**SQL**

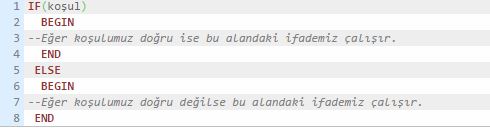
T-SQL komutlarında bazen bir ifadeyi bir koşula dayandırarak çalıştırma ihtiyacı duyabiliriz. Bu durumlarda karar ifadeleri ve döngüler yardımımıza koşar.

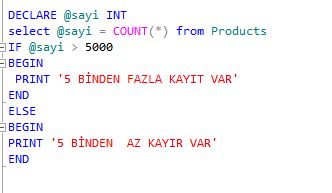
Karar ifadeleri ve döngüler yardımıyla belli bir ifadeyi belli bir sayıda veya belli bir koşul gerçekleşinceye kadar çalıştırabiliriz. Peki, bunlar nelerdir inceleyelim.

**IF-ELSE KULLANIMI**

IF-Else koşul ifadelerimiz çoğu yazılım dilinde kullanıldığı gibi burada da aynı mantıkta çalışır.

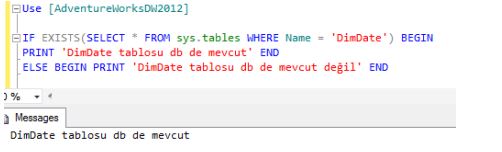
**IF** komutunu izleyen boolen ifade **TRUE** olursa takip eden ifadeler, **FALSE** olursa **ELSE** komutundan sonra gelen ifadeleri gerçekleştirecektir.





**IF EXISTS (Eğitmen Notu )**

Bu komut ile de bir kaydın var olup olmadığını sorgularız. Daha çok database de bir objenin var olup olmadığını sorgulamak istediğimiz zaman kullandığımız bir komuttur.



**WHILE**

 SQL deyimlerinin döngü içinde yinelenmesini sağlar. WHILE ile belirtilen döngü koşulu yerine getirildiği sürece deyimler yerine getirilir.

WHILE ifadesindeki şartı doğrulayacak şekilde bir çevrim oluşturulur; "Break" ve "Continue" ifadeleri ile çevrim kontrol edilir.

While döngüsünün yapısı

|  |
| --- |
| WHILE Boolean\_ifadesi      {sql\_ifadesi | ifade\_blogu}      [BREAK]      {sql\_ifadesi | ifade\_blogu}      [CONTINUE] |

**Açıklamalar:**

Boolean\_ifadesi:

            Doğru veya yanlış değer döndüren ifade. Boolean\_ifadesi bir select sorgusu içerir, select sorgusu parantez içinde yazılmalıdır.

{SQL ifadesi | ifade bloğu}

            BEGIN ve END blokları arasına yazılmış bir T\_SQL ifadesi

BREAK

            While döngüsünü istediği zaman sonlandırmak için kullanılır. END cümlesinden sonra yazılan ifadelere geçer.

CONTINUE

            While döngüsünün başına gider, CONTINUE kelimesinden sonra yazılan ifadeler çalışmaz.

###### ÖRNEK:

###### Tüm ürünlerin fiyat ortalamalarını kontrol edelim eğer ellinin altında ise ürün fiyatını ikiye katlayalım.

###### Tüm ürünlerin fiyatlarının ortalaması ellinin üstüne çıkana kadar döngümüz devam etsin. While döngüsü içindeki kısımda ürün fiyatını ikiye katladıktan sonra tabloda bulunan en pahalı ürünü bulup 50 den büyük mü diye bakalım eğer büyükse döngüyü BREAK ile sonlandıralım değil ise CONTINUE ile döngüye devam edelim.

###### 

**STORED PROCEDURE (SAKLI YORDAM)**

Stored prosedüre dilimizde “saklı alt yordam” veya “saklı işlem grubu” olarak da ifade edilmektedir. Stored procedure’lerin en önemli özellikleri data base server’ın içinde saklanmalarıdır.

Derlenmesi için başlangıçta çalışır ve daha sonraki kullanımlarda derlenmez.

Buda bize artı zaman kazandırır  ve  iyi bir performans sunar. Stored prosedüreler C,C#, Java ya da başka programlama dillerindeki fonksiyonlar gibi parametreler içermektedir.

Bir özelliği daha ise  oluşturulacak bir procedure’ün içinde declare, set, if ,try catch gibi deyimlerinde kullanılmasıdır.

\* İçlerinde tanımlı fonksiyonlar kullanılabilir fakat proclar ilgili fonksiyon içerisinde kullanılmazlar.

-- Niçin SP Kullanmalıyız?

Sql Server ile veri tabanı üzerinde yapılan sorgular (select), veri girişleri (insert), veriler üzerinde yapılan güncellemeler (update) ve verilerin veri tabanından kaldırılması (delete) gibi işlemleri her seferinde ilgili komutu yazarak işleme koymak Sql Server üzerinde yük oluşturur.

Her yazılan sorgu komutu için Sql Server komutu yazım kuralları açısından (syntax) denetler ve sonraki aşamada ilgili tabloların ve alanların var olup olmadığının kontrolü yapılır ve son olarak en kısa ve en hızlı sorgunun nasıl yapılacağı belirlendikten sonra ilgili Sql scripti derlenir ve komut çalıştırılır.

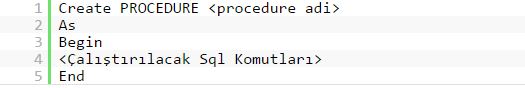
İşte biz Sql Server’a bu kadar işi her seferinde tekrarlatmamak için “Stored Procedure” (saklı yordam) adını verdiğimiz yapılardan faydalanırız. Stored Procedure (kısaca sp veya procedure) kullanılarak komutun derlenmesine kadar olan aşamaları bir kez tamamlanır ve procedure her çağırıldığında sadece ilgili komutun çalıştırılması sağlanır. Böylece Sql Server üzerinde daha performanslı sorgular ve komutlar çalıştırılmış olur.

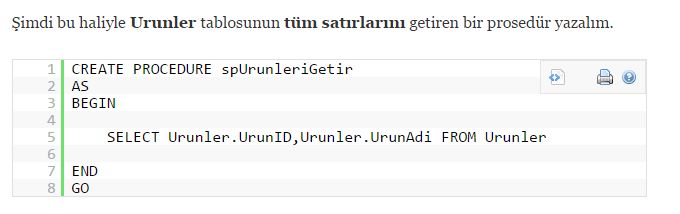
Stored Procedure’ ler hem sorgu komutlarında hem de insert, update ve delete işlemlerinde kullanılabilir. C# taki kullanılan metot (yordam) kavramına benzerler. Parametre alabilirler ve geriye değer döndürebilirler. **Geriye değer döndürme işlemi “output” anahtar kelimesi ile yapılır.**

Procedure’ lere isim verilirken genellikle “sp” ön takısı verilerek isimlendirilirler. Ayrıca procedurler de metotlar gibi iç içe çağrılabilirler.

Örneklerle Stored Procedure’n mantığı ve işlevi daha iyi oturacaktır.

--Parametre almayan prosedur oluştururken alttaki yapı kullanılır.





* Parametreli alt yordamın  söz dizim (syntax) şekli:

CREATE PROCEDURE [Prosedüre İsmi]

(

– -girilecek parametre değerleri buraya yazılacak

@prm1,

@prm2

)

WITH  { RECOMPILE **|**ENCRYPTION } (eğitmen notu ileriki konularda anlatılacak.)

AS

BEGIN

         – -BEGIN END Kullanılması zorunlu değildir.

END

—————————————————–

* With ile beraberinde yazılan ifadeler  { RECOMPILE **|**ENCRYPTION }
  + RECOMPILE  ifadesi Stored prosedüre her çalıştırmada (execute) yeniden derlenecek anlamına gelir.**(Eğitmen notu)**
  + ENCRYPTION  ifadesi  yazılan kodun şifrelenmesidir. Şifrelenmiş kodu sadece o prosedürü yaratan ve system admin olan görebilir.

 Alt yordamın söz dizimini de öğrendikten sonra sık kullanılan veri ekleme prosedürünü yazalım:

Şirkete yeni bir çalışan alındığında, bu çalışanı rahatlıkla eklemek istiyoruz**.**

**(bu örnek northwind üzerinde yapılmadı northwind db deki örnekleri en sona ekliyorum burada konuyu anlatıyorum arkadaşlar.)**

CREATE PROC SP\_CalisanEkle

(

@CalisanID nchar(5),

- -Parametreli oluştururken başına @İşareti

Koyulur ve bitişik olarak hangi işleve yarayacaksa onun

Hatırlatacak isim girilir daha sonra veri tipi yazılarak

Bir sonraki parametreye geçilir.

@CalisanIsim nvarchar(40),

@CalisanSoyisim nvarchar(40),

@CalisanTelefon nvarchar(10),

@CalisanAdres nvarchar(100),

@CalisanNotlari nvarchar(100),

@CalisanDugumTarihi DATETIME

)

WITH ENCRYPTION

AS

IF EXISTS(SELECT \* FROM dbo.CalisanlarinTablosu WHERE CalisanID=@CalisanID)

BEGIN

PRINT 'Sistemde id numarası mevcuttur!'

END

ELSE

BEGIN

INSERT INTO dbo. CalisanlarinTablosu VALUES (@CalisanID,@CalisanIsim,

@CalisanSoyisim,@CalisanTelefon,@CalisanAdres,@CalisanNotlari,

@CalisanDogumTarihi )

END

-------------------------------------------------------

SP\_CalisanEkle  prosedürü If exist() komutunu ve begin end işlevlerini görebilmek için oluşturulmuştur.

IF EXIST() , bize exist() bölümünde oluşturulan kod içinde bir  değer varsa ” True ” olur ve ilk “begin end” bloğuna girip sistemde aynı çalışan numarasına ait başkasının olduğunu söyler.

Eğer sistemde aynı id numarasına ait bir çalışan bulunmadıysa, yeni çalışan else bloğuna girip ekleniyor.

\*\*\*Eğer tek bir satırdan oluşuyorsa kodunuz **Begin End** bloğunu yazmak zorunda değilsiniz.

\*\*\***BEGIN END** bloğu programlama dillerindeki süslü parantezlerdir (scope) { }.

\*Stored Prosedürleri çalıştırmak için oluşturtulan metot adının önüne **EXEC**  ya da **EXECUTE** komutları yazılabilir.

Örneğin:

EXEC SP\_CalisanEkle ‘OSM99’,‘Osman’,‘Oztürk’,‘05541231212’,‘134 sokak daire:5 Mecidiyeköy / İstanbul’,‘İsinde basarılıdır’,‘1980-02-02 ‘

EXECUTE SP\_CalisanEkle ‘OSM99’,‘Osman’,’Oztürk’,‘05541231212’,‘134 sokak daire:5 Mecidiyeköy / İstanbul’,‘İsinde basarılıdır’,‘1980-02-02’

\*\*\*Alt yordamı çalıştırmak için bir seçeneğiniz daha var o da exec ya da execute komutları yazmadan sadece prosedürünüzün adını yazıp gereken parametreleri doldurmaktır.

**SP\_CalisanEkle** ‘OSM99’,‘Osman’,‘Oztürk’,‘05541231212’,‘134 sokak daire:5 Mecidiyeköy / İstanbul’,‘İsinde basarılıdır’,’1980-02-02′

* Stored Prosedürlerde önceden hazırlanmış olan alt yordamın üzerinde değişikliklerde yapılabilir. Değişiklikleri kaydetmek için ALTER komutunu kullanıyoruz.

Oluşturduğumuz SP\_CalisanEkle procedure’ne ek olarak çalışanların hangi program dillerini bildiklerini de ekleyelim ayrıca eğer yeni çalışanın yaşı 30’dan büyükse  firmadaki projelerin denetimine katılacakların listesine de (tabloya) eklensin.

Proje denetmen tablosunda 4 özellikli bir tablodur. Sonuç olarak burada istenen ek programlama dili bilgisi, bizim esas tablomuzda da bu değişikliğin yapılmış olduğunu gösterir ve buna göre tekrar alt yordamın düzeltilmesi istenir.

Bunun için yapılan değişiklikler aşağıdaki gibidir.

**ALTER** PROC SP\_CalisanEkle

(

@CalisanID nchar(5),

@CalisanIsim nvarchar(40),

@CalisanSoyisim nvarchar(40),

@CalisanTelefon nvarchar(10),

@CalisanAdres nvarchar(100),

@CalisanNotlari nvarchar(100),

@CalisanDugumTarihi DATETIME,

@CalisanProgDilleri nvarchar(100)

)

WITH ENCRYPTION

AS

IF EXISTS(SELECT \* FROM dbo. CalisanlarinTablosu WHERE CalisanID=@CalisanID)

BEGIN

PRINT 'Sistemde id numarası mevcuttur!'

END

ELSE

BEGIN

IF (30 <= DATEPART(yyyy,GETDATE())-DATEPART(yyyy,@CalisanDogumTarihi))

BEGIN

INSERT INTO dbo.ProDenetmenTablosu VALUES

(@CalisanID,@CalisanIsim,@CalisanSoyisim,@CalisanTelefon)

INSERT INTO dbo.CalisanlarinTablosu VALUES (@CalisanID,@CalisanIsim,

@CalisanSoyisim,@CalisanTelefon,@CalisanAdres,@CalisanNotlari,

@CalisanDogumTarihi,@CalisanProgDilleri )

END

ELSE

BEGIN

INSERT INTO dbo.CalisanlarinTablosu VALUES (@CalisanID,@CalisanIsim,

@CalisanSoyisim,@CalisanTelefon,@CalisanAdres,@CalisanNotlari,

@CalisanDogumTarihi,@CalisanProgDilleri )

END

END

* Son olaraktan oluşturulan stored prosedüre silmek için değişiklik yaparken kullanılan komutun yerine **DROP** komutu konulup, sistemdeki prosedürü ortadan kaldırabilirsiniz.
* Şunu unutmamak gerekir ki oluşturulan stored prosedürlerin encryption özelliği var ise sadece sistem admin ve o prosedüre yapan kişi silebilir.**(Eğitmen Notu)**

**DROP** PROC SP\_CalisanEkle

**Northwind örnekleri**

--Kargolara kayıt ekleyen proc yazınız.

create proc KayıtEkle

(

@name nvarchar(30),

)

as

insert into Shippers (CompanyName) values (@name)

--Düzenlemek için

\*Telefon numarasını da eklesin istiyorum.

alter proc KayıtEkle

(

@name nvarchar(30)

@phone char(20)

)

as

insert into Shippers (CompanyName,Phone) values (@name,@phone)

**Çalıştırıyoruz sp yi;**

exec KayıtEkle 'Deneme','0900 00 00'

--silmek

drop proc KayıtEkle

--1-Dısarıdan girilen kargo firması tarafından tasınmış ve yine dışarıdan girilen kargo ödemesi değer aralığındaki siparişleri proc ile listeleyiniz.

Create proc KargoFirmasınaUrun

(

@name nvarchar(50),

@minfreight decimal,

@maxfreight decimal

)

as

begin

select o.OrderID,s.CompanyName from Orders o

join Shippers s on s.ShipperID=o.ShipVia

where s.CompanyName like '%'+@name+'%' and o.Freight between @minfreight and @maxfreight

end

exec KargoFirmasınaUrun 'Express',10,100

--Dışarıdan girilen değer kadar ürüne zam yapan proc yazınız.

create proc UrunFiyatGuncelle

(

--işlem dışarıdan zam gelirse zam yapsın indirim gelirse fiyattan çıkarsın

@islem nvarchar(10),

@zam decimal,

@proid int

)

as

if(@islem='zam')

begin

update Products set UnitPrice+=@zam where ProductID=@proid

end

Else if(@islem='indirim')

begin

update Products set UnitPrice-=@zam where ProductID=@proid

end

exec UrunFiyatGuncelle 'zam',100,2

-- InsertProduct SP'ni yazalım.

Create Proc InsertProduct

(

@ProductName nvarchar(40),

@SupplierID int,

@CategoryID int,

@QuantityPerUnit nvarchar(20),

@UnitPrice money,

@UnitsInStock smallint,

@UnitsOnOrder smallint,

@ReorderLevel smallint,

@Discontinued bit

)

AS

Insert Into Products (ProductName, SupplierID, CategoryID, QuantityPerUnit, UnitPrice, UnitsInStock, UnitsOnOrder, ReOrderLevel, Discontinued)

Values (@ProductName, @SupplierID, @CategoryID, @QuantityPerUnit, @UnitPrice, @UnitsInStock, @UnitsOnOrder, @ReOrderLevel, @Discontinued)

**//Procedurler arasında GetCategories isimli proc un olup olmadığını kontrol ediyoruz varsa siliyoruz ve yeniden oluşturuyoruz.**

IF exists(Select Name From sys.procedures Where Name = 'GetCategories')

BEGIN

Drop Proc GetCategories

END

go

Create Proc GetCategories

as

Select \* From Categories Order By CategoryName

--Örnekk—

--Şimdi stored proc. İle veri tabanı, tablo vs. gibi nesneler oluşturalım. Tabi dışardan tablo ya da veri tabanı ismini alacak, kolonların isimlerini ve tiplerini de dışarıdan alacak.

--Tablo oluşturma--

--create proc Sp\_TabloOlustur

--(

--@tabloadi nvarchar(50),

--@kolon1adi nvarchar(50),

--@kolon1tipi nvarchar(50),

--@kolon2adi nvarchar(50),

--@kolon2tipi nvarchar(50),

--@kolon3adi nvarchar(50),

--@kolon3tipi nvarchar(50)

--)

--as

--create table @tabloadi

--(

--@kolon1adi @kolon1tipi,

--@kolon2adi @kolon2tipi,

--@kolon3adi @kolon3tipi

--)

--Not: Yukardaki şekilde stored proc. derlenmeyecektir. Çünkü tablo adı olarak gerçek bir değer istiyor. Parametreden gelen değeri kabul etmiyor. Tablo oluşturulan kısmı (as den sonraki kodları) exec(‘’) komutu içinde yazmamız gerekiyor.

create proc Sp\_TabloOlustur

(

@tabloadi nvarchar(50),

@kolon1adi nvarchar(50),

@kolon1tipi nvarchar(50),

@kolon2adi nvarchar(50),

@kolon2tipi nvarchar(50),

@kolon3adi nvarchar(50),

@kolon3tipi nvarchar(50)

)

as

exec ('create table ' + @tabloadi + '(' + @kolon1adi + ' ' + @kolon1tipi + ',' + @kolon2adi + ' ' + @kolon2tipi + ',' + @kolon3adi + ' ' + @kolon3tipi + ')')

exec Sp\_TabloOlustur 'MujdeDeneme','Id','int primary key identity(1,1)','Adi','nvarchar(50)','Soyadi','nvarchar(50)'

**Geriye Değer Döndüren proc örneği;**

-Geriye değer döndüren sp

--Sp den değer döndürmek istiyor isek output keywordunu kullanıyoruz.

--Örnek—

--Dışarıdan aldığı kategori ID sine göre o kategoride kaç adet ürün olduğunu döndüren sp yazınız ve sp yi çalıştıran kodları yazınız.

create proc sp\_AdetDöndür

(

@ID int,

@adet int output -- Geriye adet değerini döndüren parametre

)

as

BEGIN

select @adet=COUNT(\*) from Products where CategoryID=@ID

END

--Çalıştırmak için

--Yukarıdaki örneğimizde alınan “@ID” parametresine “3” değeri verilmiş olup geriye bu kategoriden 13 adet ürün olduğunu “output” anahtar kelimesi ile aldık ve procedure çağırıldığında dönüş değeri bir değişkende tutularak ekrana sonucunu yazdırmış olduk.

Go

declare @adet int

exec sp\_AdetDöndür 3, @adet output

select @adet as UrunAdet

Şimdi Ürünler tablosundaki Stok Adedini geri döndüren sp yazalım. ID parametresini alarak da sadece bir tane ürünün kaydının döndürelim.

create proc sp\_UrunStokGetir

(

@UrunID int,

@UrunAdet int output

)

as

BEGIN

select @UrunAdet=Products.UnitsInStock from Products where Products.ProductID=@UrunID

END

**“@UrunAdet INT OUTPUT”** kullanarak **@UrunAdet**adlı parametrenin **OUTPUT** anahtar kelimesi sayesinde**geriye değer döndürdüğü** belirtiyorum.

**Çalıştırılması ise;**

go

declare @adet int

execute sp\_UrunStokGetir @UrunId=1,@UrunAdet =@adet out

select @adet as StokDurumu

**\*\*\*\*\*\*\*\*\*Güzel Örnek\*\*\*\*\*\*\*\*\*\***

--4-Dısarıdan girilen kategori adı ve urun adı alan bir proc yazınız.

--eğer böyle bir kategori varsa urun o kategoriye eklensin

--yoksa önce kategoriyi ekleyip daha sonra urunu o kategoriye eklesin

create proc KategoriEkle

(

@name nvarchar(50),

--Bir procedure içerisinden dışarıya değer döndürülmek isteniyorsa output kullanılır.

@id int output

)

as

insert into Categories (CategoryName) values (@name)

set @id=(select SCOPE\_IDENTITY())

create proc UrunEkleyici

(

@urunadi nvarchar(50),

@kategoriadi nvarchar(50)

)

as

declare @katid int

if not exists(select \* from Categories c where c.CategoryName=@kategoriadi)

begin

exec KategoriEkle @urunadi,@katid output

end

else

begin

set @katid=

(select c.CategoryID from Categories c where c.CategoryName=@kategoriadi)

end

insert into Products (CategoryID,ProductName) values (@katid,@urunadi)

exec UrunEkleyici 'deniyoruzz','mujde'

**Özet olarak;**

-- Parse -> SQL ifadesinin syntax'ı kontrol edilir.

-- Resolve -> SQL ifadedeki tablo ve kolonların varlığı kontrol edilir.

-- Optimize -> SQL ifadesi sonuca en çabuk nasıl gidecek ona karar verilir.

-- Compile -> SQL ifadesi derlenir.

-- Execute -> SQL ifadesi çalıştırılır.

--Normalde bir SQL ifadesi çalıştırıldığında bu 5 adım gerçekleşir. SP kullanırken ise sadece compile ve execute işlem adımları gerçekleşir.

-- SP'ler bu yüzden çok daha hızlı çalışırlar.

-- SP'ler parametreli çalıştıkları için SQL Injection'a kapalıdırlar. Bu yüzden son derece güvenlidirler.

-- SP'leri bir kere yazıp birçok yerde kullanabiliriz.