Veritabanı Yönetim Sistemleri

(View, Saklı Yordamlar(SP), Trigger)

Konular

- ✓ Saklı Yordamlar (Stored Procedures)
- ✓ Cursor
- ✓ Trigger
- ✓ Kaynaklar

Saklı Yordamlar (Stored Procedures)

- ✓ Veritabanı kataloğunda saklanan SQL ifadeleridir. Saklı yordamlar; uygulama yazılımları, trigger yada başka bir saklı yordam ile çağrılabilirler.
- ✓ Uygulamanın başarımını iyileştirir. Saklı Yordamlar bir defa oluşturulduktan sonra derlenerek veritabanı kataloğunda saklanır. Her çağrıldığında SQL Motoru tarafından derlenmek zorunda olan SQL ifadelerine göre çok daha hızlıdır.
- ✓ Uygulama ile veritabanı sunucusu arasındaki trafiği azaltır. (Uzun SQL ifadeleri yerine saklı yordamın adını ve parametrelerini göndermek yeterlidir)
- ✓ Defalarca kullanılabilir (reusable). Tasarım ve uygulama geliştirme sürecini hızlandırır.
- ✓ Güvenliğin sağlanması açısından çok kullanışlıdır. Veritabanı yöneticisi, saklı yordamlara hangi uygulamalar tarafından erişileceğini, tabloların güvenlik düzeyleriyle uğraşmadan, kolayca belirleyebilir.
- ✓ SP ile program yazmak, değiştirmek ve hata bulmak zordur.
- ✓ DBMS veri depolama ve listeleme işlerine ek olarak farklı işler yapmak zorunda da kalacağı için bellek kullanımı ve işlem zamanı açısından olumsuz sonuçlara neden olabilir.(Saklı yordamların yapacağı işler uygulama yazılımlarına da yaptırılabilir)

Saklı Yordamlar (Koşullu İfadeler)

```
USE AdventureWorks2012;
GO
DECLARE @AvgWeight decimal(8,2), @BikeCount int
IF
(SELECT COUNT(*) FROM Production.Product WHERE Name LIKE 'Touring-3000%' ) :
BEGIN
   SET @BikeCount =
        (SELECT COUNT(*)
         FROM Production.Product
         WHERE Name LIKE 'Touring-3000%');
   SET @AvgWeight =
        (SELECT AVG(Weight)
         FROM Production.Product
         WHERE Name LIKE 'Touring-3000%');
   PRINT 'There are ' + CAST(@BikeCount AS varchar(3)) + ' Touring-3000 bik
   PRINT 'The average weight of the top 5 Touring-3000 bikes is ' + CAST(@A
END
ELSE
BEGIN
SET @AvgWeight =
        (SELECT AVG(Weight)
         FROM Production.Product
         WHERE Name LIKE 'Touring
   PRINT 'Average weight of the T
END ;
GO
```

Declare: Bir değişken tanımlamak için kullanılır. Önce *Declare* daha sonra @ işareti ile birlikte *değişken adı* ve son olarak *değişkenin türü.*
Set: Declare ile tanımladığımız değişkenin içeriğini değiştirmek için kullanırız.
While: *While* koşul sağlandığı sürece kullanılacak bir döngü yapısıdır. IF te olduğu gibi *Begin-End* blokları arasında tekrarlı yapı yer alır.

Saklı Yordamlar (Akış Denetimi)

```
IF
(SELECT COUNT(*) FROM Production.Product WHERE Name LIKE 'Touring-3000%' ) > 5
PRINT 'There are more than 5 Touring-3000 bicycles.'
ELSE PRINT 'There are 5 or less Touring-3000 bicycles.' ;
GO
```

```
DECLARE @Number int;
SET @Number = 50;
IF @Number > 100
   PRINT 'The number is large.';
ELSE
   BEGIN
      IF @Number < 10
      PRINT 'The number is small.';
   ELSE
      PRINT 'The number is medium.';
   END ;
GO
```

```
WAITFOR TIME '23:00'
WAIT FOR DELAY ''00:01:00''
```

```
RETURN @return_code
```

Saklı Yordamlar (Akış Denetimi)

```
DECLARE @intFlag INT
SET @intFlag = 1
WHILE (@intFlag <=5)
BEGIN
PRINT @intFlag
SET @intFlag = @intFlag + 1
END
GO
```

```
Use NorthWind5;
GO -- Go ya kadar olan ifadeler VT sunucuya gönderilir.
--Dolayısıyla, yukarıda bir değişken var ise aşağıda kullanılamaz...
declare @CustId nchar(5), @companyName nvarchar(40)
declare @RowNum int
select top 1 @CustId=CustomerID, @companyName=CompanyName from Northwind.dbo.Customers
set @RowNum = 0
WHILE @RowNum < 5
BEGIN
  set @RowNum = @RowNum + 1
  print cast(@RowNum as char(1)) + ' ' + @CustId + ' '+ @companyName
  select top 1 @CustId=CustomerID, @companyName=CompanyName from Northwind.dbo.Customers
               where CustomerId > @CustID
END
```

Messages

```
1 ALFKI Alfreds Futterkiste
2 ANATR Ana Trujillo Emparedados y helados
3 ANTON Antonio Moreno Taquería
4 AROUT Around the Horn
5 BERGS Berglunds snabbköp
```

Saklı Yordamlar (Akış Denetimi)

```
USE AdventureWorks2012;
GO
SELECT   ProductNumber, Name, "Price Range" =
      CASE
         WHEN ListPrice =  0 THEN 'Mfg item - not for resale'
         WHEN ListPrice < 50 THEN 'Under $50'
         WHEN ListPrice >= 50 and ListPrice < 250 THEN 'Under $250'
         WHEN ListPrice >= 250 and ListPrice < 1000 THEN 'Under $1000'
         ELSE 'Over $1000'
      END
FROM Production.Product
ORDER BY ProductNumber ;
GO
```

Saklı Yordamlar (Stored Procedures)

```
CREATE PROCEDURE dbo.[Musteriler]
AS
BEGIN
    SET NOCOUNT ON; --Etkilenen Satır sayısı sonuç ile beraber döndürülmez
    SELECT CustomerID, CompanyName, ContactName, ContactTitle,
    Address, City, Region, PostalCode, Country, Phone, Fax
    FROM dbo.Customers
END
-- ---------------------------------
Exec Musteriler -- Saklı Yordamı çalıştırmak için kullanılır
```

Results | Messages

	CustomerID	CompanyName	ContactName	ContactTitle	Address
1	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57
2	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2

Saklı Yordamlar (Stored Procedures)

```
ALTER PROCEDURE dbo.[Musteriler]
 AS
BEGIN
     SET NOCOUNT ON; --Etkilenen Satır sayısı sonuç ile beraber döndürülmez
     SELECT CustomerID, CompanyName, ContactName, ContactTitle,
     Address, City, Region, PostalCode, Country, Phone, Fax
     FROM dbo.Customers
END
```

```
DROP PROCEDURE [dbo].[Musteriler]
```

Saklı Yordamlar (Stored Procedures)

```
CREATE PROCEDURE dbo.spMusteriEkle
(
    @CustomerID nchar(5),
    @CompanyName nvarchar(40),
    @ContactName nvarchar(30),
    @ContactTitle nvarchar(30),
    @Address nvarchar(60),
    @City nvarchar(15),
    @Region nvarchar(15),
    @PostalCode nvarchar(10),
    @Country nvarchar(15),
    @Phone nvarchar(24),
    @Fax nvarchar(24)
)
AS
BEGIN
    SET NOCOUNT OFF;
    INSERT INTO [dbo].[Customers] ([CustomerID], [CompanyName], [ContactName],
        [ContactTitle], [Address], [City], [Region], [PostalCode], [Country], [Phone], [Fax])
    VALUES (@CustomerID, @CompanyName, @ContactName, @ContactTitle, @Address, @City,
        @Region, @PostalCode, @Country, @Phone, @Fax);

    SELECT CustomerID, CompanyName, ContactName, ContactTitle, Address,
        City, Region, PostalCode, Country, Phone, Fax
    FROM Customers WHERE (CustomerID = @CustomerID)
END
```

```
exec spMusteriEkle a,a,a,a,a,a,a,a,a,a,a
```

Saklı Yordamlar (Stored Procedures)

```
CREATE PROCEDURE dbo.UpdateCustomers
(
    @CustomerID nchar(5),
    @CompanyName nvarchar(40),
    @ContactName nvarchar(30),
    @ContactTitle nvarchar(30),
    @Address nvarchar(60),
    @City nvarchar(15),
    @Region nvarchar(15),
    @PostalCode nvarchar(10),
    @Country nvarchar(15),
    @Phone nvarchar(24),
    @Fax nvarchar(24),
    @Original_CustomerID nchar(5)
)
AS
    SET NOCOUNT OFF;
UPDATE [dbo].[Customers] SET [CustomerID] = @CustomerID, [CompanyName] = @CompanyName, [Conta

SELECT CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode,
```

Saklı Yordamlar (Stored Procedures)

```
CREATE PROCEDURE dbo.DeleteCustomers
(
    @Original_CustomerID nchar(5)
)
AS
    SET NOCOUNT OFF;
DELETE FROM [dbo].[Customers] WHERE (([CustomerID] = @Original_CustomerID))
```

Saklı Yordamlar (Stored Procedures)

```
CREATE PROCEDURE [dbo].[spAramaKaydiGoruntule]
AS

Begin
    SELECT      dbo.Kisiler.ad, dbo.Kisiler.soyad, dbo.TelefonNumaralari.telefonNo,
                dbo.Operator.operatorAdi, dbo.AramaKaydi.zaman, dbo.AramaKaydıTuru.turAdi
    FROM         dbo.Kisiler INNER JOIN
                      dbo.TelefonNumaralari ON dbo.Kisiler.kisiNo = dbo.TelefonNumaralari.kisiNo INNER JOIN
                      dbo.AramaKaydi ON dbo.TelefonNumaralari.telefonNumaralari =
                      dbo.AramaKaydi.telefonNumaralariNo INNER JOIN
                      dbo.Operator ON dbo.TelefonNumaralari.operator = dbo.Operator.operatorNo INNER JOIN
                      dbo.AramaKaydıTuru ON dbo.AramaKaydi.aramaTuru = dbo.AramaKaydıTuru.turNo
End
```

```
Exec spAramaKaydiGoruntule
```

Results | Messages

	ad	soyad	telefonNo	operatorAdi	zaman	turAdi
1	Ahmet	Tansel	2643032234	Telekom	2012-01-01 00:00:00.000	Giden Arama

Saklı Yordamlar (Parametre Gönderme, Değer Döndürme)

```
CREATE PROCEDURE [dbo].[spKisiSayisi]
 @ad varchar(40), -- default değer yazılırsa (='xy') çağrılırken parametre
                  -- default değer parametre olarak kabul edilir.
 --birden fazla parametre var ise virgül ile yazılabilir
 @kisiSayisi int Output
AS

Begin
    SELECT      @kisiSayisi= COUNT(*)
    FROM          dbo.Kisiler
    where ad Like @ad+'%'
End
GO
```

```
declare @sayi int
exec spKisiSayisi 'A', @sayi output
print @sayi
```

Cursor

```
DECLARE myCursor CURSOR
FOR SELECT  TOP(5) ContactName FROM Customers
DECLARE @RowNo int,@ContactName nvarchar(30)
SET @RowNo=0
OPEN myCursor
FETCH NEXT FROM myCursor INTO @ContactName
WHILE @@FETCH_STATUS=0
  BEGIN

        SET @RowNo=@RowNo+1

        PRINT + LEFT(CAST(@rowNo as varchar) + '       ',6)+'  '+ @ContactName
        FETCH NEXT FROM myCursor INTO @ContactName
  END
CLOSE myCursor
DEALLOCATE myCursor
```

Messages

```
1       Maria Anders
2       Ana Trujillo
3       Antonio Moreno
4       Thomas Hardy
5       Christina Berglund
```

Cursor

```
DECLARE @VAR_ID INT
DECLARE @VAR_ADI VARCHAR(50)

DECLARE CRS_PERSONEL CURSOR FOR
SELECT ID,ADI FROM PERSONEL ORDER BY YASI

OPEN CRS_PERSONEL

FETCH NEXT FROM CRS_PERSONEL INTO @ID,@ADI

WHILE @@FETCH_STATUS =0
        BEGIN
                SELECT A.ADI, B.ACIKLAMA FROM AILE_BILGILER A
INNER JOIN YAKINLIK_DERECELERI B ON B.ID = A.YAKINLIK_ID
WHERE A.YAKIN_ID = @ID


                FETCH NEXT FROM CRS_PERSONEL INTO @ID,@ADI

        END

CLOSE CRS_PERSONEL

DEALLOCATE CRS PERSONEL
```

Soru: Arama kaydı en çok olan 5 telefon numarasının sahibini sık kullanılanlar gurubuna dahil eden SP yi Cursor kullanarak gerçekleştiriniz

http://www.godoro.com/Divisions/Ehil/Mecmua/Magazines/Articles/txt/html/article_CursorSQLServer.html adresinden alınmıştır

Trigger

- ✓ SQL trigger is an SQL statements or a set of SQL statements which is stored to be activated or fired when an event associating with a database table occurs. The event can be any event including INSERT, UPDATE and DELETE.

- ✓ SQL Trigger provides an alternative way to check integrity.
- ✓ SQL trigger can catch the errors in business logic in the database level.
- ✓ SQL trigger provides an alternative way to run scheduled tasks. With SQL trigger, you don't have to wait to run the scheduled tasks. You can handle those tasks before or after changes being made to database tables.
- ✓ SQL trigger is very useful when you use it to audit the changes of data in a database table.

- ✓ SQL Triggers executes invisibly from client-application which connects to the database server so it is difficult to figure out what happen underlying database layer.
- ✓ SQL Triggers run every updates made to the table therefore it adds workload to the database and cause system runs slower.

Trigger

```
CREATE TRIGGER triggerdemo
ON Customers
FOR delete

AS
      insert into CustomersYed
      SELECT * FROM deleted

GO
```

```
CREATE TRIGGER trSilinenMusteri ON Customer
FOR delete

AS
    insert into CustomerYed
        SELECT *  FROM deleted

GO
```

Silinen müşteriyi başka bir tabloya alan trigger

Trigger

```
create trigger stokGuncelle on satis -bu satırda trigger adını yazıyoruz
ve tetiklemenin kaynağı olan tabloyu yazıyoruz.
for insert - trigger'ın hangi komut için yazıldığını gösterir.
as
begin
   declare @satisAdedi int, @urunid int - iki adet değişken
belirliyoruz.
   select @satisAdedi=satisadedi, @urunid=urunid from inserted
-yaptığımız insert ipşeminde satisadedi ve urunid değerlerini alıp
bunları daha önce belirlediğimiz değişkenlere atıyoruz.
   update urunler set stokadedi=stokadedi-@satisAdedi where
urunid=@urunid -şimdi aldığımız değişkenleri başka bir tabloda
değişiklik yapmak için kullanıyoruz.
end
```

Trigger

```
create trigger duzelt on satis
for update- update işlemi ile tetiklemenin başlatıldığını gösterir.
as
begin
      declare @yenisatisadedi int, @satisadedi int, @urunid int
      select @satisAdedi=satisadedi, @urunid=urunid from deleted -silme
işleminde silinen satırdaki satisadedi ve urunid değerlerini alıyoruz.
      select @yenisatisadedi=satisadedi , @urunid=urunid from inserted
-ekleme işleminde eklenen satisadedi ve urunid değerlerini alıyoruz.
      if(@yenisatisadedi<@satisadedi) -eğer eklenen satisadedi değeri
silinen satisadedi değerinden küçük ise işlem yapar. yani güncelleme
yaptığımız değer eski değerden az ise.
      begin
            update urunler set stokadedi=stokadedi-(@satisadedi-
@yenisatisadedi) where urunid=@urunid - ürünlerdeki stokadedini
güncelleme yaptığımızda satisadedi değerindeki değişim kadar azaltır.
      end
      else
      begin
            update urunler set stokadedi=stokadedi+(@yenisatisadedi-
@satisadedi)
            where urunid=@urunid- ürünlerdeki stokadedi'ni güncelleme
yaptığımızda satisadedi değerindeki değişim kadar artırır.
      end
end
```

Trigger

```
DELIMITER $$

DROP TRIGGER IF EXISTS ortalamaHesapla$$
CREATE TRIGGER ortalamaHesapla
BEFORE UPDATE ON OgrenciKayit FOR EACH ROW

 BEGIN

    IF (NEW.not2 <> OLD.not2) THEN
        SET NEW.ortalama=(NEW.not1+NEW.not2)/2;

    END IF;
 END$$
DELIMITER ;
```

```
DELIMITER $$

USE `dershaneyonetimsistemi`$$

DROP TRIGGER /*!50032 IF EXISTS */ `degisikligiKaydet`$$

CREATE TRIGGER `degisikligiKaydet`
BEFORE UPDATE ON `ogrenciler` FOR EACH ROW
BEGIN
   IF (NEW.aktif = 0) THEN
        INSERT INTO mezunlar
        VALUES (OLD.ogrenciNo,OLD.adi,OLD.soyadi);
   END IF;
END;
$$
```

Kaynaklar

- ✓ http://msdn.microsoft.com