STORED PROCEDURE'LER
(Saklı Yordamlar)

Eskiden yazılımlar, prosedür denilen kod parçalarınadn oluşurdu. Her prosedür belli bir işlevi yerine getirmek için yazılmış kod parçalarıdır.
Mesela, 2 sayıyı alıp, bunların toplamını hesaplayan kod parçasını "toplayıcı" adında bir prosedür içinde paketleyebiliriz.

Bir prosedür, başka bir prosedür içinde 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 gerektiğinde Stored Prosedürler kullanılır. İşlemlerden kastedilen Transact SQL ile yapılan her şeydir.

Bir stored prosedure oluşturulduktan sonra veritabanı sunucusunda saklanır. Her ihtiyaç duyulduğunda defalarca çağrılabilir.
Network bazlı çalışmalarda ağ trafiğini azaltır, sistem performansını yükseltir.

Örnek c++ prosedürü

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

Hesap: bu prosedürün adı
a ve b : prosedürün parametreleri

Stored procedure'ün yapısı

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

- İstendiğinde parametreye varsayılan değer atanabilir.
- Prosedür içinde her türlü sql deyimi bulunabilir.

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
```

Bu prosedürü çalıştıralım

```
EXEC spHesaplarListe
```

Örnek:
HesapNo parametre olsun, bu hesapNo'sundaki bilgiyi veren stored prosedürü yazınız.

```
create procedure spHesaplar @HesapNo
varchar(10)
as
SELECT * FROM Hesaplar where
HesapNo=@HesapNo
Go
```

Çalıştıralım.

```
exec spHesaplar '0415'
```

Parametre yazım şekilleri

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

@Parametre1 veritipi1, parametre2 veritipi2, ….
Şeklinde yazılır.

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

gibi yazılır.

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

@ad varchar(20)=’kenan’

Yukarıdaki prosedürü varsayılan değer 0415 olarak yeniden yazınız.

```
ALTER PROCEDURE spHesaplar
@HesapNo varchar(10)='0415'
AS
SELECT * FROM Hesaplar WHERE
HesapNo=@HesapNo
Go
```

Neden alter? Çünkü veritabanında az önce spHesaplar adın prosedür oluşturmuştuk. Bunun üzerine yeni kodu koyuyoruz.

Çalıştıralım

```
exec spHesaplar
```

@HesapNo parametresinin varsayılan değeri 0415 olduğu için, bu varsayılana uygun çıktı gelir.

```
exec spHesaplar '6804'
```

@HesapNo parametresinin değeri 6804 olduğu için, bu değere uygun çıktı gelir.

Eğer parametre veri geri dönmesi istenirse output kelimesi kullanılır

@Basarili tinyint output

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

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 tranferi yapan bir prosedür oluşturunuz. Ancak transfer

sonucu bakiyenin sıfırın altına düşmesine izin vermeyin.
Çözüm:
Kullanacağımız parametreler
Gonderen hesap no, --- @GonderenHesap
Alici hesapno, --- @AliciHesap
Gonderilen para miktari -- @ParaMiktari

Prosedürün adı : spParaTransfer

```
CREATE PROCEDURE spParaTranfer
@GonderenHesap varchar(10),
@AliciHesap varchar(10),
@ParaMiktari Money
AS
BEGIN
    ….. SQL deyimleri yazacağız. (Para tranferi yapan)
END
```

Prosedür içinde çalışacak SQL deyimin yazalım.

Gönderen hesabın sıfırın altına düşüp, düşmediğinin kontrolünü yapalım. Bunun için gönderen hesabın bakiyesini bulacağız. Bu bakiyeyi IF deyimi ile @ParaMiktari değeri ile karşılaştıracağız. Buluduğumuz değer @ParaMiktarından büyük ise tranferi gerçekleştireceğiz. Küçük ise tranfer olmadığını bildireceğiz.

```
IF (Select bakiye from Hesaplar where
HesapNo = @GonderenHesap) >
@ParaMiktari
BEGIN
        . Parayı gönder
END
ELSE
    Para gönderilmedi bilgisi ver
```

Para gönder kodunu yazalım.
Gönderen hesaptan @ParaMiktari kadar eksilme olacak

```
UPDATE Hesaplar SET Bakiye= Bakiye -
@ParaMiktari where
HesapNo=@GonderenHesap
```

Alıcı hesabın parası @ParaMiktari kadar artacaktır

```
UPDATE Hesaplar SET Bakiye=Bakiye +
@ParaMiktari WHERE
HesapNo=@AliciHesap
```

Para tranferi olmadığını bildirelim

```
PRINT 'Bakiye yetersiz. Transfer olmadı'
```

Toplu olarak kodlayalım

```
CREATE PROCEDURE spParaTransfer
@GonderenHesap varchar(10),
@AliciHesap varchar(10),
@ParaMiktari Money
AS
BEGIN
    -- yeterli bakiye var mı ?
```

```
    IF (select bakiye from Hesaplar where
HesapNo=@GonderenHesap) >
@ParaMiktari
    BEGIN
        -- Para gönder
        Update Hesaplar set bakiye=bakiye
- @ParaMiktari where HesapNo=
@GonderenHesap
        Update Hesaplar set bakiye=bakiye
+ @ParaMiktari where
HesapNo=@AliciHesap
    END
    ELSE
        --Mesaj ver
        Print 'Bakiye Yetesiz. '
END
```

Çalıştıralım.

```
EXEC spParaTransfer '6804', '2053', 3000
```

6804 nolu hesaptan, 2053 nolu hesaba 3000 para gönderildi.

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

```
ALTER PROCEDURE spParaTransfer
@GonderenHesap varchar(10),
@AliciHesap varchar(10),
@ParaMiktari Money,
@Basarili tinyint OUTPUT
AS
BEGIN
    -- yeterli bakiye var mı ?
    IF (select bakiye from Hesaplar where
HesapNo=@GonderenHesap) >
@ParaMiktari
    BEGIN
        -- Para gönder
```

```
        Update Hesaplar set bakiye=bakiye
- @ParaMiktari where HesapNo=
@GonderenHesap
        Update Hesaplar set bakiye=bakiye
+ @ParaMiktari where
HesapNo=@AliciHesap
    SELECT  @Basarili=1
    END
    ELSE
        SELECT @Basarili=0

END
```

Çalıştıralım

```
DECLARE @Basarili tinyint
EXEC spParaTransfer '6804', '2053', 500,
@Basarili OUTPUT
SELECT @Basarili
```

Parametre değerleri prosedürdeki sırada olması gerekir. Eğer parametre ismiyle birlikte yazarsak sıralı olması gerekmez.

```
DECLARE @Basarili tinyint
EXEC spParaTransfer
@GonderenHesap='6804',
@AliciHesap='2053', @ParaMiktar=500,
@Basarili=@Basarili OUTPUT
SELECT @Basarili
```

Yukarıdaki gibi yazdığımızda sıralama önemli değil.