

T-SQL

► T-SQL NEDİR?

SQL dilinin yeteneklerinin sınırlı olması sebebiyle, SQL üzerine çeşitli iyileştirmeler ve eklemeler yapılmıştır. Örneğin Oracle firması SQL üzerine yaptığı iyileştirmeleri standartlaştırmış ve PL-SQL adını vermiş ve geliştiricilerine sunmuştur. **Microsoft** kendi platformu için SQL üzerine yaptığı iyileştirmeleri standart haline getirmiş ve **T-SQL** ismini vermiştir. **Transact-SQL**'in kısaltması olan T-SQL günümüz veritabanı yönetim ihtiyaçlarının tamamını karşılayabilecek yeterliliğe sahiptir.

T-SQL

- **T-SQL verileri işleme**, değişken kullanma ve hata ayıklama gibi bir çok programlama yeteneğine sahiptir. Tabiki bu yetenekleri veritabanı yönetim sistemi (*SQL Server için Management Studio*) üzerinde geçerlidir. Yani bir programlama dili değil gelişmiş bir **sorgu dilidir**.

T-SQL ile veritabanı üzerinde işlem yapabileceğimiz temelde 3 komut türü vardır.

- Veritabanı ve tablolarla ilgili işlemler yapmak için **Veri Tanımlama Dili** (*Data Definition Language*),
- Veritabanı üzerindeki kullanıcılar ve bu kullanıcıların yetkileri ile ilgili işlemler yapmak için **Veri Kontrol Dili** (*Data Control Language*)
- Veritabanında saklanan veriler üzerinde işlem yapmak için kullanılan **Veri İşleme Dili** (*Data Manipulation Language*)

T-SQL

- SQL deyimleri veritabanları üzerinde çeşitli işlemleri yerine getirir. Veri tabanından sorgulama yapmak için SELECT, ekleme yapmak için INSERT güncelleme yapmak için UPDATE, silme yapmak için DELETE, yeni tablo oluşturmak için CREATE TABLE gibi komutlara sahiptir.

Bu komutlar, işlevlerine göre şu şekilde ayrılır:

- DDL (Data Definition Language): Veri tanımlama dili
- DML (Data Manipulation Language) : Veri işleme dili
- DCL (Data Control Language): Veri kontrol dili

T-SQL

► Veri Tanımlama Dili (DDL)

SQL Server içinde veri tabanı, tablo ve kullanıcı tanımlı veri tipleri gibi nesnelere

oluşturmak ve bunları yapılandırmak için kullanılır. Temel komutları aşağıdaki şekildedir:

Temel Komutlar Açıklama

CREATE Nesne oluşturmak için kullanılır.

ALTER Nesnelere üzerinde değişiklik yapmak için kullanılır.

DROP Nesnelere silmek için kullanılır.

T-SQL

► Veri İşleme Dili (DML)

Veri tabanı içindeki veriler ile ilgili işlemler yapılmasını sağlar. Temel komutları aşağıdaki şekildedir.

Temel Komutlar Açıklama

SELECT Veri tabanındaki verileri seçmeyi sağlar.

INSERT Veri tabanına yeni veriler eklemek için kullanılır.

UPDATE Veriler üzerinde değişiklik (güncelleme) yapmak için kullanılır.

DELETE Veri tabanından veri silmek için kullanılır.

T-SQL -Login Oluřturma

► Veri Kontrol Dili (DCL)

DCL, bir veri tabanı ile iliřkili kullanıcıları ve rollerin izinlerini deęiřtirmek için kullanılır. Dięer bir deyiřle verilere eriřim yetkilerini dñzenlemede kullanılır. Temel komutları ařaęıdaki řekildedir.

Temel Komutlar Açıklama

GRANT Bir kullanıcının verileri kullanmasına ve T-SQL komutlarını alıřtırmasına izin verir.

DENY Bir kullanıcının verileri kullanmasını kısıtlar.

REVOKE Daha önce yapılan tüm kısıtlama ve izinleri iptal eder.

DCL komutlarını kullanabilmek için SQL Server'da varsayılan deęer (default) olarak yetki sahibi olan gruplar: sysadmin , dbcreator , db_owner , db_securityadmin 'dir.

Sunucuya dıřarıdan bir eriřim saęlamak için bir giriř (login) oluřturulmalıdır.

T-SQL ile ÇALIŞMAK

2.1. Değişkenler

Değişken, verilerin bellekte geçici olarak kaydedilmesini ve gerektiğinde kullanılmasını sağlayan değerdir. T-SQL kullanmanın en büyük kolaylıklarından biri de değişken kullanımına olanak tanımasıdır. Burada ifade edilen; değişken diğer tüm programlama dillerinde yer alan bir veri tipi ile sınırlandırılmış, oluşturulmasının ardından hafızada belli bir yer kaplayan, üzerine veri ataması yapılabilen ve daha sonra ismi kullanılarak program içerisinden çağrılıp kullanılacak yapıdır.

SQL Server'da da değişkenler yerel ve genel olmak üzere ikiye ayrılır. Yerel değişkenler, "@" ön eki ile tanımlanır (@değişken). Genel değişkenler ise SQL Server tarafından tanımlanmıştır ve kullanıcı tarafından oluşturulamaz. "@@" ön eki ile tanımlanırlar (@@SERVERNAME). Genel değişkenler genellikle SQL Server hakkındaki bilgileri verir. SQL Server'da tanımlanmış birçok genel değişken vardır.

2.1.1. Nesne ve Değişken İsimlendirme Kuralları

Nesne veya değişkene bir isimlendirme yaparken aşağıdaki kurallara dikkat etmelisiniz:

- Harf veya alt çizgi (_) ile başlamalıdır.
- Türkçe karakterler ve boşluk isimlendirmede kullanılmamalıdır.
- Değişken ismi SQL'de özel anlamı olan sembollerle (@, @@, #, ##, \$) başlamamalıdır.
- T-SQL komutları değişken ismi olarak verilmemelidir (SELECT, UPDATE vb).

T-SQL ile ÇALIŞMAK

- SQL ifadeleri prensip olarak büyük harfle yazılır.
- Nesne isimleri kısa ve anlamlı olmalıdır.
- Nesne isimlendirilirken işlerin kolaylaştırılması açısından tekil isim tercih edilmelidir (TabloOgrenciler yerine tblOgrenci gibi).
- NULL terimi, daha önce hiçbir şey girilmemiş (değersiz) anlamındadır. Klavyedeki SPACE (ASCII 32) tuşu ile NULL aynı değerleri içermez. NULL boş veya bilinmeyen değerler için kullanılır.

2.1.2. Değişken Tanımlama

SQL Server'da değişkenler DECLARE ifadesi kullanılarak oluşturulur.

➤ Yazım Şekli

```
DECLARE @degisken_adi <veri_tipi> [(boyut)]
```

Örnek:

```
DECLARE @ogr_no VarChar(10)  
DECLARE @tckimlik_no int
```

Aralara virgül koyarak da birden fazla değişkeni tek bir DECLARE ifadesi ile oluşturabilirsiniz.

Örnek:

```
DECLARE @ogr_no varchar(10),@tckimlik_no int
```

Varchar, int türlerinde değişken tanımlayabildiğiniz gibi tablo türünde değişken de tanımlayabiliriz.

T-SQL ile ÇALIŞMAK

Değişken, verilerin bellekte geçici olarak kaydedilmesini ve gerektiğinde kullanılmasını sağlayan değerdir. T-SQL kullanmanın en büyük kolaylıklarından biri de değişken kullanımına olanak tanımasıdır. Burada ifade edilen; değişken diğer tüm programlama dillerinde yer alan bir veri tipi ile sınırlandırılmış, oluşturulmasının ardından hafızada belli bir yer kaplayan, üzerine veri ataması yapılabilen ve daha sonra ismi kullanılarak program içerisinde çağrılıp kullanılacak yapıdır.

Değişkenler şu şekilde tanımlanır:

```
declare @degisken_adi veritipi[(boyut)]
```

T-SQL ile ÇALIŞMAK

Örnek:

```
declare @kitapNo INT  
declare @kitapAdi VARCHAR(63)  
  
go
```

Aralara virgül koyarak da birden fazla değişkeni tek bir DECLARE ifadesi ile oluşturabilirsiniz.

```
declare @kitapNo INT,@kitapAdi VARCHAR(63)  
  
go
```

Değişkenlere değer atama,

```
declare @kitapNo INT, @kitapAdi VARCHAR(63)  
  
SET @kitapNo=255  
  
SET @kitapAdi= '107 Kimya Öyküsü'  
  
SET @kitapNo=256 – Artık kitapNo değişkeninin değeri 256,255 silindi.
```

T-SQL ile ÇALIŞMAK

Ancak T-SQL'in asıl amacı SQL'in yeteneklerini artırmaktır. Bundan dolayı değişkenlerin en genel kullanım amacı, bir sorgunun sonucundaki değerlerden birini alıp bir değişkene aktarmaktır.

```
declare @enSonEklenenKitap INT  
  
SELECT @enSonEklenenKitap=MAX(kitapNo)  
  
FROM Kitap
```

SQL Server'da da değişkenler yerel ve genel olmak üzere ikiye ayrılır.

Yerel değişkenler, "@" ön eki ile tanımlanır (@değişken).

Genel değişkenler ise SQL Server tarafından tanımlanmıştır ve kullanıcı tarafından oluşturulamaz. "@@" ön eki ile tanımlanırlar (@@SERVERNAME). Genel değişkenler genellikle SQL Server hakkındaki bilgileri verir. SQL Server'da tanımlanmış birçok genel değişken vardır.

T-SQL ile ÇALIŞMAK

Bir değişken oluşturulduğunda NULL değere sahiptir. Değişkenlere değer atamanın SET, SELECT ve tablolar için INSERT INTO gibi birkaç farklı şekli vardır.

➤ **SET ifadesi kullanılarak değişkene değer atama**

SET @değişken_adi=değer

şeklinde yapılır.

➤ **SELECT ifadesiyle değer atama**

SELECT @değişken_adi=değer

şeklinde yapılır.

➤ **Tablo değişkenlere INSERT INTO ifadesi ile değer atama**

INSERT INTO @tablo_değişken SELECT adi, soyadi FROM person

ifadesi ile person tablosunun adı ve soyadı sütunlarının içerdiği değerlerden oluşan bir tabloyu @tablo_değişken adlı değişkene atamış olursunuz.

Tip: SET vs SELECT

- SET is the ANSI standard for variable assignment, SELECT is not.
- SET can only assign one variable at a time, SELECT can make multiple assignments at once.
- If assigning from a query, SET can only assign a scalar value. If the query returns multiple values/rows then SET will raise an error.

T-SQL ile ÇALIŞMAK

1. SQL Veri Tipleri: (Sık Kullanılanlar)

Tam Sayılar:

BIGINT : 8 byte olarak depolanır, -2^{63} (-9223372036854775808) ile $2^{63}-1$ (9223372036854775807) aralığındaki tüm tam sayıları kapsar.

INT : 4 byte olarak depolanır, -2^{31} (-2.147.483.648) ile $2^{31} - 1$ (2.147.483.647) aralığındaki tüm tam sayıları kapsar.

SMALLINT : 2 byte olarak depolanır. -2^{15} (-32,768) ile $2^{15} - 1$ (32,767) aralığındaki tüm tam sayıları kapsar.

TINYINT : 0 ile 255 arasındaki tüm tam sayıları ifade eder. Bir byte yer tutar.

Kesirli Sayılar (Gerçek Sayılar):

FLOAT : $-1.79E + 308$ den $1.79E + 308$ 'e kadar olan tüm gerçek sayılar. Float kullanılırken FLOAT(n) şeklinde kullanılabilir. Burada n basamak sayısını ifade eder. Maksimum 53 yazılabilir. N 1 ile 24 arasında ise float 4 byte yer tutar, eğer 25-53 arasında ise 8 byte yer tutar.

n	Basamak sayısı (Hassasiyet)	Hafızada kapladığı alan
1-24	7	4 byte
25-53	15	8 byte

T-SQL ile ÇALIŞMAK

Diğer Sayı Biçimleri

DECIMAL : Genelde işlemlerde kullanılmayan ancak sayı olarak yazılan verilerde kullanılır. Decimal(n) şeklinde kullanılır. n maksimum kullanılacak basamak sayısıdır.

NUMERIC : Decimal ile aynı işlevi görmektedir. Numeric(n) şeklinde kullanılır. n maksimum kullanılacak basamak sayısıdır.

Bit (doğru/yanlış)

BIT : 1 veya 0 olabilir.

Para Değerleri

MONEY : -2^{63} (-922,337,203,685,477.5808) ile $2^{63} - 1$ (+922,337,203,685,477.5807) arasındaki tüm para değerleri (8 byte)

SMALLMONEY : -214,748.3648 ile +214,748.3647, arasındaki para değerleri. (4 byte)

T-SQL ile ÇALIŞMAK

DATETIME

: Tarih ve saat tarih formatları, 1 Ocak 1753 ten başlar, 31 Aralık 9999 yılına kadar devam eder. Saat, dakika, saniye ve salise ile beraber kullanılır.

SMALLDATETIME

: 1 Ocak 1900 den 6 Haziran 2079 tarihine kadarki değerleri alır. Saat, dakika ve saniye ile beraber kullanılır.

Yazı Tipleri

CHAR(n)

: Unicode olmayan yazıları kaydetmek için kullanılır, n 1 ile 8000 arasında olmalı. Ayrılan yer tamamıyla kullanılır.

VARCHAR (n)

: Unicode olmayan yazıları kaydetmek için kullanılır, n 1 ile 8000 arasında olmalı. Maksimum n karakter uzunluğunda yazılar kaydedilebilir. Yazılan karakter sayısı kadar yer kullanılır.

TEXT(n)

: Unicode olmayan maksimum $2^{31} - 1$ (2.147.483.647) karakter uzunluğunda yazılan kaydetmek için kullanılır.

UYARI: VARCHAR ile CHAR arasındaki fark; Her ikisinde de maksimum ayrılan alan kadar yazı girişi yapılabilir, ama VARCHAR ile belirtilen alan hafızada yazılan karakter sayısı kadar yer tutar, CHAR ile belirtilen alan ise ne kadar yazılırsa yazılsın, belirtilen alanın tamamını kullanmış olur. Örneğin isim alanı VARCHAR(25) ile belirtilmiş olsun, bu durumda isim için "Ali" girilirse hafızada 3 karakter yer tutar, "Mehmet" girilirse 6 karakter yer tutacaktır. Örneğin isim alanı CHAR(25) ile belirtilmiş olsun, bu durumda isim için "Ali" de girilirse "Mehmet" te girilirse veya "Mustafa Furkan" da girilirse yine hafızada 25 karakter yer tutacaktır. Bunlara rağmen CHAR(25) te olursa VARCHAR(25) te olursa maksimum 25 karakter uzunluğunda bir isim girilebilir.

T-SQL ile ÇALIŞMAK

Değişkenlerin Tip Dönüşümleri

Tip dönüşümü iki şekilde yapılabilir:

1- CAST kullanarak değişken tip dönüşümü yapılır:

```
CAST (degisken_adi AS veri_tipi(uzunluk))
```

Örnek:

```
CAST(@a as varchar(50))
```

2- CONVERT kullanarak değişken tip dönüşümü yapılır:

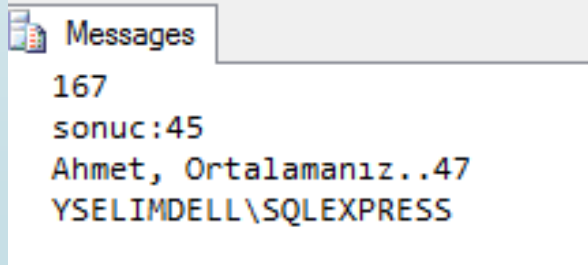
```
CONVERT (veri_tipi, degisken_adi, stil)
```

Örnek:

```
CONVERT(varchar(50), @a, 102)
```


T-SQL ile ÇALIŞMAK

```
declare @vizeNotu tinyint,@final tinyint,@adiSoyadi nvarchar(20)
declare @tur nvarchar(10), @yeni int
set @adiSoyadi='Ahmet'
set @vizeNotu=45 set @final=50;
print cast('123' as int)+44
Set @yeni=cast('123' as int)+44
print 'sonuc:'+cast (@vizeNotu as varchar(10))
print @adisoyadi+', Ortalamanız..' +cast ((@vizenotu+@final)/2 as nvarchar(10))
print @@SERVERNAME
```



Messages

```
167
sonuc:45
Ahmet, Ortalamanız..47
YSELIMDELL\SQLEXPRESS
```

T-SQL ile ÇALIŞMAK

SELECT_CASE_THEN

SELECT_CASE_THEN: Sql cümlelerimiz içinde belirli durumlara göre farklı işlemler yapmak istiyorsak Case-When yapısını kullanabiliriz. Personel tablosunda Eğer bolno=1 ise ekrana 'elektrik' 2 ise 'bilisim' diğer durumlarda 'başka bölüm' yazacak Select-case satırları

```
use pers
SELECT ad,soyad,

CASE

WHEN bolno = 1 THEN 'elektrik'

WHEN bolno=2 THEN 'bilisim'

ELSE 'baska bölüm'

END as BolumDurumu
From PersTablosu
```

T-SQL ile ÇALIŞMAK

SELECT_CASE_THEN

Bu örnekte her kayıt için maaş durumuna göre yüksek ve düşük kazanç yazdırılmakta

```
SELECT adi,soyadi,  
  
CASE  
  
WHEN maas >2000 and maas<3000  
THEN 'yüksek kazanç'  
  
WHEN maas<=2000 THEN 'düşük'  
  
ELSE 'baska maas'  
  
END as MaasDurumu  
  
from isciler
```

Tanımlı vize notuna göre geçti kaldı durumu kontrolü?

```
declare @vize as tinyint  
declare @vizeDurumu as nvarchar(10);  
set @vize=55;  
set @vizeDurumu=case  
when @vize<50 then 'kaldi'  
when @vize>=50 then 'geçti'  
else 'sınava girmemiş'  
end  
print @vizedurumu
```

T-SQL ile ÇALIŞMAK

Öğrenciler tablosu listesinde 7. öğrencinin bölüm kodunu nasıl elde ederiz. Sırano yada ogrno gibi bir alan yok yada sıralı artmıyorsa..

```
declare @bolumkod as tinyint
select top 1 @bolumkod=bkod from @bolumkodTablo
print @bolumkod
```

Öğrenciler tablosundaki bulunan bölümlerin listesini **ayrı bir değişkene** aktarıp case-when ile bölüm koduna göre bölümleri yazdırınız.

```
declare @bkodTablo as table(bolumk tinyint)
insert into @bkodTablo select BolumKodu from Ogrenciler
select bolumk as [Bölüm Kodları] from @bkodTablo
```

T-SQL ile ÇALIŞMAK

IF-ELSE YAPISI

IF-ELSE yapısı diğer programlama dillerindeki kullanım şekline benzemektedir. Koşul gerçekleştiğinde IF bloğu , aksi takdirde ise ELSE bloğu icra görür

```
IF (koşul)
  BEGIN
  --Eğer koşulumuz doğru ise bu alandaki ifademiz çalışır.
  END
ELSE
  BEGIN
  --Eğer koşulumuz doğru değilse bu alandaki ifademiz çalışır.
  END
```

T-SQL ile ÇALIŞMAK

IF-ELSE YAPISI

SatisTablosu tablosundaki verilerin sayısına bağlı olarak şöyle bir IF ELSE yapısı kurulsun. Eğer kayıtlar 30 binden fazla ise, 30 binden fazla kayıt var, değilse 30 binden az kayıt var yazdırılsın.

```
DECLARE @Sayi INT;  
SELECT @Sayi = COUNT(*) FROM SatisTablosu;  
IF @Sayi > 30000 BEGIN  
PRINT '30 binden fazla kayıt var';  
END  
ELSE BEGIN  
PRINT '30 binden az kayıt var';  
END;
```

T-SQL ile ÇALIŞMAK

IF-ELSE YAPISI

İkinci örneğimize geçelim. Bu örnekte iç içe IF bloğu kullanarak gerçekleştirilecek bir örnek olsun. Dış blokta ay kontrolü yapalım. Diyelim ki ay Eylül'mü. İç blokta da yıla bakalım. Yıl tek bir yıl mı, çift bir yıl mı? Yani rakam değerleri tek mi, çift mi?

```
IF MONTH(GETDATE()) IN (9) BEGIN
PRINT 'Aylardan Eylül';
  IF YEAR(GETDATE()) % 2 = 0 BEGIN
PRINT 'Çift bir yıldayız';
  END
  ELSE BEGIN
PRINT 'Tek bir yıldayız';
  END
END;
```

T-SQL ile ÇALIŞMAK

T-SQL IF EXISTS

Personel tablosunda maaşı 5000tl altında personel var ise bu personelin bölümlerini ve isimlerini **if exists** kullanarak bulun.

```
use pers
declare @adet as tinyint
if exists(select bolno from PersTablosu where maas<5000)
begin
    select ad,
    case
    when bolno=1 then 'elektrik'
    when bolno=2 then 'bilisim'
    else 'bolumsuz'
    end
    as [Bolum Bilgisi]
    from PersTablosu
    where maas<5000
end
```


T-SQL ile ÇALIŞMAK

T-SQL WHILE

MS SQL'de de WHILE döngüsünü kullanabiliriz. Tıpkı .NET programlama ve diğer programlama platformlarında veya dillerinde olduğu gibi WHILE döngüsü SQL'de de bir şarta bağlı olarak sonlandırılabilir. Yani ilgili şartın TRUE ya da FALSE olması ile birlikte durdurulabilir.

```
DECLARE @Sayac INT = 1;  
WHILE @Sayac < 10  
BEGIN  
PRINT @Sayac;  
SET @Sayac += 1;  
END;
```

T-SQL ile ÇALIŞMAK

T-SQL WHILE EXISTS

Maaşı 5000tl altında kayıt mevcut (**while-exists**) ise döngü değeri sayaç değerini arttıracak.

```
use pers
declare @sayac int=0
while exists(select sicilNo from PersTablosu
where maas>5000)
begin
    set @sayac=@sayac+1
Update pers set maas=maas*0.9
end
print 'toplam:'+cast(@sayac as nvarchar(10))
```

T-SQL ile ÇALIŞMAK

Değişken Tanımlama Örnekleri

```
declare @ogrno int
set @ogrno=10
select 'ogr notu:' + convert(varchar,@ogrno)
print 'ogr notu:' + convert(varchar,@ogrno)
```

While – If tek/çift sayı bulma

```
declare @sayac int
set @sayac=1
while(@sayac<100)
begin
select @sayac=@sayac+1
if(@sayac%2=0)
begin
print 'çiftsayı:' + convert(char,@sayac)
end
end
```

Table tanımlama-Insert into Table Select

```
use pers
```

```
declare @tablo table(ad nvarchar(20))
```

```
insert into @tablo select ad from
PersTablosu
```

```
select * from @tablo
```

T-SQL ile ÇALIŞMAK

1-100 arası tek sayıları bulan 'tekler' isimli yeni bir Tablo değişkenine aktaran ve bu listede 5. tek sayıyı değişkene atayan kodu yazınız?

```
declare @tekler as table(sirano tinyint identity(1,1), tek tinyint)
declare @i tinyint
set @i=0
while (@i<100)
begin
    if(@i%2=1)
        begin
            insert into @tekler values(@i)
        end
    set @i=@i+5
end
--5. tek sayı hangisidir?
declare @sonuc tinyint
select top 5 @sonuc=tek from @tekler
print @sonuc
--yada
set rowcount 5
select @sonuc=tek from @teklek
print @sonuc
set rowcount 0
```

T-SQL ile ÇALIŞMAK

1-100 arası tek sayıları bulan 'tekler' isimli Tablo değişkenine aktaran kodu yazınız?

```
declare @tekler as table(sirano tinyint tek tinyint)
declare @i tinyint
set @i=0
while (@i<100)
begin
    if(@i%2=0)
        begin
            insert into @tekler values(@i)
        end
    set @i=@i+1
end
select * from @tekler
```

T-SQL ile ÇALIŞMAK

CURSOR Kullanımı

SQL Server'da bir sorgu sonucu dönen kayıtlar üzerinde satır bazlı işlem yapmak için **CURSOR** kullanırız. CURSOR hangi satır üzerinde ise o satırda bulunan verilerle işlem yapılır. CURSOR kullandığımız Select cümlesinde dönen her kayıt bir değişkene atanmalıdır. Select cümlesinden hangi veri tipinde ne kadar kayıt dönecek ise o kayıt sayısı kadar aynı veri tiplerinde değişkenler tanımlanır.

T-SQL ile ÇALIŞMAK

CURSOR Kullanımı

CURSOR kayıtlar üzerinde dolaşmaya başlamadan önce **Open** komutu ile açılır, **Fetch Next** komutu ile kayıtlar üzerinde ilerlenir, kayıtlar ile ilgili işlemler bittikten sonra ise Close komutu ile kapatılır.

Bir veri tipi olarak da ele alınabilen Transact-SQL Sunucu Cursor su asamalardan geçirilerek kullanılır.

0. Cursor bir **SELECT** ifadesi için tanımlanır.
1. Select ifadesi hangi veri tiplerinde ne kadar sütun döndürecekse esdegeri degiskenler tanımlanır.
2. Cursor, resultset üstünde gezinilmek üzere OPEN deyimi ile açılır.
3. Resultset'in sonuna gelinceye kadar her seferinde bir kayıt olmak üzere **FETCH NEXT** komutu ile kayıtlar üstünde ilerlenir.
4. Resultset ile ilgili işlemler sona erdiğinde cursor **CLOSE** ile kapatılır. Ancak kapatılan cursor henüz hafızada yer kaplamaya devam eder. Gerek duyulursa, yeniden açılabilir. Ama bu cursor, kapalı bile olsa bir sonraki adıma geçilmeden aynı adda bir cursor tanımlanamaz.
- 5.5. Cursor ile ilgili işlerimiz bittiği anda hafızadan da silmek için cursor **DEALLOCATE** ile hafızadan boşaltılır.

T-SQL ile ÇALIŞMAK

CURSOR Kullanımı

Bir imleç satırlar üzerinde dört temel işlevi yerine getirir: İlk kayıta, son kayıta, önceki veya sonraki kayıta gidebilir ve SCROLL CURSOR olarak adlandırılırlar.

İfade	İşlevi
FETCH FIRST	İlk satıra konumlanır.
FETCH LAST	Son satıra konumlanır.
FETCH NEXT	Bir sonraki satıra konumlanır.
FETCH PRIOR	Bir önceki satıra konumlanır.
FETCH RELATIVE n	Bulunulan satırdan n kayıt ileriye konumlanır.
FETCH ABSOLUTE	Baştan n. kayıta konumlanır.

3.2.2. @@FETCH_STATUS ve @@ROWCOUNT

Yapılan işlemler sonucunda imlecin son satıra gelip gelmediğini anlamak için @@FETCH_STATUS ve @@ROWCOUNT fonksiyonları kullanılır.

FETCH_STATUS fonksiyonu, en son çalıştırılan FETCH komutunun sonucu hakkında bilgi verir.

FETCH_STATUS fonksiyonu üç farklı değer alabilir.

Değer	Açıklama
0	Bir önceki FETCH komutu başarıyla gerçekleştirildi.
-1	Bir önceki FETCH komutunda hata ile karşılaşıldı.
2	Son kayıta ulaşıldı. (end of resultset)

Tablo 3.1: @@FETCH_STATUS fonksiyonu değer tablosu

@@ROWCOUNT fonksiyonu, bir önceki FETCH komutu başarıyla gerçekleştirildikten sonra sonuç setinde toplam kaç kaydın bulunduğunu tutan fonksiyondur. FETCH komutu hiç kullanılmamışsa, imlecin işaretlediği sonuç setinde toplam kaç kaydın yer aldığını gösterir.

T-SQL CURSOR

@@FETCH_STATUS fonksiyonu, en son çalıştırılan FETCH komutunun sonucu hakkında bize bilgi verir. Bu fonksiyon, su üç değerden birini verecektir:

0 : Bir önceki FETCH komutu başarı ile gerçekleştirildi.

-1 : Bir önceki FETCH komutunda bir hata ile karşılaşıldı.

-2 : Resultset'teki tüm kayıtlar bittiği için en sona gelindi, daha fazla kayıt yer almıyor. (end of resultset)

--CURSOR kullanımı

```
declare @sicil tinyint
```

```
declare @ad nvarchar(20)
```

```
declare crs_pers cursor for select sicilno,ad from PersTablosu
```

```
open crs_pers
```

```
fetch next from crs_pers into @sicil,@ad
```

```
while(@@FETCH_STATUS=0)
```

```
begin
```

```
print 'sicil:'+cast(@sicil as nvarchar(20))
```

```
print 'ad:'+@ad
```

```
fetch next from crs_pers into @sicil,@ad
```

```
end
```

```
close crs_pers
```

```
deallocate crs_pers
```