gerktiğinde Stored Procedureler kullanılır. İşlemlerden kastedilen Transact SQL ile yapılan her şeydir.

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

Network bazlı çalışmalarda ağ trafiğini azaltır, sistem performansını yükseltir.

Örnek bir c++ prosedürü

```
void hesap (int a, int b)
{
  …..
}
```

Hesap : prosedürün adı
a ve b, bu prosedürün parametreleri

Stored Prosedure lerin yapısı

- parametreli olabilir.
- Parametresiz olabilir
- İstendiğinde parametreden veri geri dönebilir.

- İstendiğinde parametreye varsayılan değer atabilir
- Her türlü sql deyimi çalıştırılabilir.

Genel Oluşturma ifadesi

CREATE : oluşturma
ALTER : Yapı değiştirme
DROP : Silme

```
CREATE PROCEDURE prosedürAdı
varsaParametreler
AS
BEGIN
….. sql deyimleri (prosedür içinde
çalışması istenen)
…..
END
```

Örnek Prosedür

```
CREATE PROCEDURE spHesaplarListe
AS
SELECT * FROM Hesaplar
```

GO

Yukarıdaki prosedürü çalıştıralım
EXEC spHesaplarListe

Örnek:
HesapNo parametre olsun, bu hesap nosundaki bilgiyi veren stored procedure'u yazınız.

```
CREATE PROCEDURE spHesap
@HesapNo varchar(10)
AS
  SELECT * FROM Hesaplar WHERE
      HesapNo=@HesapNo
Go
```

Çalıştıralım
EXEC spHesap '0415'

Parametre yazım şekilleri

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

@Parametre1 veritipi1, @parametre2 veritipi2, ….

Örnek
@ad varchar(20), @soyad varchar(20), @para Money

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

@ad varchar(20)=’Kenan’

Yukarıdaki prosedürde varsayılan 0415 olsun.

```
ALTER PROCEDURE spHesap
@HesapNo varchar(20)=’0415’
AS
```

SELECT * from hesaplar where HesapNo=@HesapNo
Go
Neden alter? spHesap veritabanında az önce mevcuttu, biz bunu değiştiriyoruz.

Yukarıdaki prosedürü çalıştıralım

EXEC spHesap
Çıktısı
@HesapNo parametresinin varsayılan değeri 0415 olduğu için, 0415'e ait değeri listeler

EXEC spHesap '6804'
Çıktısı:
@HesapNo değeri 6804 olduğu için, 6804 listelenir.

Eğer parametrede veri geri dönmesi istenirse OUTPUT kelimesi kullanılır.

@Basarim int OUTPUT

@Basarim parametresi ile amacımız, veri göndermek değil veri olmak, prosedürden bilgi öğrenmek içindir.

Tablo : Hesaplar

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

Soru:
Hesaplar arası para transferi yapan bir prosedür oluşturunuz. Ancak transfer sonucu bakiyenin sıfırın altına düşmesine izin vermeyin.
Çözümü:
Kullanacağımız parametreleri tanımlayalım,
AlacaklıHesapNo,

BorcluHesapNo,
ParaMiktari

Prosedür Adı : spParaTransfer olsun.

```
CREATE POCEDURE spParaTranfer
@AlacakliHesapNo varchar(10),
@BorcluHesapNo varchar(10),
@ParaMiktari Money
AS
BEGIN
        …. SQL deyimlerini yazacağım.
END
```

Prosedür içinde çalışacak sql kodunu yazalım.

Borçlu hesabın sıfırın altına düşüp, düşmediğinin kontrolünü yapalım. Bunu için Borçlu hesabın bakiyesini hesaplayacağız. Bu hesabı IF deyimi ile @ParaMiktarı ile karşılaştıracağız. Bulduğumuz değer @ParaMiktar'ından

büyük ise transferi gerçekleştireceğiz, değilse transfer olmadığını bildireceğiz. Bu amaçla IF deyimi içine alt sorgu yazacağım

```
IF (select Bakiye from Hesaplar where
HesapNo = @BorcluHesapNo) >
@ParaMiktari
BEGIN
        … Para tranferini gerçekleştir.
END
ELSE
   … Para transferinin olmadığını bildir.
```

Mavi renkli olan alt sorgudan @BorcluHesapNo numarasının bakiyesini verir. Bu bakiye @ParaMiktarından büyük olduğunda para tranferi gerçekleşir.

Şimdi tümünü tek kod altında toplayalım.

```
CREATE PROCEDURE spParaTransfer
@AlacakliHesapNo varchar(10),
@BorcluHesapNo varchar(10),
@ParaMiktari Money
As
BEGIN
  IF (select Bakiye from hesaplar where
HesapNo=@BorcluHesapNo) >
@ParaMiktari
  BEGIN
      UPDATE Hesaplar SET Bakiye =
Bakiye + @ParaMiktari WHERE
HesapNo = @AlacakliHesapNo
      UPDATE Hesaplar SET Bakiye =
Bakiye - @ParaMiktari WHERE
HesapNo=@BorcluHesapNo
  END
ELSE
      PRINT 'Bakiye Yetersiz. Tranfer
olmadı'

END
```

Örnek:
Yukarıdaki prosedürden tranfer gerçekleştiğinde 1 değerini, gerçekleşmediğinde 0 değerini parametre geri döndüren kodu yazınız.Bu prosedürü çalıştırarak test ediniz.

```
ALTER  PROCEDURE spParaTransfer
@AlacakliHesapNo varchar(10),
@BorcluHesapNo varchar(10),
@ParaMiktari Money,
@Basarili tinyint OUTPUT
As
BEGIN
  IF (select Bakiye from hesaplar where
HesapNo=@BorcluHesapNo) >
@ParaMiktari
  BEGIN
      UPDATE Hesaplar SET Bakiye =
Bakiye + @ParaMiktari WHERE
HesapNo = @AlacakliHesapNo
```

```
        UPDATE Hesaplar SET Bakiye =
Bakiye - @ParaMiktari WHERE
HesapNo=@BorcluHesapNo
        SELECT @Basarili=1
  END
ELSE
        BEGIN
        PRINT 'Bakiye Yetersiz. Tranfer
olmadı'
        SELECT @Basarili=0
        END
END
```

Çalıştıralım

```
DECLARE @Basarili tinyint
EXEC spParaTransfer '0415', '2543',
5000, @Basarili OUTPUT
SELECT @Basarili
```

Çıktısı

```
0
```

EXEC deyimi prosedürün çalışmasını sağlar.

Exec deyimini çalıştırırken parametreler, prosedürdeki sırada olmalıdır. Eğer parametreler isimleri ile yazılırsa, aynı sırada olma koşulu yoktur. Yukarıdaki deyim aşağıdaki gibi de yazılabilir

```
EXEC spParaTranfer
@AlacakliHesapNo='2053',
@BorcluHesapNo='0415',
@ParaMiktari=5000,
@Basarili=@Basarili OUTPUT
```

Örnek
HesapNosuna göre bakiye azaltıp, arttıran bir prosedür yazınız. Bu prosedürde parametreler: HesapNo, Artim ve Islem olsun. Islem değeri 0 ise artsın ,1 ise azalsın. Islemin varsayılan değeri 0 olsun.

Çözüm

```
CREATE PROCEDURE
spBakiyeArttirAzalt @HesapNo
varchar(10), @Artim Money, @Islem
tinyint=0
AS
BEGIN
      IF @Islem=0
            Update hesaplar set
bakiye=bakiye+@Artim where
HesapNo=@HesapNo
      ELSE
            Update hesaplar set bakiye =
bakiye - @Artim where
HesapNo=@HesapNo
END
```

Çalıştıralım. Test için ayrı ayrı çalıştıralım.

```
EXEC spBakiyeArttirAzalt
@HesapNo='2749',@Islem=0,
@Artim=2000
```

```
EXEC spBakiyeArttirAzalt
@HesapNo='2749',@Islem=1,
@Artim=2000

EXEC spBakiyeArttirAzalt
@HesapNo='2749', @Artim=2000
```