**VERİTABANI –II**

#  2.TRANSACT SQL (T-SQL)

Veritabanı üzerinde tutulan bilgiler üzerinde sadece düzenleme ve temel işlemleri yapma yeteneğine sahip olan SQL ile programlama yapılamamaktadır. SQL in kendi yapısında programlama yapmayı sağlayacak ifadeler bulumadığından büyük çaplı projelerde veya veritabanı programlama gerektiren durumlarda yetersiz kalmaktadır.

 Temel SQL de bulunan bu eksikliklerin giderilmesi amacı ile Sysbase ve Microsoft firmalarının ortaklaşa çalışmaları sonucu programlamaya olanak sağlayan koşul ifadeleri döngü ifadeleri gibi ifadelerinde bulunduğu T-SQL programlama dili geliştirilmiştir. Benzer bir şekilde oracle firması da kendi veritabanı sistemleri üzerinde programlamaya olanak sağlayan PL/SQL programlama dilini geliştirmiştir.

 T-SQL dili ile klasik bir programlama dilinde yapılabilecek birçok işlem yapılabilmektedir. Bu dil ile değişken tanımı, farklı türlerde veri tanımı, döngüler ,koşul ifadeleri ve hata denetimi gibi işlemler kolaylıkla yapılabilmektedir.

 Ayrıca bu programlama dili ile c# ,vbasic gibi orta sevye programlama dilleri ortak bir çalışma mekanizmasına sahip olacak şekilde tasarlanmışlardır. Yani bir web uygulaması geliştirirken arka planda veritabanı uygulaması senkronize bir şekilde veri işleme işini ortaklaşa bir şekilde yapabilmektedirler.

## 2.1 DEĞİŞKEN TANIMLAMA

T-SQL de değişkenler belli bir türdeki verilerin tek veri değerini tutan programlama elemanlarıdır. Bu elemanlar program çalıştığı sürece aktif olarak kalabilmekte gerektiğinde tutulan değer değiştirilebilmekte veya silinebilmektedir. Bu değişkenler oluşutuluduldukları anda hafızada kendi veri türlerine göre bir alan oluşturulur ve programın çalışması bitiiğinde değişkenin bulunduğu alan serbest bırakılmaktadır.

Program içerisinde bir değişken kullanılabilmek için öncelikle onun tanımlanmış olması gerekmektedir. T-SQL programlama dilinde değişken tanımı yapılırken değişken isminin öncesinde “@” ifadesi kullanılmak zorundadır. Her programlama dilinde olduğu gibi T-SQL de de doğru değişken tanımlama koşulları bulunmaktadır-SQL değişken tanımlarken dikkat edilmesi gereken kurallar şöyledir.

* Değişken isimleri tanımlanırken Boşluk karakteri kullanılmamalıdır.

Yanlış Doğru

@degisken 1 @degisken1

* Değişken isimleri tanımlanırken Türkçe karakter kullanılmamalıdır.

Yanlış Doğru

@değişken @degisken

* Değişken isimleri rakam, harf ve \_ karakterini içerebilir. Fakat her durumda başlangıç değeri bir harf olmak zorundadır.

Yanlış Doğru

@\_degisken1 @degisken\_1

* Değişken isimlerinde büyük küçük harf ayrımı kullanılmamaktadır.

@degisken = @DegiSkeN

* And, Not, While gibi T-SQL ifadeleri değişken olarak kullanılamazlar

Yanlış

@Not, @in,@Like

Değişken Tanımlanması ve tanımlanan bir değişkene değer atama işlemi aşağıdaki şekilde yapılmaktadır;

T-SQL dilinde bir değişken tanımlanmadan önce *Declare* ifadesi kullanılmak zorundadır.Bu ifadeden sonra daha öncede belirtildiği gibi @ anahtar ifadesi kullanılarak oluşturulacak olan değişkenin ismi yazılır. Değişken ismi yazıldıktan sonra son olarak o değişkenin hangi veri tipine ait olduğu ve ne kadarlık bir boyutta ihtiyaç duyduğu belirtilir.

Örnek;

Declare @degisken\_ismi as veritipi[ihtiyaç duyulan boyut miktarı]

Declare sayac as int 🡪 integer tipinde bir değişken tanımlanmıştır

Çoğunlukla kullanılan bazi veri tiplerine uygun değişken tanımlamaları aşağıdaki şekilde yapılmaktadır.

*Declare @sayilar as int*

*Declare @harf as char(1)*

*Declare @buyuk\_boyutta\_yazi as text-- 8000 'e kadar karakter alabilir*

*Declare @karekter\_katari as nvarchar(50)--0-4000 arası değer alablir*

*Declare @karekter\_katari1 as varchar(50)--0-4000 arası değer alablir*

*Declare @ondalikli\_sayi as float*

*Declare @tarih as Datetime*

*Declare @parasal\_deger as Money*

Ayrıca değişken tanımı yapılırken tek bir satırda farklı veri tiplerine ait değişken tanımlamalarıda yapılabilmektedir.

*Declare @degisken\_1 Money,@degisken2 int , @degisken3 float*

**Değişkenlere Değer atama işlemi;**

T-SQL de tanımlanan değişkenlere değer atama işlemi iki yöntemle yapılabilmektedir. Çoğunlukla kullanılan yöntem SET anahtar sözcüğü ile doğrudan değer atama işlemidir.Bu yöntemde Set ifadesinden sonra değişken adı yazılır ve arada eşittir ifadesi olmak koşulu ile ifadenin sağ tarafına değişkene atanmak istenen değer atılır.

Örnek;

*Declare @degisken1 as int*

*Declare @degisken2 as int*

*Declare @Sonuc as int*

*SET @degisken1=5*

*SET @degisken2=10*

*SET @Sonuc=@degisken1 + @degisken2*

Set ifadesi ile Değer atama işlemi yukarıdaki örnekte görüldüğü gibi yapılabilmektedir.

 Tanımlanan bir değişkene Atanacak değer karekter veya karekter katarı şeklinde bir değer içeriyorsa bu değerin bir değişkene atanabilmesi için öncelikle tek tırnak ifadesi içerisine alınması gerekmektedir.

Aşağıda örnek gösterimi yapılmaktadır.

*Declare @degisken1 as varchar(15)*

*Set @degisken1=” Deneme değeri”*

Değer atama yapılırken kullanılan diğer bir yöntem ise sorgu içi atama yöntemidir. Bu yöntemde Select ifadesi ile yapılan bir sorgulama işlemi sonucunda dönen değer bir değişkene atanabilmektedir. Burada dikkat edilmesi gereken önemli bir nokta bulunmaktadır. Sorgu sonucunda dönen değer bir değişkene atanmak istendiğinde sorgudan sadece bir değer döndürüldüğüne dikkat edilmelidir aksi taktirde hata ile karşılaşılacaktır. Bu hatanın sebebi ise tek bir atama işlemi için hafızada yer ayırılmışken birden fazla değeri hafızanın aynı bölgesine yazma isteğidir.Hafızada böyle bir alan tahsis edilmediği için server tarafından yapılmak istenilen işlemin hatalı olduğuna dair bir uyarı alınmaktadır.

Örnek;

*Declare @ogrenci\_sayisi as int*

*Select @Ogrenci\_sayisi =Count(\*) from Ogrenci\_Tablosu*

Örnekte görüldüğü üzere Count(\*) ifadesi ile Ogrenci\_Tablosu tablosunda bulunan öğrenci sayısı döndürülmektedir.Yani tek bir değer döndürülmekte ve buda tanımlanan değişkene sorunsuz bir şekilde değer ataması yapılmaktadır.

Select ifadesi ile değer atama yapılırken sadece sorgudan değer döndürme zorunluluğu yoktur. Bu ifade ile sabit değerleride değişkenlere atamak mümkündür.

*Declare @ogrenci\_sayisi as int*

*Select @Ogrenci\_sayisi =1550*

Örnek;



## 2.2 GO İLE YIĞIN KONTROLÜ

Veri tabanı sistemlerinde programlama yapılırken çoğunlukla birden fazla ifadenin bir araya gelerek çalıştırılması istenmektedir.SQL serverde bu tarz işlemleri yapmamıza olank sağlayan yığın yönetim mekanizması bulunmaktadır.Bu mekanizma ile ard arda yazılımış Sql ifadeleri sırası ile çalıştırılmaktadır. Birden fazla SQL sorgusunun belli gruplar halinde çalıştırılıp server’a gönderilme işlemi GO yığın işaretçisi ile yapılmaktadır. Sql Server GO komutunu gördükten sonra o ana kadar olan bütün ifadeleri işlenmek üzere server’ gönderir.

Kullanım Şekli Şöyledir;

T-sql ifadeleri

T-sql ifadeleri Sql Yığın Gösterimi

T-sql ifadeleri

GO

 Dikkat edilmesi gereken hususlardan biri GO ifadesinden sonra yazılan SQL ifadelerinin işleme alınmayacağı ve sistem tarafından hata ile geri dödürüleceğidir.Benzer şekilde GO ile aynı satırda Sql ifadelerininde bulunmaması gerektiğide bilinmelidir.

Örnek;



Görüldüğü gibi GO ifadesinden sonra gelen Sql kodu işleme alınamadı ve çalıştırılamadı. Bu hatanın nedeni @degisken ile ifade edilen değişkenin GO ifadesinden önce tanımlanmış olması ve GO yığını dışında yaşam alanına sahip olmamasıdır. Bu nedenden dolayı server böyle bir değişkeni tanımamakta ve hata dönüşü yapmaktadır. Yığının sonunda bulunan sorgu yığın içerisine dahil edildiğinde hata ortadan kalkacak ve sorgu sonucu doğru bir şekilde sonuçlanacaktır.



## 2.3 PRINT İFADESİ

 T-SQL ifadeleri kullanılarak programlama yapılırken değişken içeriklerinin gösterilmesi, kullanıcıya bilgi verilmesi yada hata kontrolleri gibi bazı durumlarda ekrana bilgi yazdırılma ihtiyacı doğmaktadır. Çoğu programlama dilinde bu işlem printf, echo,write gibi ifadeler kullanılmaktadır. T-Sql de bu işlemleri yapmak için Print ifadesi kullanılmaktadır. Print ifadesi yerine bazı durumlarda Select ifadeside kullanılmaktadır. Fakat bu iki ifadeyi bir birinden ayıran en önemli özellik uyarı metninin gösterildiği ekrandır. Select ifadesi bir Sql sorgu ifadesi olduğundan dolayı Sql server sonucu bir tablodan çekilmiş gibi göstermektedir. Print ifadesinde ise sadece Uyarı metni gösterilmektedir. Örnek kullanım ve sonuç gösterim şekli aşağıdaki şekilde görülmektedir.

 

Örnek;



## 2.4 TABLO TİPİ DEĞİŞKEN TANIMI VE KULLANIMI

Programlama yapılırken bazı durumlarda tek bir değişken üzerinde birden fazla değer tutulması ihtiyacı duyulmaktadır. Orta seviyeli ve ileri seviyeli programlama dilleri bu işlemi yapabilmek için diziler mantığından faydalanmaktadırlar ve dizileri kullanarak birden fazla bilgiyi tek bir değişken üzerinde tutabilmektedirler.T-SQL de bu işlemi yapabilmek için tablo tipi değişkenler kullanılmaktadır.Bu yapılar diziler ile tamamen aynı yapıda değillerdir daha çok veritabanında bulunan fiziksel tablolara benzemektedirler.

Tablo tipi değişkenler de diğer değişkenler gibi aynı tanımlama formatına sahip olmakla beraber aradaki tek fark veri tipi belirlenirken TABLE ifadesinin kullanılmasıdır.

Declare @tablo\_degisken\_ismi TABLE

(

Degisken1 as veri tipi,

Degisken2 as veri tipi

)

Tablo tipi değişkenler tanımlanırken fiziksel bir tablo tanımında kullanılan sabit değilken ifadeleride kullanılabilmektedir.

Declare @banka TABLE

 (

ID int IDENTITY(1,1) ,

Kullanici adi varchar(200) NOT NULL,

Sifre varchar(20)

 )

Tablo tipi değişkenler kullanılarak bu değişkenlere fiziksek table gibi işlemler yapmak mümkündür. Örneğin geçici bir tablo ile fiziksel bir tabloyu birleştirip sorgulamalar yapılabilir, fiziksel bir tabloda ekleme, çıkarma gibi temel işlemler geçici tablolar üzerinde de aynı şekilde yapılabilmektedir.



Tablo tipi değişkenlere bilgi atama ve bu bilgiler üzerinde işlemler yapmak kolaylıkla yapılabilmektedir;



Benzer şekilde güncelleme ve silme işlemleride aşağıdaki şekilde yapılabilmektedir.

