VERİTABANI PROGRAMLAMA

2

Amaçlarımız

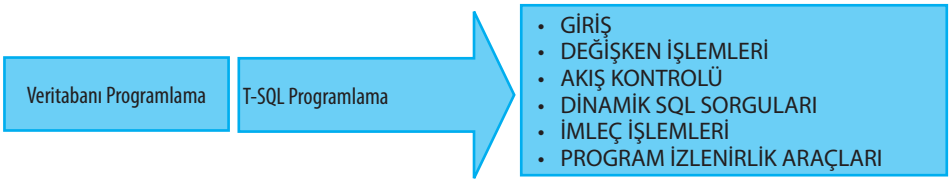
Bu üniteyi tamamladıktan sonra;

- SQL ve T-SQL arası farkı açıklayabilecek,
- T-SQL'de değişken tanımlayabilecek,
- T-SQL akış kontrol komutlarını kullanabilecek,
- T-SQL'de program izlenirlik araçlarını açıklayabilecek bilgi ve becerilere sahip olabileceksiniz.

Anahtar Kavramlar

- Yordamsal Dil
- Veri İşleme Dili
- T-SQL'de Akış Kontrolü
- T-SQL'de Hata Denetimi

İçindekiler



T-SQL Programlama

GİRİŞ

SQL dili, ilişkisel veritabanı yönetim sistemlerinde veri tanımlama, veri işleme ve veri kontrolü gibi farklı amaçlar için kullanılmaktadır. ANSI standartları ile tanımlı olup farklı firmalara ait ilişkisel veritabanı yönetim sistemlerinde kullanım imkanı vardır.

SQL komutları kullanım amaçlarına göre

- Veri Tanımlama Dili (DDL)
- Veri İşleme Dili (DML)
- Veri Kontrol Dili (DCL)

üç genel kategoriye ayrılır. Veri Tanımlama Dili (DDL), verilerin tutulduğu nesneler olan tabloların oluşturulmasını, silinmesini ve bazı temel özelliklerinin düzenlenmesini sağlar. Bu kategoride, yeni bir tablo oluşturmak için CREATE TABLE, tabloda değişiklik yapmak için ALTER TABLE ve tabloyu silmek için DROP TABLE kullanılan bazı yaygın komutlardır. Veri İşleme Dili (DML), veri girmek, değiştirmek, silmek ve verileri almak için kullanılan DML komutlarının tümüdür. Bu kategoride, veri seçmek için SELECT, veri silmek için DELETE, veri güncellemek için UPDATE ve veri girmek için INSERT en sık kullanılan DML komutlarıdır. Veri Kontrol Dili (DCL) veritabanı kullanıcısı veya rolü ile ilgili izinlerin düzenlenmesini sağlar. Bu kategoride, kullanıcıya yetki tanımlama için GRANT, kullanıcı yetkilerini engellemek için DENY ve daha önce yapılmış olan yetki ve izinleri kaldırmak için REVOKE komutları kullanılır.

SQL'deki yukarıdaki temel komutlara yıllar içinde birçok **Yordamsal Dil** (procedural language) bileşeni eklenmekle beraber, SQL gerçek bir programlama dilinin özelliklerine sahip değildir. İşletmeler ile ilgili bazı veritabanı projelerinde ise veritabanı yönetim sistemi içinde akış kontrolü, döngü vb. yordamsal dil özelliklerinin raporlama, analiz vb. işlemler için kullanılması gerekmektedir. Bu durumlarda, Microsoft ve Sysbase tarafından geliştirilen ve bazı yordamsal dil özelliklerini barındıran T-SQL (Transact-SQL) dili geliştirilmiştir. T-SQL ile çeşitli veri tipleri ve fonksiyonlar tanımlanabilmektedir. Ayrıca yukarıda bahsedilen döngüler, akış kontrolü gibi işlemlerde gerçeklenebilir.

Yordamsal dil (procedural language): Hedeflenen çıktıları üretmek üzere belirlenebilecek algoritmaların kodlanabildiği dildir. Değişken tanımlama, akış kontrolü, döngüler vb. özellikleri vardır.

SQL ve T-SQL arasında diğer farklılıkarı araştırınız.

SIRA SİZDE 1

T-SQL, Microsoft SQL Server ve Sybase için kullanılabilecek bir dildir. Bu platformlar dışında geliştirilen veritabanı sistemlerinde veya harici programlama dillerinde kullanılamaz. Örneğin, Oracle veritabanı yönetim sistemleri için yordamsal dil özelliklerine sahip PL/SQL geliştirilmiştir. Microsoft ortamında, T-SQL ifadeleri ile SQL sunucu üzerinden işlem yapabilmek için istemci bir yazılıma ihtiyaç vardır. SQL Server Management Studio

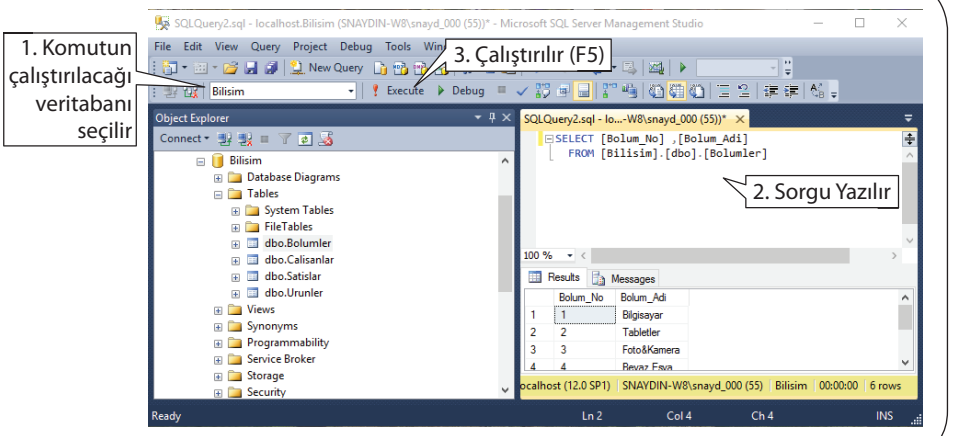
bu amaçla kullanılabilecek bir yazılımdır. Birinci ünitede kurulumu anlatılan SQL Server Management Studio, 2005'ten itibaren MS SQL Server sürümlerini desteklemektedir. Bu yazılım, standart SQL komutlarının kullanılmasına imkan vermektedir. Ayrıca, veritabanı oluşturma, tablo oluşturma, indeks oluşturma vb. birçok işlem kod kullanılmadan Yardımcı (Wizard) görsel arayüzler kullanılarak sağlanabilmektedir. Takip eden kısımlardaki örnek uygulamalar ve ekran çıktıları Bölüm 1'de oluşturulan örnek Bilişim Veritabanı dikkate alınarak verilmektedir. Okuyucu verilen örnek uygulamaları, SQL Server Management Studio ortamında Resim 2.1.' deki gibi Bilisim veritabanı seçili iken, SQL sorgu yazarak çalıştırabilir.

DİKKAT

Bilisim Veritabanının yapısı ve içeriği bazı örnek uygulamalar sonucunda değişebileceği için tekrar Bölüm 1' deki tablo değerlerine döndürülerek aynı sonuçlar alınabilir.

Resim 2.1

Örnek uygulamaları çalıştırma ekranı: Bilişim veritabanı seçili iken, SQL sorgu penceresine kitap içerisindeki örnek kodlar yazılarak çalıştırılabilir.



DEĞİŞKEN İŞLEMLERİ

Değişken bir niceliği (miktarı) ifade etmek için kullanılan, bilgisayar belleğinde belli bir yer kaplayan ve üzerine değer ataması yapılabilen bir yapıdır. T-SQL'de değişken oluşturulabilir ve program içinde kullanılabilir. T-SQL dilinde değişkenler oluşturmak için değişken adı @ simgesi ile başlamalıdır. Bu şekilde tanımlanan değişkenler yerel değişkenler olarak da adlandırılır. Ayrıca sunucu tarafından tanımlanmış ve sunucu hakkında genel bilgileri veren evrensel değişkenler "@@" ön eki ile gösterilmektedir.

@: T-SQL dilinde yerel değişken tanımlamak için kullanılan ön ek, @@: Sistem tarafından bilgi vermek amaçlı oluşturmuş evrensel değişken.

T-SQL' de değişken isimlendirme kuralları diğer diller ile aynı olup bazıları aşağıda verilmektedir.

- Değişkenler Türkçe karakter ve boşluk içermez
- Değişken isimleri ilk karakteri harf ile başlayıp harf, rakam ve alt çizgi (_) ile devam edebilir.
- SQL veya T-SQL için kullanılan komutlar ve ayrılmış sözcükler (SELECT, INSERT, UPDATE, NOT vb.) kullanılmaz.
- Değişken ismi, SQL'de özel anlamı olan sembollerle (@, @@, #, ##, $) başlamamalıdır.
- Değişken isimlerinde küçük veya büyük harf kullanımı fark etmez.

Değişken tanımlama DECLARE komutu ile yapılır. Genel yazım şekli aşağıdaki gibidir.

```
DECLARE @degiskenadi <veri tipi> [(boyut)]
```

Burada "degiskenadi" değişkenin taşıdığı anlama yakın fazla uzun olmamalı ve yukarıdaki isimlendirme kurallarına uygun olmalıdır. Veri tipi ise sayısal değerler için INT veya

karakter türündeki veriler için VARCHAR(boyut) olabilir. Tablo oluşturulması sırasında sütun alanlarında kullanılanabilen veri tipleri değişkenler için de geçerlidir. Aynı satırda değişken tanımlama ve atama yapabilmek için ilk komuttan sonra noktalı virgül(;) kullanılması gerekir.

Örnek Uygulama 2.1: Bu kitabın ilk ünitesinde tanımlanan örnek veritabanındaki "Bölümler" tablosundaki bazı sütünlar için T-SQL'de kullanılabilecek değişken tanımlaması aşağıdaki gibi yapılabilir.

Örnek 2.1'de değişken tanımlama komutları tek bir satırda yazılarak da çalıştırılabilir.
`DECLARE @Bolum_No INT; DECLARE @Bolum_Adi VARCHAR(50).`

```
DECLARE @Bolum_No INT;
DECLARE @Bolum_Adi VARCHAR(50)
```

Bu ünitede yer alan uygulamaların kodlarına https://goo.gl/NXGpdW adresinden ulaşabilirsiniz.

DİKKAT

Tanımlanan değişkenlere SET ya da SELECT ifadeleri kullanılarak değer ataması yapabiliriz. Genel yazım şekli aşağıdaki gibidir.

```
SET @degiskenadi =deger
SELECT @degiskenadi =deger
```

Örnek Uygulama 2.2: Örnek 2.1' de tanımlanan değişkenlere değer atamak için T-SQL'de kod aşağıdaki şekilde yazılır.

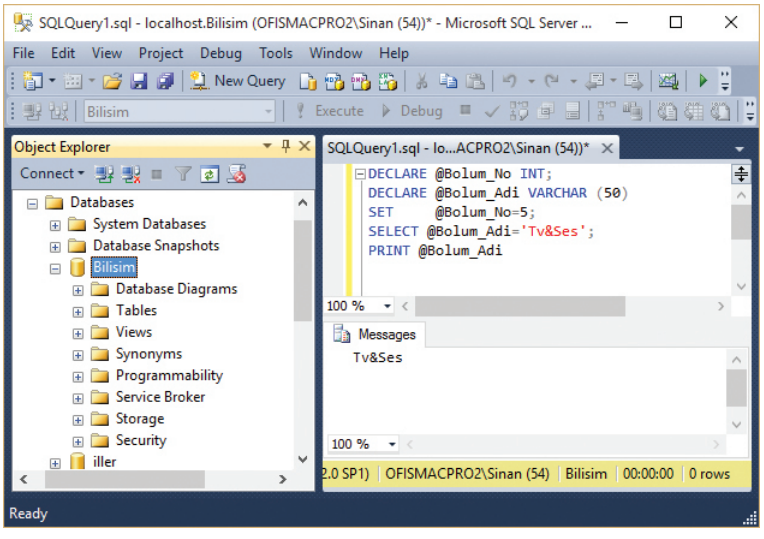
```
DECLARE @Bolum_No INT;
DECLARE @Bolum_Adi VARCHAR (50)
SET @Bolum_No=5;
SET @Bolum_Adi='Tv&Ses';
```

Değişkenlere atanan değerleri ekranda görüntülemek için PRINT (veya SELECT) komutu kullanılabilir. Genel yazım şekli aşağıdaki gibidir.

```
PRINT @degiskenadi
```

Örnek Uygulama 2.3: Örnek 2.2' deki Bolum_Adi değişkeninin değerlerini yazdırmak için T-SQL'de kod aşağıdaki şekilde yazılır ve ilgili sonuç elde edilir.

Resim 2.2



DİKKAT

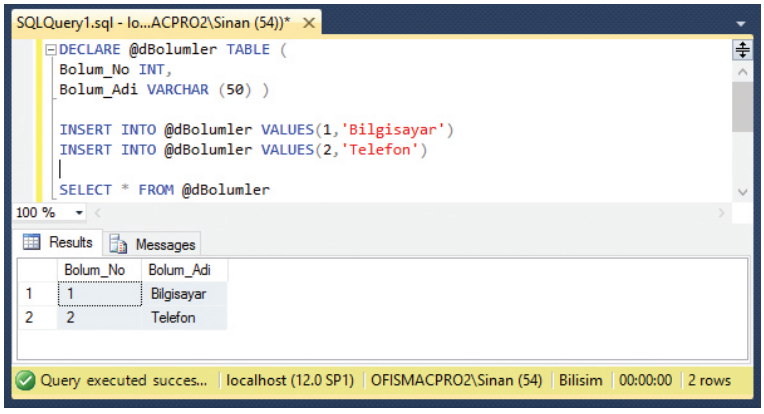
Değişken tipi VARCHAR(metin veri türü) olan bir değişkenle integer(tamsayı veri türü) olan bir değişkeni PRINT ile aynı satırda görüntülemek için değişkenleri birleştirmek gerekir. Bu amaçla tamsayı veri türdeki değişkeni metin türüne dönüştürmek için CAST(@intDegisken as VARCHAR) komutu kullanılabilir. İki ifadeyi yan yana getirmek için + işareti kullanılabilir.

T-SQL programlama dilinde tam sayı, ondalıklı sayı, metin gibi SQL veri türlerinin kullanılabileceği gibi veritabanı yönetim sistemindeki diğer nesneler de değişken olarak tanımlanabilmektedir. Örneğin Tablo tipi verilerin değişkenlere ataması için bir değişken tablo tipi olarak tanımlanabilir.

```
DECLARE @degiskenadi TABLE(
degisken_adi1 veritipi,
degisken_adi2 veritipi )
```

Örnek Uygulama 2.4: Bolumler isminde bir tablo değişkeni oluşturup bu değişkene değerler atayarak tablo değerlerini görüntüleyen T-SQL kodları aşağıdaki gibi oluşturulabilir.

Resim 2.3



DİKKAT

Örnek Uygulama 2.4'te örnekler kendi bilgisayarlarınızda nasıl görüneceğini örneklemek için SQL Server Management Studio ekran görüntüsü şeklinde verilmiştir. Diğer örnekler de kitap yazım formatında gösterilecektir.

AKIŞ KONTROLÜ

Programlama dillerinde kullanılan akış kontrolleri T-SQL içerisinde de yer almaktadır. Akış kontrollerine bağlı işletilecek komut sayısı birden fazla olduğu takdirde BEGIN... END bloğu arasında yazılır. T-SQL içinde yaygın olarak kullanılan akış kontrollerinin bazıları aşağıda örneklenecektir.

IF...ELSE Yapısı

IF ya da Türkçe karşılığı ile "EĞER" koşul yapısı programlamada oldukça yaygın olarak kullanılır. Belirli koşullar sağlandığı takdirde gerçekleştirmesi istenen işlemler için kullanılır. Aşağıda verilen genel yazım kuralı açıklanacak olur ise "Koşul 1 ifadesi" nin sağlanması durumunda bir alt komut ya da komut bloğu çalıştırılarak işlem tamamlanır. Ancak "Koşul 1" doğru değilse bu durumda ELSE IF ile tanımlanan "Koşul İfadesi 2" ifadesine

bakılır ve doğru ise bu komutun alt satırındaki komut ya da komut bloğu çalıştırılır. Eğer IF komutlarının hiçbiri sağlanmıyor ise en son ELSE komutunun alt satırındaki komut ya da komut bloğu çalıştırılır.

```
IF (koşul İfadesi 1)
     Bir SQL ifadesi ya da BEGIN…END bloğu
ELSE IF (Koşul İfadesi 2)
     Bir SQL ifadesi ya da BEGIN…END bloğu
ELSE
     Bir SQL ifadesi ya da BEGIN…END bloğu
```

Örnek Uygulama 2.5: Örnek veritabanımız için "Urunler" tablosunda fiyatı 200 TL'nin altında olan ürünler varsa sayısı, yok ise "yoktur" ifadesi oluşturan T-SQL kodu ve ekran çıktısı aşağıdaki gibi oluşturulabilir.

```
DECLARE @Urun_Sayisi VARCHAR(10)
SELECT @Urun_Sayisi=COUNT(*) FROM Urunler WHERE (Urun_Fiyati<200)
IF (@Urun_Sayisi>0)
    PRINT 'Fiyati 200TL den az '+@Urun_Sayisi+'  ürün vardır'
ELSE
    PRINT 'Fiyati 200TL den az ürün yoktur'
Sonuç:
Fiyati 200TL den az 4 ürün vardır
```

CASE Yapısı

Case komutu, birden fazla koşulun kontrol edilmesi gerektiğinde kolay kullanım sağlayan bir akış kontrolü komutudur. Case yapısı SQL sorgu ifadelerinin içinde de koşul amacıyla kullanılmaktadır. T-SQL' de bu yapının iki farklı kullanımı aşağıda tanımlanmıştır.

Kullanım 1

```
CASE Değer
     WHEN Değer ya da aralık THEN Sonuç_ifadesi_1
        WHEN Değer ya da aralık THEN Sonuç_ifadesi_2
        ...
     ELSE Sonuç_ifadesi_n
END
```

Kullanım 2

```
CASE
     WHEN Koşul_1 THEN Sonuç_ifadesi_1
        WHEN Koşul_2 THEN Sonuç_ifadesi_2
        ...
     ELSE Sonuç_ifadesi_n
END
```

Case ifadesi yordamsal programların akış mantığı ile yukarıdan aşağıya akışı gerçekleştirerek sonuç elde edecektir. İlk doğru ifade bulununca ilgili satırdaki değer ya da ifade çalıştırılacaktır.

Örnek Uygulama 2.6: Örnek veritabanındaki "Calisanlar" tablosundaki kayıtlar için aşağıdaki case ifadesi oluşturulabilir. Bu örnekte case ifadesi seçme sorgusunun bir alanı olarak her satır için çalışacağına dikkat ediniz.

```
SELECT Adi, Cinsiyet=
    CASE Cinsiyet
        WHEN 'E' THEN 'Erkek'
        WHEN 'K' THEN 'Bayan'
    END
FROM Calisanlar
```

Sonuç:

Adi	Cinsiyet
Ziya Doğan	Erkek
Ayşe Saygı	Bayan
Ali Yılmaz	Erkek
Fatma Doğan	Bayan
Hasan Çiçek	Erkek
Kader Kara	Bayan

SIRA SİZDE 2

Örnek veritabanı için, "Urunler" tablosunu kullanarak "Notebook" ürününün satış miktarı 4 adetten fazla ise satış miktarını yazıp tebrik eden, 3-4 arası ise "satışlara dikkat edelim", 3 adetten az ise yetersiz satış yazan T-SQL komutlarını yazınız.

WHILE Yapısı

Programlama dillerinde döngüler, tekrar gerektiren işlemler için kullanılmaktadır. WHILE döngüsü birçok programlama dilinde olduğu gibi verilen bir koşulun sağlanması durumunda belirlenen komut bloğununu tekrar eden bir yapıdır. Genel yazım şekli aşağıdaki gibidir.

```
WHILE koşul_ifadesi
BEGIN
    Tekrarlanacak işlemler
END
```

Örnek Uygulama 2.7: Birden 100'e kadar sayıların toplamını hesaplayarak görüntüleyen T-SQL kodu aşağıdaki gibi yazılabilir.

```
DECLARE @sayac INT
DECLARE @toplam INT
SET @sayac=1
SET @toplam=0

WHILE (@sayac<=100)
    BEGIN
        Set @toplam=@toplam+@sayac
        Set @sayac=@sayac+1
    END
Print @sayac
Print @toplam

Sonuç:
101
5050
```

While yapısı içinde CONTINUE ve BREAK komutları bir sonraki adıma geçme ve döngüden çıkmak için kullanılabilir. CONTINUE komutundan sonra gelen ifadeler göz ardı edilerek bir sonraki adımdan döngü devam eder. BREAK komutu ise While yapısı içinde döngüden çıkmayı sağlar.

Örnek Uygulama 2.8: Örnek uygulama 2.7'de yer alan T-SQL kodunun, CONTINUE ve BREAK komutu ile nasıl çalıştığına dikkat ediniz.

```
DECLARE @sayac INT
DECLARE @toplam INT
SET @sayac=1
SET @toplam=0

WHILE (@sayac<=100)
    BEGIN
        Set @toplam=@toplam+@sayac
        Set @sayac=@sayac+1
            If @toplam >3000
                   BREAK
                   ELSE
                   CONTINUE
    END
Print @sayac
Print @Toplam

Sonuç:
78
3003
```

GOTO Yapısı

GOTO komutu ile kod içerisinde belirlenen bir etikete direkt geçiş yapılıp bu etiketten sonra devam edilir. Komut sade kullanılabileceği gibi IF veya WHILE yapısı ile farklı amaçlar için de kullanılabilir. Genel yazım şekli aşağıdaki gibidir.

```
Etiket_adı:
......
Komutlar
......
GOTO Etiket_adı
```

Örnek Uygulama 2.9: Örnek uygulama 2.7'de yapılan örneği GOTO ile yapan T-SQL kod ve ekran çıktısı aşağıdadır.

```
DECLARE @sayac INT
DECLARE @toplam INT
SET @sayac=1
SET @toplam=0
Basla:
        Set @toplam=@toplam+@sayac
        Set @sayac=@sayac+1
If  (@sayac<=100)
GOTO basla

Print @sayac
Print @Toplam
```

Örnek veritabanı için, "Urunler" tablosundaki ilk 5 ürün için toplam satış miktarlarını hesaplayarak yazdıran bir T-SQL sorgu yazınız?

SIRA SİZDE 3

DİNAMİK SQL SORGULARI

SQL sorgularını esnek bir şekilde T-SQL programlama ortamında kullanmak mümkündür. Bunun için çalıştırılması planlanan SQL sorgu komutunun bir değişkene aktarılarak çalıştırılması sağlanır. Bu yöntemin temel amacı program kodları tarafından bir sorgunun hazırlanarak çalıştırılmasına ve sonucunun alınmasına olanak sağlamaktadır. "Dinamik SQL Sorguları" ismi de verilen bu yöntemde hazırlanan sorgu değişkenleri "EXECUTE" fonksiyonu ile çalıştırılabilirler. Bu tip sorgular, bir sorgunun diğer sorgu sonucuna göre oluşturulması veya kullanıcının atayacağı değişkenlere göre veri oluşturmak için kullanılabilir. Bir sonraki bölümde saklı yordam kullanımında bu sorguların kullanımı daha iyi anlaşılacaktır.

Örnek Uygulama 2.10. Örnek veritabanı için tablo ismini değişkenden alan dinamik T-SQL kod ve ekran çıktısı aşağıdadır.

```
DECLARE @tabloAdi VARCHAR(50)
SET     @tabloAdi='Bolumler'
DECLARE @Sorgu VARCHAR(50)
SET     @Sorgu='SELECT * FROM '+@tabloAdi
EXECUTE(@Sorgu)
```

Sonuç:

Bolum_No	Bolum_Adi
1	Bilgisayar
2	Tabletler
3	Foto&Kamera
4	Beyaz Eşya
5	Tv&Ses
6	Telefon

İMLEÇ İŞLEMLERİ

Veritabanı Yönetim Sisteminde gerçekleştirilen seçme sorguları; verilerin süzülmesi, hesaplanması ya da türetilmesi gibi işlemleri gerçekleştirmek için kullanılır. Ancak bazı özel durumlarda kullanıcıların, her birini izleyen satırlardaki verileri kullanarak işlem yapması gerekebilmektedir. Kullanıcıların belirleyecekleri bir veri kümesinde her bir satırda birer birer ilerlemelerini sağlayan **İMLEÇ (Cursor)** yapısı kullanılmaktadır. İşleyiş bakımından performansı olumsuz etkileyen bu yapı zorunlu durumlarda tercih edilmelidir.

İmleç (cursor,) veri kümelerinin satırları arasında birer birer ilerlemeyi sağlayan programlama yapılarıdır.

İmleç yapısının çalışma prensibini aşağıdaki örnek ile anlamak daha kolay olacaktır. Örnek veritabanındaki "Calisanlar" tablosundaki "Adi" alanı alfabetik olarak bir metin içerisinde oluşturmak istensin. Bu durumda yazılması gereken T-SQL kodu aşağıdaki gibi olacaktır. Komut açıklamalarına dikkat ederek inceleyiniz.

```
--Okunacak değişkenler tanımlanır.
DECLARE @AD nvarchar(100);DECLARE @CINS nvarchar(5);DECLARE @METIN
nvarchar(max)
--İmleç veri kümesi için tanımlanır.
DECLARE Calısan_cursor CURSOR FOR
SELECT Adi, Cinsiyet From Calisanlar Order By Adi
--İmleç açılır
Open Calısan_cursor
-- ilk kayıt talep edilen veri kümesi sırası ile okunur.
--(Ad ve Cins değişkeni sorgu ile aynı sırada olmalı )
```

```
   FETCH NEXT FROM Calısan_cursor INTO @AD, @CINS
   SET @METIN= 'Firmamızda '
   SET @METIN=@METIN + @AD +'('+ @CINS+')'

   FETCH NEXT FROM Calısan_cursor INTO @AD, @CINS-- veri kümesi sırası
ile okunur
   --While döngüsü ile son kayda gelinceye kadar birer birer ilerleme
kurgulanır.
   WHILE @@FETCH_STATUS = 0
   Begin
      SET @METIN=@METIN +', '+  @AD +'('+ @CINS+')' -- Her bir veri metne
eklenir.
      FETCH NEXT FROM Calısan_cursor INTO @AD, @CINS --Soraki kayda iler-
lenir.
   END
   SET @METIN= @METIN +' adlı çalışanlar bulunmaktadır.'
   Select @METIN
   --İmleç kapanarak silinir.
   Close Calısan_cursor
   deallocate Calısan_cursor
```

Sonuç:

Firmamızda Ali Yılmaz(E), Ayşe Saygı(K), Fatma Doğan(K), Hasan Çiçek(E), Kader Kara(K), Ziya Doğan(E) adlı çalışanlar bulunmaktadır.

Bu ünitede yer alan uygulamaların kodlarına https://goo.gl/NXGpdW adresinden ulaşabilirsiniz.

İNTERNET

Örnek Uygulama 2.11: Örnek veritabanındaki Urunler tablosundaki tüm ürünlerin fiyatını imleç kullanarak %10 arttıran T-SQL kodu aşağıdaki gibi oluşturulabilir.

```
   DECLARE @urunFiyati decimal(18,2)
   DECLARE FiyatArtimiImleci CURSOR FOR SELECT Urun_Fiyati FROM Urunler
   OPEN FiyatArtimiImleci
   FETCH NEXT FROM FiyatArtimiImleci INTO @urunFiyati
   WHILE @@FETCH_STATUS=0
   BEGIN
       UPDATE Urunler
       SET Urun_Fiyati=Urun_Fiyati+(Urun_Fiyati*0.1)
       WHERE CURRENT OF FiyatArtimiImleci
       FETCH NEXT FROM FiyatArtimiImleci INTO @urunFiyati
   END

   CLOSE FiyatArtimiImleci
   DEALLOCATE FiyatArtimiImleci
   SELECT * FROM Urunler
```

İmleç işlemlerindeki `FETCH` komutu kayıtlar arasında gezinmeyi sağlar, `FETCH NEXT` bir sonraki, `FETCH PRIOR` bir önceki, `FETCH LAST` son kayda ve `FETCH FIRST` ilk kayda ilerlemeyi sağlar.

Bu kod, SQL sorgu penceresinde çalıştırıldıktan sonra Tablo 1.3 ile verilen Urunler tablosundaki Urun_Fiyati alanında %10 artış olur. Urunler tablosunun imleç çalıştırıldıktan sonraki hâli aşağıdaki gibi gözlemlenebilir.

Urun_ No	Urun_Adi	Urun_ Sayisi	Urun_ Fiyati	Bolum_ No
1	Notebook	3	2530.00	1
2	Ultrabook	5	3300.00	1
3	USB Bellek	7	22.00	1
4	Harici Disk	4	198.00	1
……	……..	…….	……..	……
18	Apple Telefon	10	3850.00	6
19	Nokia Telefon	0	770.00	6

PROGRAM İZLENİRLİK ARAÇLARI

T-SQL'de yazılan programın daha sonra tekrar okunabilmesine veya çalışma zamanında olası hata önleyici veya denetimine yönelik araçlar da bulunmaktadır. Bu araçları kullanmak çevrim dışı ve çevrim içi program izlenirliğini ve güvenirliğini artıracaktır.

Açıklama Ekleme

T-SQL'de açıklama eklemek için "--" veya "/*...*/" ifadeleri kullanılır. Tek satırlık açıklama eklemek için açıklama satırının önüne "--" ifadesi konur. Bir veya daha fazla satırı açıklama satırı olarak göstermek için açıklama satırının başına "/*" işareti ve açıklama satırının sonuna "*/" işareti eklenir. Açıklamalar kodların takip edilirliği için önemlidir.

Örnek Uygulama 2.12: Örnek Uygulama 2.11'de T-SQL koduyla hangi adımda neler yapıldığını ifade etmek üzere açıklamalar bulunmaktadır.

```
--IMLEC Orneği
/*Ürün fiyati değişkeni tanımlanır*/
DECLARE @urunFiyati decimal(18,2)
/*Imlec tanımlanır*/
DECLARE FiyatArtimiImleci CURSOR FOR SELECT Urun_Fiyati FROM Urunler
/*Imlec Acılır*/
OPEN FiyatArtimiImleci
/*ilk okuma yapılır*/
FETCH NEXT FROM FiyatArtimiImleci INTO @urunFiyati
--dongü ile son satır bitim kontrolü yapılır.
WHILE @@FETCH_STATUS=0
BEGIN
    --VT de fiyat %10 artırılrak güncellenir.
    UPDATE Urunler
    SET Urun_Fiyati=Urun_Fiyati+(Urun_Fiyati*0.1)
    WHERE CURRENT OF FiyatArtimiImleci
    --Bir sonraki adim için okuma yapilir.
    FETCH NEXT FROM FiyatArtimiImleci INTO @urunFiyati
END
--Imlec kapatılır.
CLOSE FiyatArtimiImleci
--İlgili kayıtlar silinir.
DEALLOCATE FiyatArtimiImleci
```

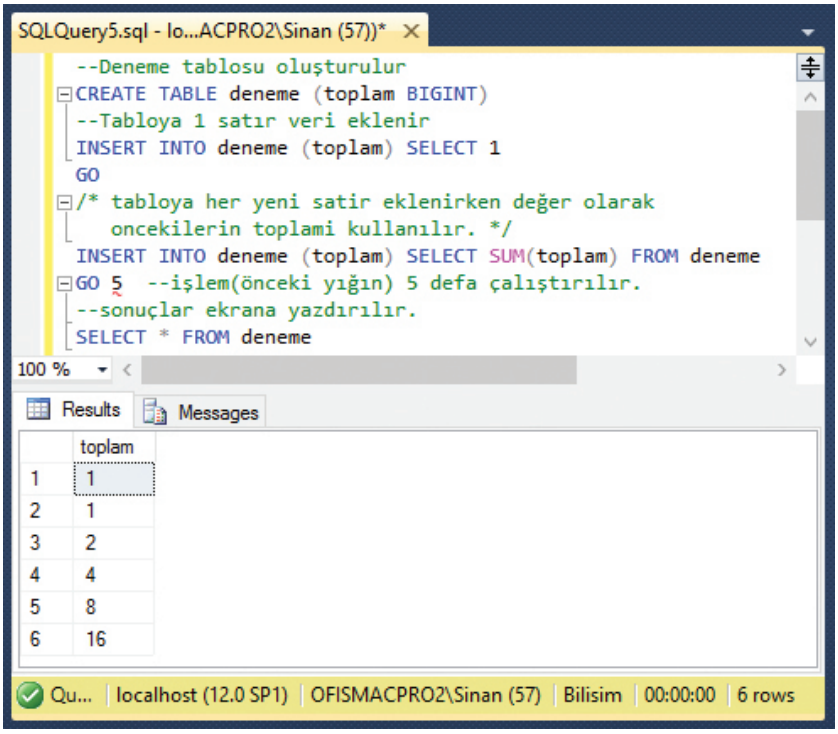
Görüldüğü üzere yeni hâli ile kod daha izlenebilir durumdadır.

Yığın işlemi

GO Komutu ile SQL sorgularının sırasıyla yığınlar olarak çalıştırılması sağlanır. GO komutu bir T-SQL ifadesi olmayıp yığının son satırında kullanılarak yığının sonlandığını ifade eder. GO komutu GO [sayi] şeklinde yazılarak en son yığının yazılan sayı kadar çalıştırılması sağlanır. Böylece aynı komut tekrar tekrar çalıştırılarak bir döngü elde edilir.

Örnek Uygulama 2.13: "Deneme" adında bir tablo oluşturan ve tabloya 5 kere veri ekleme işlemini gerçekleştirip sonuçları ekrana yazdıran SQL kodu ve ekran çıktısı Resim 2.4'tedir.

Resim 2.4



Veritabanı Veri İşlem Değişikliklerinin Kaydedilmesi

Veri İşleme Dili kullanımı sonucunda değişiklik olan kayıtları listelemek için OUTPUT komutu kullanılır. Komutun genel kullanım şekli aşağıdadır.

OUTPUT OP.Listenecek_Alan1,...OP.Listenecek_Alann, INTO Kayıt_Tablo_Adı

Burada OP yerine INSERTED veya DELETED kullanılıp noktadan sonra alan isimlerinin yazılması gerekir. Kayıt_Tablo_Adı, etkilenen verilerin tutulduğu tabloyu veya tablo değişkenini tutar.

Örnek Uygulama 2.14: Bölüm 1'de Tablo 1.2 ile verilen calısanlar tablosuna yeni bir kayıt ekleyip ve bu yeni kaydın bilgilerinin OUTPUT komutu ile bir aktarilantabloEkleme tablo değişkenine aktaran kod ve çıktısı aşağıdadır.

```
DECLARE @aktarilantabloEkleme TABLE(
    TC_no varchar(11),
    Adi nvarchar(100),
    Bolum_No int,
    Cinsiyet nchar(1))
```

```
INSERT INTO Calisanlar
     OUTPUT inserted.TC_no,inserted.Adi, inserted.Bolum_No,inserted.
Cinsiyet
      INTO @aktarilantabloEkleme
VALUES ('1245789635','Mehmet Yılmaz', 1,'E')

SELECT * FROM @aktarilantabloEkleme
```

Sonuç:

TC_no	Adi	Bolum_No	Cinsiyet
1245789635	Mehmet Yılmaz	1	E

Örnek Uygulama 2.15: Örnek veritabanında "Satislar" tablosunda fiyatı 2000 TL'den az olan ürünleri silen ve silinen ürünleri OUTPUT komutu ile bir tablo değişkenine aktaran T-SQL kod ve sonuç aşağıdadır.

```
DECLARE @KayitSilme TABLE(
    Urun_Fiyati decimal(18,2),
    Urun_No INT
)

DELETE FROM Satislar
    OUTPUT deleted.Fiyat,deleted.Urun_No
    INTO @KayitSilme
WHERE Fiyat<2000

SELECT * FROM @KayitSilme
```

Sonuç:

Urun_Fiyati	Urun_No
25.00	3
25.00	3
75.00	5
150.00	8
1500.00	15
850.00	19
28.00	3

Hata Denetimi

Veritabanı yönetim sisteminde hata algılama ve işlem yapmaya yönelik farklı özellikler bulunmaktadır.

T-SQL'de oluşan hata mesajları ve kodları "sys.messages" adlı sistem tablosunda tutulmaktadır. Hata mesajlarının bir kısmı Tablo 2.1'de verilmiş olup SQL sorgu penceresinde "SELECT * FROM sys.messages" ile hepsi listelenebilir.

Message _id	Language _id	severity	is_event _logged	text
21	1033	20	0	Warning: Fatal error %d occurred at %S_DATE. Note the error and time, and contact your system administrator.
101	1033	15	0	Query not allowed in Waitfor.
102	1033	15	0	Incorrect syntax near '%.*ls'.
103	1033	15	0	The %S_MSG that starts with '%.*ls' is too long. Maximum length is %d.
104	1033	15	0	ORDER BY items must appear in the select list if the statement contains a UNION, INTERSECT or EXCEPT operator.
105	1033	15	0	Unclosed quotation mark after the character string '%.*ls'.
...				

Tablo 2.1
sys.messages içindeki bazı hata ve uyarı listesi

"@@ERROR" sistem fonksiyonunda ise en son hata mesajının kodu tutulmaktadır. Bu hata kodu, "sys.messages" tablosundan da gösterilebilir. Alt alta birden fazla sorgu yazıp en sonda hata kontrolü yapıldığı takdirde yalnızca son SQL sorgusundan kaynaklı hatalar yakalanabilir. Bu nedenle yazılan her sorgu ifadesinden sonra hata kontrolü yapılmalıdır.

Eğer hata denetimi yapılması ve gerektiğinde bir işlem yapılması gerekiyorsa TRY... CATCH komutu, diğer programlama dillerinde de olduğu gibi kullanılabilir. TRY bloğunda hata oluştuğu takdirde CATCH bloğuna geçilir. Hata CATCH bloğu tarafından yakalanır. Hata oluşmaz ise CATCH bloğu devreye girmez. Genel kullanım şekli aşağıdaki gibidir.

```
BEGIN TRY
    SQL_Komutları
END TRY
BEGIN CATCH
    SQL_Komutları
END CATCH
```

TRY..CATCH komutununda CATCH bloğu içinde kendi değerlerini tutan CATCH bloğu dışında ise NULL değerler döndüren çeşitli fonksiyonları vardır. Bu fonksiyonlar Tablo 2.2'de tanımlanmaktadır.

FONKSİYON	İŞLEVİ
ERROR_NUMBER()	Oluşan hata numarasını döndürür.
ERROR_SEVERITY()	Hatanın önem düzeyini döndürür.
ERROR_STATE()	Hatanın durum numarasını döndürür.
ERROR_PROCEDURE()	Hataya neden olan saklı yordam (stored procedure) veya tetikleyici (trigger) adını döndürür.
ERROR_LINE()	Hatanın oluştuğu satır numarasını döndürür.
ERROR_MESSAGE()	Hatayı açıklayan tam metin mesajı döndürür.

Tablo 2.2
CATH bloğu için hata bilgisi döndüren fonksiyonlar

Tablo 2.2'deki ERROR_SEVERITY hata önem düzeyleri 0-10 arası değer alırsa bu uyarı ve gayriresmî hatalar olup CATCH tarafından yakalanmaz. 20 ve üstü değer alırsa kritik hata anlamı taşıyıp eğer veritabanı sunucusu bağlantısı problemli değilse CATCH tarafından yakalanır. Bağlantı problemi varsa yakalanmaz.

Örnek Uygulama 2.16: Sorgu içine 0'a bölme hatasını yakalayan ve gösteren T-SQL kod ve sonuç aşağıdadır.

```
BEGIN TRY
    SELECT 4/0
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS Hata_Numarasi,
        ERROR_SEVERITY() AS Hata_Duzeyi,
        ERROR_STATE() AS Hata_Durum_No,
        ERROR_LINE() AS Hata_Satir_No,
        ERROR_MESSAGE() AS Hata_Mesaj
END CATCH
```

Sonuç:

Hata_ Numarasi	Hata_ Duzeyi	Hata_ Durum_No	Hata_ Satir_No	Hata_Mesaj
8134	16	1	3	Divide by zero error encountered.

Örnek Uygulama 2.17: Bölüm 1'deki bilisim veritabanında olmayan bir tabloyu silmeye çalışırken oluşan hatayı yakalayan ve gösteren T-SQL kod ve sonuç aşağıdadır.

```
BEGIN TRY
    DROP TABLE DenemeTablosu
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS Hata_Numarasi,
        ERROR_SEVERITY() AS Hata_Duzeyi,
        ERROR_STATE() AS Hata_Durum_No,
        ERROR_LINE() AS Hata_Satir_No,
        ERROR_MESSAGE() AS Hata_Mesaj
END CATCH
```

Sonuç:

Hata_ Numarasi	Hata_ Duzeyi	Hata_ Durum_No	Hata_ Satir_No	Hata_Mesaj
3701	11	5	4	Cannot drop the table 'DenemeTablosu', because it does not exist or you do not have permission.

SIRA SİZDE 4

Örnek veritabanı için, "Urunler" tablosundaki 'Notebook' kaydını silme sırasında varsa hatayı yakalayan T-SQL sorgu yazınız.

Özet

SQL ve T-SQL arası farkı açıklamak.

SQL, ANSI standartlarında ilişkisel veritabanı yönetim sistemleri için geliştirilmiş açık standartlı bir dildir. Belli veri kümesini girdi olarak alıp başka bir veri kümesi üretir. Ara işlemler için yordamsal dil özelliği zayıftır veya yoktur. T-SQL ise değişken tanımlama, akış kontrolü, döngü vb. yordamsal dil özelliklerine sahiptir. Fakat T-SQL açık standartlı olmayıp Microsoft ve Sysbase veritabanı yönetim sistemleri için özelleşmiştir.

T-SQL'de değişken tanımlamak.

SQL sorgu komutlarında genellikle değişken kullanmaya ihtiyaç olmamaktadır. Ancak T-SQL program dilinde değişken tanımlamak diğer programlama dilleri kadar elzemdir. T-SQL programlama dilinde değişkenler @ karakteri ile başlar. "DECLATE @DegişkenAdı Veritürü" ifadesi değişken tanımlamak için genel yazım kuralıdır.

T-SQL akış kontrol komutlarını kullanmak.

Veritabanı programcılarının en önemli görevlerinden birisi hazır veritabanı için istenilen sonuçları üretmeye yönelik sorgular hazırlamaktır. Kullanıcı girdilerine bağlı yordamsal dil özelliği gerektiren sorgular ise TSQL' de; IF...ELSE yapısı, Case yapısı, WHILE yapısı, imleç işlemleri ve dinamik SQL gibi yapılar ile kolayca hazırlanabilmektedir.

T-SQL'de program izlenirlik araçlarını açıklamak.

Yazılan sorguların çevrim içi ve çevrim dışı izlenebilirliği için TSQL'in sağladığı bazı araçlar vardır. Yazılan programların kullanıcılar tarafından okunabilmesi için açıklamalar önemli bir yere sahiptir. Bunun yanı sıra çalışma zamanında programın yaptığı işleri izleme için kod yığınlarını sırası ile çalıştırma, programın etkilediği verilerin kaydedilmesi, hata ayıklama vb. gibi birçok ara işlemde SQL tarafından sağlanan GO yapısı, OUTPUT yapısı, TRY...CATCH vb. ile mümkündür.

Kendimizi Sınayalım

1. SQL Server ile ilgili aşağıdaki ifadelerden hangisi **yanlıştır?**

a. SQL'de veri kontrolüne yönelik komutlar vardır.
b. SQL standartlarında akış kontrolüne yönelik komut yoktur.
c. Veri işleme dili özellikleri sadece T-SQL de bulunmaktadır.
d. SQL'de yerel değişken tanımlanamaz.
e. PL/SQL Oracle tabanlı sistemler için özelleşmiştir.

2. SQL Komutlari ile ilgili olarak aşağıdakilerden hangisi **yanlıştır?**

a. GRANT komutu standart SQL ANSI standartları içinde yer almaktadır.
b. SQL Server Management Studio Oracle için de kullanılabilir.
c. SQL Server Management Studio içinde tablo oluşturma SQL kod ve yardımcı görsel arayüz ile yapılabilir.
d. T-SQL›de yerel değişken tanımlanabilir.
e. T-SQL Sysbase'i destekler.

3. Aşağıdaki ifadelerden hangisi T-SQL'de hata ile **sonuçlanmaz?**

a. DECLARE @UrunNo INT; UrunNo=5;
b. DECLARE @UrunNo INT; SET UrunNo=5;
c. DECLARE @UrunNo INT, SET @urunno=5;
d. DECLARE @UrunNo INT;SET @urunno=5;
e. DECLARE @UrunNo INT; @UrunNo=5;

4. T-SQL'de "DECLARE @islemNO INT; DECLARE @islemAdi VARCHAR (50);" ile iki değişken tanımlanmış olsun. Bu değişkenlere uygun değerler atandığını varsayarsak aşağıdakilerden hangisi T-SQL'de **hata üretmez?**

a. PRINT @islemAdi+' işlem numarası :'+CAST(@islemNO as VARCHAR)
b. PRINT @islemAdi+' işlem numarası :'+@islemNO
c. PRINT @islemAdi & ' işlem numarası :'& CAST(@islemNO as VARCHAR)
d. PRINT @islemAdi &' işlem numarası :' & @islemNO
e. PRINT @islemAdi+' işlem numarası :'+@islemNO as VARCHAR

5.

```
DECLARE @Urun_Miktar  VARCHAR(10)
SELECT @Urun_Miktar=COUNT(*) FROM Satislar
WHERE (Miktar>2)

IF (@Urun_Miktar>0)
    PRINT 'Ürün miktari =' +@Urun_Miktar
ELSE
    PRINT
        'Ürün yoktur'
```

Örnek veritabanındaki Satıslar tablosu için yukarıdaki sorgusunun sonucu aşağıdakilerden hangisidir?

a. Ürün yoktur
b. Urun miktari =1
c. Urun miktari =2
d. Urun miktari =3
e. Urun miktari =4

6.

```
DECLARE @no_Urun INT;
SET     @no_Urun=2
WHILE (@no_Urun<=10)
    BEGIN
       SELECT Urn.Urun_Adi
        FROM urunler Urn
        WHERE Urn.Urun_No=@no_Urun

        SET @no_Urun=@no_Urun+3
    END
```

Kitabın giriş kısmındaki örnek veritabanındaki Ürünler tablosu için yukarıdaki sorgunun sonucu aşağıdakilerden hangisidir?

a. Ultrabook, İşlemci, Windows Tablet
b. USB Bellek, İşlemci, Windows Tablet
c. Notebook, Ultrabook, USB Bellek
d. Ultrabook, USB Bellek, Harici Disk
e. Ultrabook, USB Bellek, Harici Disk, İşlemci

7.

```
DECLARE @tabloAdi VARCHAR(50)
SET     @tabloAdi='Calisanlar'
DECLARE @Sorgu VARCHAR(50)
SET     @Sorgu='SELECT COUNT(*) FROM '+@tabloAdi
EXECUTE(@Sorgu)
```

Kitabın giriş kısmındaki örnek veritabanı için yukarıdaki sorgusunun sonucu aşağıdakilerden hangisidir?

a. 3

b.

Ziya Doğan
Ayşe Saygı
Ali Yılmaz
Fatma Doğan
Hasan Çiçek
Kader Kara

c.

Ali Yılmaz
Fatma Doğan

d. 6

e. 2

8. T-SQL'de açıklama eklemek ile ilgili aşağıdaki ifadelerden hangisi doğrudur?

a. Açıklama '/' ile başlar '*' ile biter.

b. Tek satırlık açıklamalar '--' ile başlar..

c. Açıklamalar '%' ile başlar

d. Tek satırlık açıklamalar '//' ile başlar

e. Açıklamalar '**' ile başlar

9.

```
CREATE TABLE Deney (toplam BIGINT)
INSERT INTO Deney (toplam) SELECT 2
GO
INSERT INTO Deney (toplam) SELECT SUM(toplam)
FROM Deney
GO 2
SELECT COUNT(*) FROM Deney
```

Yukarıdaki sorgunun sonucu hangisidir?

a.

2
2

b. 2

c. 3

d.

2
2
4

e. 4

10. Sistem fonksiyonu olan '@@ERROR' ile ilgili aşağıdaki ifadelerden hangisi doğrudur?

a. Bir T-SQL kodundaki oluşan tüm hataları aynı aynda verir.

b. ERROR_SEVERITY () ile aynı sonucu döndürür.

c. ERROR_STATE() ile aynı sonucu döndürür.

d. "@@ERROR' ile TRY...CATCH birbiri yerine kullanılabilir.

e. Sadece son SQL kodu ile oluşan hatayı döndürür.

Kendimizi Sınayalım Yanıt Anahtarı

1. c	Yanıtınız yanlış ise "T-SQL Programlama Giriş" konusunu yeniden gözden geçiriniz.
2. b	Yanıtınız yanlış ise " T-SQL Programlama Giriş " konusunu yeniden gözden geçiriniz.
3. d	Yanıtınız yanlış ise "Değişken İşlemleri" konusunu yeniden gözden geçiriniz.
4. a	Yanıtınız yanlış ise "Değişken İşlemleri" konusunu yeniden gözden geçiriniz.
5. c	Yanıtınız yanlış ise "Akış Kontrolü, IF ELSE Yapısı" konusunu yeniden gözden geçiriniz.
6. a	Yanıtınız yanlış ise "Akış Kontrolü, WHILE Yapısı" konusunu yeniden gözden geçiriniz.
7. d	Yanıtınız yanlış ise "Dinamik SQL Sorguları" konusunu yeniden gözden geçiriniz.
8. b	Yanıtınız yanlış ise "Program İzlenirlik Araçları, Açıklama Ekleme" konusunu yeniden gözden geçiriniz.
9. c	Yanıtınız yanlış ise "Program İzlenirlik Araçları, Yığın işlemi" konusunu yeniden gözden geçiriniz.
10. e	Yanıtınız yanlış ise "Program İzlenirlik Araçları, Hata Denetimi" konusunu yeniden gözden geçiriniz.

Sıra Sizde Yanıt Anahtarı

Sıra Sizde 1

i) SQL açık formatlı, T-SQL ise bir kuruluşa aittir.

ii) T-SQL DELETE ve UPDATE için farklı uygulamalara sahiptir.

iii) SQL tüm VTYS için kullanılabilirken T-SQL 2005 sonrası Microsoft SQL sunucular için kullanılabilir.

Sıra Sizde 2

```
DECLARE @Satis_Sayisi  VARCHAR(10)
SELECT @Satis_Sayisi=SUM(Sat.Miktar)
FROM Urunler Ur, Satislar Sat
WHERE  (Ur.Urun_Adi='Notebook'AND  Ur.Urun_No=Sat.
Urun_No)
IF (@Satis_Sayisi>4)
     PRINT 'Tebrikler Notebook satıs sayısı: '+@
Satis_Sayisi
ELSE IF (@Satis_Sayisi>2) AND (@Satis_Sayisi<=4)
    PRINT 'Satışlara dikkat edelim'
ELSE
    PRINT 'Yetersiz satis'
```

Sıra Sizde 3

```
DECLARE @UrunNo INT
DECLARE @SatisMikt INT
SET     @UrunNo=1
WHILE (@UrunNo<=5)
 BEGIN
   SELECT @SatisMikt=SUM(Sat.Miktar)
       FROM Satislar Sat
        WHERE  Sat.Urun_No=@UrunNo
        IF @SatisMikt IS NULL
         SET @SatisMikt=0
        PRINT 'Urun No:'+cast (@UrunNo as Varc-
har(10))+
                      ' için SatiŞ Miktarı: '+cast
(@SatisMikt as Varchar(10))
    SET @UrunNo=@UrunNo+1
 END
```

Sıra Sizde 4

```
USE Bilisim
BEGIN TRY
   DELETE FROM Urunler WHERE Urun_Adi='Notebook'END
TRY

BEGIN CATCH
```

```
    SELECT
        ERROR_NUMBER() AS Hata_Numarasi,
        ERROR_SEVERITY() AS Hata_Duzeyi,
        ERROR_STATE() AS Hata_Durum_No,
        ERROR_LINE() AS Hata_Satir_No,
        ERROR_MESSAGE() AS Hata_Mesaj
END CATCH
```

Yararlanılan ve Başvurulabilecek Kaynaklar

Özseven, T. (2014).**Veri Tabanı Yönetim Sistemleri 2,** Trabzon, Türkiye.

Elmasri, R., Navathe, S.B., (2011). **Fundamentals of Database Systems,** Sixth Edition, Addison-Wesley, USA.

Ullman, J., Widom, J., (2001). **A first course in Database Systems,** 2nd edition, Prentice Hall.

Yarımağan, Ü (2010). **Veri Tabanı Yönetim Sistemleri,** Akademi Yayıncılık.

Transact-SQL Reference: https://msdn.microsoft.com/en-us/library/bb510741.aspx, son erişim tarihi: 12.11.2015