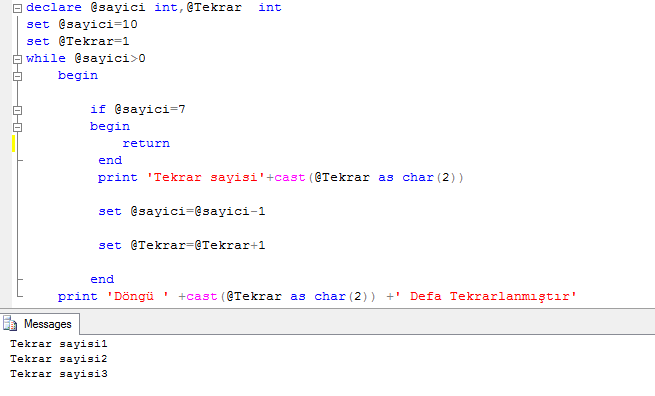
**VERİTABANI –II**

# 4.TRANSACT SQL (T-SQL)

## 4.1 T-SQL AKIŞ KONTROL İFADELERİ

### 4.1.1 RETURN

Program içerisinde herhangi bir noktada programın sonlandırılması istenmesi durumunda return ifadesi kullanılmaktadır. Bu ifadeden sonra program tamamen sonlanmaktadır. Aşağıdaki örnek incelendiğinde daha iyi anlaşılacaktır.



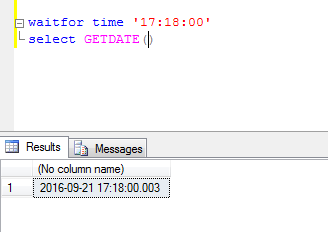
Yukarıdaki örnekten görüldüğü üzere sayici değerinin 7 olması durumunda programın sonlandırılması istenmiştir. Çıktı ekranına dikkat edildiğinde sayıcı değeri 10’dan 7 ye ininceye kadar herhangi bir hata ile karşılaşmamış olup işlemlerinde devam etmiştir. Sayici değeri 7 ye ulaştığında program başka bir kodu çalıştırmadan doğrudan programı sonlandırmıştır.

### 4.1.2 WAITFOR İfadesi

Kod bloklarının çalışması için belli bir süre bekleme yapılması veya belirlenen bir saate çalışmaya başlaması için kullanılmaktadır. Bu ifadede delay ve Time parametreleri kullanılmaktadır. Bu parametreler ile tarih, saat ve dakika ayarları yapılabilmektedir. Time parametresi ile kodun çalışacağı zaman belirlenir benzer şekilde Delay parametresi ile de kodların ne kadar süre sonra yani çalışmak için ne kadar bekleceğini belirlemek amacı ile kullanılır. Kullanım şekilleri aşağıdaki gibidir.

WAITFOR DELAY ’10:00:00’ 🡪kodların 10 dakika sonra çalışacağını belirler

WAITFOR TIME ’10:15:00’ 🡪Kodların 10:15 saatinde çalışmaya başlayacağını belirtmektedir.



## 4.2 T-SQL ‘de Hata Kontrolü

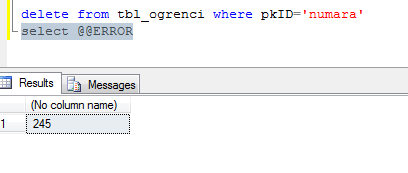
Programlama aşamaları içinde en önemli adımlardan biride hata kontrol ve denetimi adımıdır. Program çalışması esnasında oluşabilecek herhangi bir hatanın yine program içinden önceden tahmin edilip ona göre önlemi alınmalıdır. İşte bu durumda hata kontrol ifadelerinden faydalanılmaktadır-SQL ‘in önceki sürümlerinde hata denetimleri çoğunlukla if-else bloğu kullanılarak yapılmaktaydı fakat SQL server 2005 ve sonrasında TRY-CATCH bloğu kullanılmaya başlanmıştır.

T-SQL de Programın çalışması esnasında oluşacak en son hata kodu @@ERROR sistem fonksiyonunda tutulmaktadır.Herhangi bir hata oluştuğu zaman SQL server @@ERROR fonksiyonuna o an oluşan hata kodunu yüklemektedir. Bu hata kodları ve açıklamaları yine SQL serverda bulunan sys.messages sistem tablosunda tutulmaktadır.Bu tablonun içeriğine bakılarak hangi hata ile karşılşıldığı hakkında bilgi edinilebilmektedir.

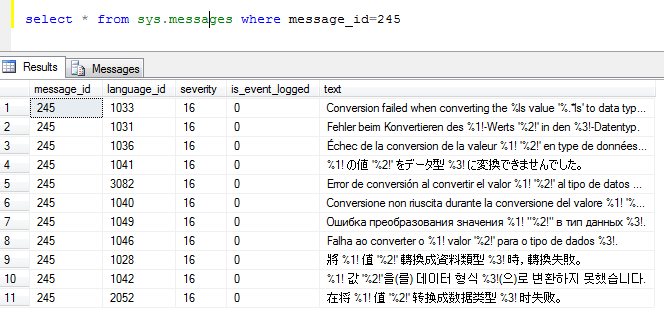


Burada dikkat edilmesi gereken durum @@ERROR fonksiyonunun sadece son oluşan hata kodunu tutmasıdır. İşlem sırasında ard arda iki hatanın oluşması durumunda bu fonksiyon sadece son oluşacak hatayı tutacaktır ve ilk oluşan hata kodu bilinmeyecektir. Bu durumdan kaçınabilmek için çalışan herbir işlemin ardından hata kontrölünün yapılması uygun olacaktır.

Örnek ; Öğrenci tablosundan ID numarasına göre bir öğrenciyi silmeyi deneyelim . Fakat burada hata oluşması için ID numarasını bir karakter katarı şeklinde verelim. Tabloda ID numarası sayısal bir değer olduğu için karekter kabul etmeyecek ve @@ERROR değişkeninde hata kodunu atacaktır.Aşağıda adım adım gösterilmektedir.



Yukarıdaki hata kodunun hangi hataya karşılık geldiğini görebilmemiz için system.message tablosunun içeriğinden kontrol edip öğrenebiliriz. Bu hata numarasına ait 11 dilde hata mesajı açıklaması ile karşılaşmaktayız.



### 4.2.1 TRY..CATCH İfadesi

Çoğu programlama dilinde kullanılan try-catch yapısı SQL Server sisteminde 2005 yılında dahil edilmiştir.Bu yapı sayesinde hata kontrolleri daha kolay bir şekilde yapılmaktadır.Try-Catch ifadesinin temel çalışma prensibi çalıştır-yakala yapısına dayanmaktadır.Bu yapıda çalıştırılacak sql kodları try bloğu arasında yazılır ve herhangi bir hata durumunda yapılacak işlemler ise catch bloğu içerinde kontrol edilir ve istenilen işlemler yapılır. Kontrol ifadelerinde olduğu gibi try-catch bloğunda da begin ve end ifadeleri ile bloğun başlangı ve bitiş noktaları belirlenmektedir.Kullanım Şekli aşağıda gösterilmektedir.

Begin Try

--Sql kodları

End Try

Begin Catch

--Hata durumunda çalışacak sql kodları

End Catch

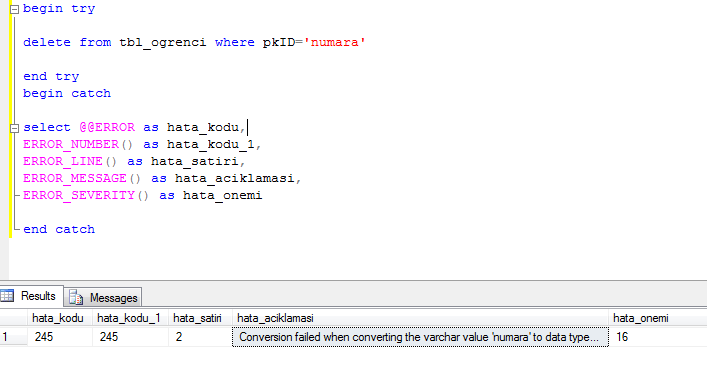
Hata oluşması durumunda hatalar hakkında detaylı bilgiler almamızı sağlayan hazır Sql fonksonları sistemde tanımlı bulunmaktadır. Bu fonksiyonlar sayesinde kod bloğunun kaçıcıncı satırında hata olduğu ve hatanın sebebi hakkında ki detaylı bilgilere ulabiliriz. Try-Catch Bloğunda kullanabilceğimiz bazı hata fonksiyonları şunlardır.

ERROR\_MESSAGE : Hatanın detaylı açıklamasını verir

ERROR\_SEVERITY: Hatanın önem bilgisini verir

ERROR\_LINE : Hatanın oluştuğu satır bilgisini Verir

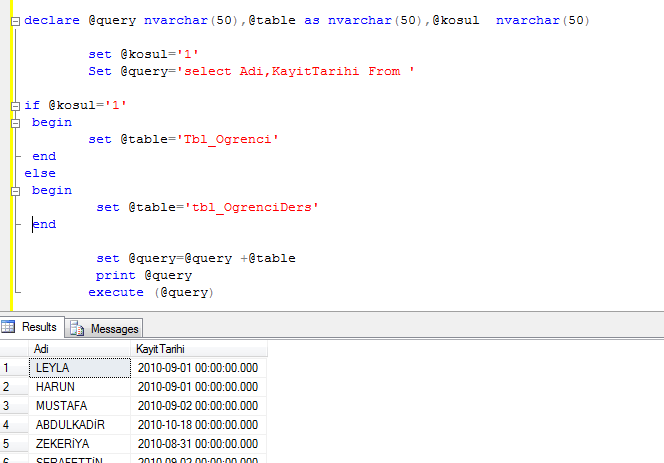
ERROR\_NUMBER: Hata kodunu verir (Daha önce görmüş olduğumuz @@ERROR fonksiyonunun işlevini gerçekleştirir )



## 4.3 DİNAMİK SQL SORGULARI

Program içerişinde koşul ifadelerinden dönen değerlere veya değişkenlerin almış olduğu değerlere göre sorguları yeniden oluşturarak veya değiştirerek oluşturulan sorgulara dinamik sorgular denmektedir. Dinamik sorgular genellik programın son noktasında çalışacak hali alırlar. Programcılar açısından gayet kullanışlı bir yöntem olmakla beraber kodlama süresini ve kod satırı sayısını da azaltmaktadır. Dinamik Sql ifadeleri oluşturma imkanımız olmamış olsaydı herbir değişilik yada herbir koşul ifadesinin sonucuna göre benzer kodları tekrar tekrar yazmamız gerekecektir. Bu yöntemle programcılar daha pratik program yazma şansını yakalamışlardır.

Dinamik olarak oluşturulan sql ifadeleri string tipine sahip bir değişken üzerinde tutulmaktadırlar ve bütün değişiklikler sonlandığında çalışma aşamasına gelindiğinde EXECUTE ifadesi ile çalıştırılığ servera gönderilmektedir. Aşağıdaki örnekler incelendiğinde konu daha iyi anlaşılacaktır.

Örnek ; 

Yukarıdaki örnekte görüldüğü üzere önceden belirtilen bir koşula göre hangi tablodan sorgu yapılacağı belirlenmektedir. Tablo belirlendikten sonra sorgunun sonunda başlangıçta oluşturulan sorgu ile sondaki tablo birleştirilerek nihai sorgu belirlenmiştir.

## 4.4 Saklı Yordamlar ( Stored Procedure)

Stored Procedure SQL Server üzerinde önceden derlenmiş olup saklanan SQL ifadeleridir. Önceden derlenmiş olarak bulunduklarından her türlü sorgulamada en iyi performansı verirler. SQL Server' da sistem tarafından "sp\_" ile başlayan isimlerle tanımlanmış bir çok yordam mevcuttur. Bunlar daha çok yönetim amaçlı olarak sistem tablolarından bilgi toplamak için kullanılırlar.

Saklı yordamlar ;

* Parametre alabilirler,
* Başka altyordamları çağırabilirler,
* Kedisini çağıran bir altyordam veya toplu işleme başarılı olduğunu ya da olamadığını, hata oluşması durumunda hatanın nedenini bir durum değeri olarak döndürebilirler.

Parametrelerin değerlerini kendisini çağıran bir altyordam döndürebilirler. Altyordamların oluşturulmasının nedeni, sıkça yapılan işlemlerin bir defa yazılarak program akışına göre tekrar tekrar kullanılmasını sağlamaktır. Böylece, kod yazımı ve programlama kolaylaştırılmış olur.

Saklı yordamlar diğer programlama dillerindeki fonksiyonlara (function) denk gelmektedir. Oluşturulan bir saklı yordama ana programdan bir komut ile ulaşılabilir.

### 4.4.1. Saklı Yordam Oluşturmak

Stored Procedure’ün oluşturulma şekli aşağıdaki gibidir

CREATE PROC [ EDURE ] prosedür\_adı

AS

T-SQL ifadeleri

GO

Oluşturacağınız saklı yordamlar ile sistemin saklı yordamlarının karışmamasına dikkat ediniz.

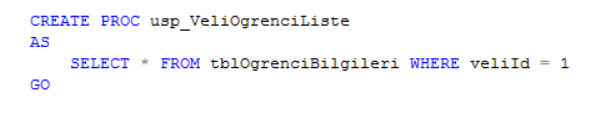
Saklı yordam oluşturabilmek için sysadmin, db\_owner veya dll\_admin rolüne sahip olmanız gerekir

Bir saklı yordam; CREATE DEFAULT, CREATE PROCEDURE, CREATE RULE, CREATE TRIGGER ve CREATE VIEW ifadelerini içeremez. Ancak, her nesneden veri alabilir

Örnek: tblOgrenciBilgileri tablosunda veliId’si 1 olan öğrencileri listeleyen bir store procedure oluşturalım.



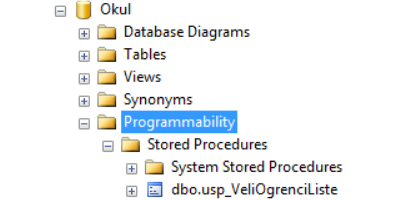
New Query buttonuna basarak yenibir sorgu ekranı oluşturmamız gerekmektedir. Açılan ekrandan aşağıdaki sql kodlarını yazarak stored procedurü oluşturabiliriz.



Şimdi Execute butonuna tıklayarak veya klavyeden F5 tuşuna basarak sorgumuzu çalıştıralım.

Çalıştırdıktan sonra sonuç penceresinde herhangi bir liste oluşmayacaktır. Biz burada sadece ağ veritabanı yazılımımıza yeni bir store procedure tanımladık.

Tanımlamış olduğumuz store procedure listesine Programmability düğümü altında yer alan Stored Procedures düğümünden erişebiliriz.

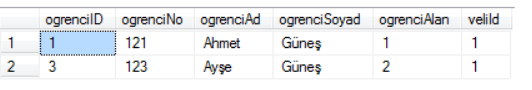


Tanımlamış olduğumuz saklı yordamı çalıştırmak için EXEC komtunu kullanırız. Şimdi tanımladığımız usp\_VeliOgrenciListe saklı yordamını çağıralım ve sonucu görelim.

Yeni bir sorgu penceresi açalım. Sorgu penceremize aşağıdaki komutu yazalım.



Execute butonuna tıklayarak veya klavyeden F5 tuşuna basarak sorgumuzu çalıştıralım. Sorgumuzu çalıştırdıktan sonra sonuç penceresinde aşağıdaki gibi bir liste oluşacaktır



### 4.4.2 Saklı Yordamlarda Değişiklik Yapmak

Saklı yordamlarda değişiklik yapmak için ALTER komutu kullanılır.

Genel Kullanımı:

ALTER PROCEDURE prosedür\_adı

AS

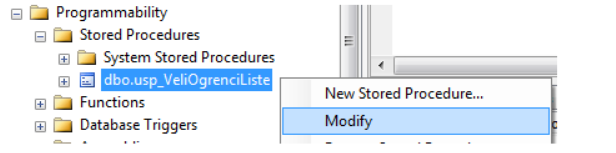
T-SQL ifadeleri

GO

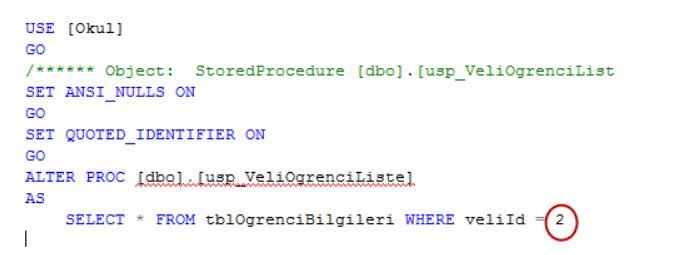
Bir Stored Procedure’de değişiklik yapabilmek için önce kaynak kodunun alınması ve bir Query ekranına kopyalanıp düzenlenmesi gerekir.

Örnek; usp\_VeliOgrenciListe adı prosedürümüzde veliId’si 1 olan öğrencileri listelemiştik. Bu prosedürü veliId’si 2 olan öğrencileri listeleyecek şekilde değiştirelim.

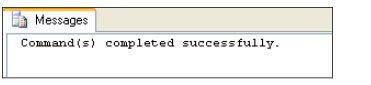
Oluşturduğunuz “usp\_VeliOgrenciListe” adındaki Stored Procedure üzerinde fareyle sağ tıklıyoruz ve açılan menüden Modify seçeneğini seçiyoruz



Modify komutunu verince kodlar bir Query sayfası şeklinde ekrana gelecektir. Üzerinde gerekli değişiklikleri yapınız ve Execute (F5) ediniz.



Execute işleminin sorunsuz bir şekilde gerçekleştiğini Messages penceresinde görebilirsiniz.



Yeni bir Query ekranı açınız ve Stored Procedure’ün çalışması için gerekli kodu yazınız.



usp\_VeliOgrenciListe stored procedure’ü işletilecek ve sonuçlar Results penceresinde size gösterilecektir.

