

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Veri Tabanına Giriş**



Microsoft Türkiye

**Açık Akademi**

## Veri Nedir?

**Veri tabanı (Database)** kavramına girmeden önce veriyi kısaca tanımlamak önemlidir. **Veri, (data)** bir veya birden fazla bilgidan oluşan bir kümedir. İsim, yaş, telefon numarası, bir toplama işleminin sonucu ya da bir sınıfın yaş ortalaması birer veridir. Bir veritabanı yapısı içerisinde tutulan bilgilere veri denilmektedir. Bilgisayar ortamına aktarılan, işlenmemiş (ham) bilgiler de veri olarak adlandırılabilir. Bilgisayara girilen, bilgisayar tarafından saklanabilen ve işlenebilen herşeye veri denir.

## Veri Neden Saklanır?

Veri saklamadaki ana amaç, daha ileri bir tarihte saklanan verilere yeniden ulaşabilmek ve kullanabilmektir. Kullanma süreci sadece verinin listelenmesi gibi basit bir işlem olabileceği gibi, veriler üzerinden hesaplamalar yapılarak raporlar hazırlanması gibi daha karmaşık bir işlem de olabilir. Şimdi sıra geldi veriyi nasıl saklayabileceğimizi öğrenmeye.

## Veri Saklama Yöntemleri

Verileri saklamak için çeşitli ortamları tercih etmek mümkündür. Verilerin klasik yöntem ile kalem kullanarak bir dosya kağıdına satır satır yazılması da bir veri saklama yöntemidir, bilgisayar üzerinde notepad kullanarak metin tabanlı basit bir dosyada depolanması da bir veri saklama yöntemidir. Daha düzenli ve detaylı bir şekilde veri saklamamızı sağlayacak yöntemler arasında, tablo yapısında veri saklamamızı sağlayan Excel ve Access gösterilebilir. Veri saklama işlemini bir örnek ile detaylandıralım:

Bakkal ve marketlerde veresiye satış yapıldığında müşterinin aldığı ürünlerin adı, miktarı, fiyatı gibi bilgiler bir veresiye defteri içerisinde müşteriye ait bir sayfada saklanır. Bu şekilde bakkalın sahibi, gerektiğinde ilgili müşterinin sayfasını açıp hangi ürünlerden ne kadar miktarda aldığını ve borç toplamının ne kadar olduğunu hesaplayabilir. Fakat verinin bu şekilde saklanması birçok açıdan kötü sonuçlara yol açabilir. Verilerin kâğıttan yapılmış bir defterde tutulması düzensizliklere ve karışıklıklara yol açabileceği gibi yırtılma, ıslanma, kaybolma gibi veri kaybına sebep olabilecek risklere de açıktır. O zaman gelin biz bu defterdeki verileri bilgisayar ortamında saklayalım. Bilgisayarımızda veresiye adında bir klasör oluşturalım. Bu klasör içerisine Notepad gibi bir metin editörü ile her müşterimiz için bir metin dosyası açalım. Açtığımız dosyaların adlarına, daha sonra kolay bir şekilde ulaşabilmek için bir numara verelim. (Örneğin Ahmet isimli müşterimiz için 302 numarasını verelim) Bu metin dosyasından faydalananarak, defterde tuttuğumuz verileri daha düzenli bir şekilde saklayabiliriz. Bu bilgileri düzenli bir şekilde kaydedip, yedekleyerek başımıza gelebilecek veri kayıplarını da en aza indirebiliriz.

İkinci senaryomuz, yani verileri bilgisayarda saklamamız ilk senaryodaki veresiye defterine göre elbette ki daha esnek, hızlı ve sağlıklı bir yöntem olacaktır. Fakat bir metin dosyası üzerinde girilen verilere müdahale etme şansımız pek olmayacaktır. Örneğin ücret bilgisinin yazılacağı alana 40, 40,5 veya kırk gibi farklı tipte veriler girilebilir ya da ücret kısmı boş bırakılabilir. Bu şekilde kaydedilen bir dosyanın daha sonra bazı karışıklıklara yol açabilme riski yüksektir. Peki, bu bilgiler üzerine yeni veri eklemek, istediğimiz kıstaslara göre veri seçmek, var olan veriyi değiştirmek veya veriler üzerinde işlemler yapmak istediğimizde çok iyi sonuçlara ulaşabilir miyiz? Elbette ki bu tip işlemleri metin dosyası üzerinde yapmamız biraz olsun işimizi kolaylaştıracaktır. Fakat *“Daha iyi bir yol olabilir mi?”* diye düşünecek olursak cevabı kesinlikle *“Evet”* olacaktır.

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Veri Tabanı Nedir?**



Microsoft Türkiye

**Açık Akademi**

Veri tabanı (Database) temel olarak farklı tiplerdeki verileri düzenli bir şekilde saklamamızı ve kullanmamızı sağlayan depolama ortamıdır. Bu ortam içerisinde verileri saklayabilir, onlara kolay bir şekilde ulaşabilir ve gerektiğinde bu verilerin üzerinde değişiklikler yapabiliriz. Veri tabanı, veriler arasında bütünlük ve düzen sağlarken, veriye hızlı erişim ve bakım kolaylığı da sunar. Bu sistemin kullanılması sonucu veriyi hatasız ve sağlıklı bir biçimde işlemek de kolaylaşır.

Bir veri tabanı sistemi üzerinde istediğimiz verileri depolayabilir, depolanan verileri çağırırken bazı şartlar belirtebilir, verileri değiştirebilir veya silebiliriz. İlerleyen konularda öğreneceğimiz T-SQL dili ile aşağıdaki örnekleri çok basit bir şekilde gerçekleştirebileceğiz:

- Burak Batur isimli müşterinin borç bilgisine 1,5 YTL fiyatıyla 2 adet margarin ekle.
- Burak Batur isimli müşterinin satın aldığı tüm ürünlerin listesini getir.
- Ercan Bozkurt isimli müşterinin 1 Ocak 2011 ile 10 Şubat 2011 tarihleri arasında satın aldığı tüm ürünleri getir.
- Ferda Demir isimli müşterinin toplam borç bilgisini getir.
- Tüm müşterilerin toplam borç bilgisini getir.

Bu örnekleri genişletmemiz ve arttırmamız elbette mümkün. Fakat şimdilik bu örnekler bir veri tabanı sistemi üzerinde ne gibi işlemler yapabileceğimizi görmemiz açısından yeterli bir temel teşkil edecektir.

Günümüzde veritabanları hemen hemen her alanda sıklıkla kullanılmaktadır. Bu alanlara örnek verecek olursak;

- Kişisel adres defterleri
- Telefon rehberi
- TV rehberi
- Online sözlükler
- Kütüphane sistemleri
- Ödeme ve borç sistemleri
- Ürün satış ve sipariş sistemleri
- E-Ticaret siteleri
- Banka sistemleri
- Okul sistemleri
- Hastane sistemleri

gibi birçok alanda gerekli bilgiler veritabanlarında tutulmaktadır. Örneğin bir kütüphaneden ödünç aldığımız kitaplarla ilgili olarak, kitap adı, alış tarihi, geri verişi tarihi gibi bilgiler veri tabanındaki bir tabloda tutulmaktadır. Kütüphanedeki görevli istediği zaman kayıtlı bir kullanıcı ile ilgili bilgilere ulaşabilmektedir.

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Veri Tabanı Yönetim  
Sistemleri**



Microsoft Türkiye

**Açık Akademi**

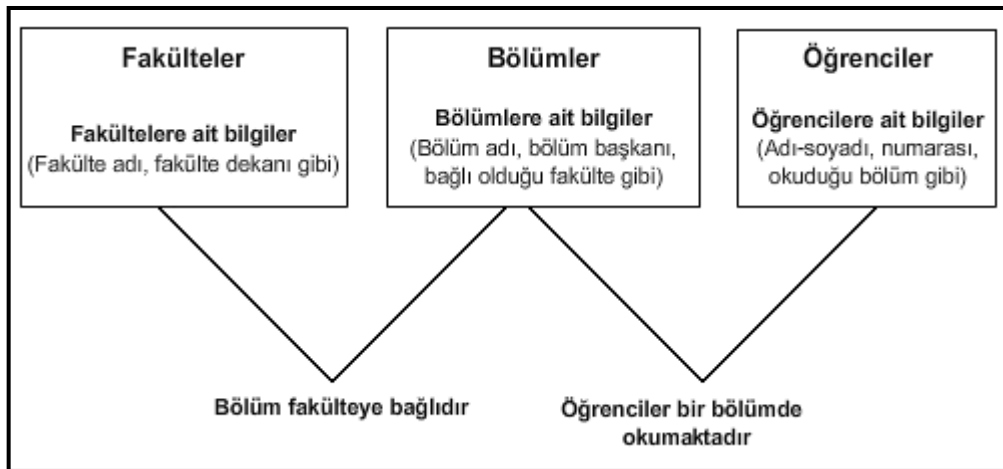
**Veri Tabanı Yönetim Sistemleri (DataBase Management System - DBMS)**, veri tabanında tutulacak olan verilerin uyacağı standartları, bu verilere nasıl erişilebileceğini ve verilerin disk üzerinde nasıl tutulacağını belirleyen sistemlerdir. Bu sistemler aracılığıyla verilerin bütünlüğü ve güvenliği sağlanmaktadır. Programların veya sistemlerin veri tabanı içerisindeki verilere kolay ve hızlı bir şekilde erişebilmesini ve verileri yedekleyebilmesini sağlamaktadır. Bu sistemler, birden fazla veri tabanı üzerinde işlemler yapabilecek şekilde tasarlanmıştır. Bunların yanı sıra, olabilecek felaket senaryoları (disaster case) karşısında, veri tabanının sorunsuzca işleyebilmesi için tedbirler almaktadır.

## İlişkisel Veri Tabanı Yönetim Sistemleri

Gelişen bilgi teknolojileri ile birlikte, veri tabanı sistemlerinde de birçok yenilikler olmuştur. İlk veri tabanı sistemlerinde veriler gelişigüzel bir şekilde depolandığı için veriler üzerinde değişiklikler yapmak zor bir işlemdi. Kullanıcıların veriler üzerinde değişiklik yapabilmesi için veri tabanının yapısını iyi bilmesi gerekiyordu. Yine bu sistemler tam olarak veri bütünlüğünü ve güvenliğini sağlayamıyordu. Günümüzde en yaygın olarak kullanılan veri tabanı mimarisi olan ilişkisel veritabanlarında ise veriler tablolar, satırlar ve alanlar halinde tutulmaktadır. Bu sayede veriler arasında ilişkilendirmeler yapılabilmekte, verilerin kullanımı ve yönetimi daha verimli hale getirilmektedir. İlişkisel veri tabanı mimarisini kullanan sistemlere ise **İlişkisel Veri Tabanı Yönetim Sistemi (Relational Database Management System - RDBMS)** adı verilmektedir. Günümüzde en sık kullanılan ilişkisel veri tabanı yönetim sistemleri arasında Microsoft SQL Server , Oracle , Microsoft Access , PostgreSQL , Sybase , MySQL , Berkeley ve Firebird'ü gösterebiliriz.

Bir veri tabanını doğru bir biçimde tasarlayabilmek için varlıklar arasındaki ilişkileri (entity relationship) iyi şekilde kavramak gerekmektedir. Verileri gerçek hayat ile ilişkilendirip, aralarındaki bağlantıları oluşturup, veri tabanının mantıksal şablonu çıkarılmalıdır. SQL Server gibi bir RDBMS ile hazırlanan veri tabanında yer alacak veriler, tablolar ve alanlar (sütunlar) içerisinde tutulur. Tablolara kaydedilecek olan verilerin kendilerine ait uygun veri tipleri belirlenip, girilecek olan değerler için bazı sınırlandırmalar getirilebilir. Yine veriler arasında ilişkilendirmeler yapıp, veri tabanına ilişkisel bir yapı kazandırılabilir. **SQL (Structured Query Language)** dili ile sorgular oluşturularak RDBMS üzerinden veri tabanı ile iletişim kurulabilir.

Veri tabanı tasarlanırken, verilerin gerçek dünyada aralarında olan ilişkilerini göz önüne aldığımızda, verileri ilişkilendirmemiz daha da kolaylaşır. Örneğin bir üniversite için temel olarak fakülteleri, bölümleri ve öğrencileri ele aldığımızda *bölüm ile fakülte* arasında ve *öğrenci ile bölüm* arasında bazı ilişkiler olacaktır. Alt birimden üst birime doğru bir sıralama yaptığımızda *öğrenci-bölüm-fakülte* gibi bir ilişkilendirme ortaya çıkmaktadır. Böyle bir senaryoda, veri tabanı tasarlanırken öğrenci, bölüm ve fakülte arasındaki ilişkiler ele alınarak tablolar tasarlanabilir ve gerekli ilişkilendirmeler yapılabilir. Aşağıdaki şekilde öğrenci, bölüm ve fakültenin işlevleri ve aralarındaki ilişkilerin şekle dökülmüş hali bulunmaktadır.



Fakülte, bölüm ve öğrencilerin şekil üzerinde tanımlanması ve ilişkilendirilmesi

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**SQL Server Management  
Studio**



Microsoft Türkiye

**Açık Akademi**

## SQL Server nereden indirilir?

<http://www.microsoft.com/download> adresini tarayıcında açıp, arama kutusuna 'sql server 2008 r2 express' yazarak arama işlemini başlattıktan sonra, gelen sonuçlar arasından **Microsoft SQL Server 2008 R2 RTM – Express with Management Tools** linkini takip ederek indirme sayfasına ulaşabilirsin. Bu sayfada işletim sistemine uygun olan sürümü (32-bit veya 64-bit) indirebilirsin. Detayların anlatıldığı videoyu izlemek için aşağıdaki simgeye tıkla.

## SQL Server Kurulumu

SQL Server kurulumu ile ilgili detayları kurulum videomuzda bulabilirsin.

## SQL Server Management Studio

Veri tabanı üzerinde gerçekleştirilecek olan işlemleri, programlama dilleri aracılığıyla yapabildiğimiz gibi SQL Server Management Studio uygulamasıyla da gerçekleştirebiliriz. SQL Server Management Studio, **Microsoft SQL Server 2008 R2 RTM – Express with Management Tools** kurulumu sırasında makinana yüklenecektir.

SQL Server makinanda daha önceden kurulu ise ve sen sadece Management Studio'yu indirmek istiyorsan; <http://www.microsoft.com/download> adresini tarayıcında açıp, arama kutusuna 'management studio express' yazarak arama işlemini başlattıktan sonra, gelen sonuçlar arasından **Microsoft® SQL Server® 2008 Management Studio Express** linkini takip ederek indirme sayfasına ulaşabilirsin. Bu sayfada işletim sistemine uygun olan sürümü (32-bit veya 64-bit) indirip kurabilirsin.

## SQL Server Management Studio Üzerinde Sorgularla Çalışmak

**SQL Server Management Studio, Microsoft SQL Server** ile birlikte gelmiş olan yeni bir sorgulama ve yönetim aracıdır. Management Studio, SQL Server veritabanlarına erişme, veritabanları üzerinde ayarlama işlemlerini gerçekleştirme, yönetim ve veri tabanı üzerinde sorgular çalıştırma gibi işlemleri yapabilmektedir. Yine Management Studio üzerinde SQL Server projeleri oluşturulup, çalışmaların Visual Studio ortamında olduğu gibi proje şeklinde kaydedilip, daha sonra bu projeler üzerinde geliştirme işlemleri yapılabilmesi sağlanmaktadır.

SQL Server Management Studio aynı zamanda, veri tabanı üzerinde T-SQL sorgularını çalıştırmak, sonuçlarını görmek, sorguları analiz etmek ve veri tabanı üzerinde bazı ayarlamaları yapmak için de kullanılabilir. SQL Server Management Studio ile yapabileceğin işlemler ve sunulan kolaylıklardan bazılarını aşağıda liste halinde bulabilirsin.

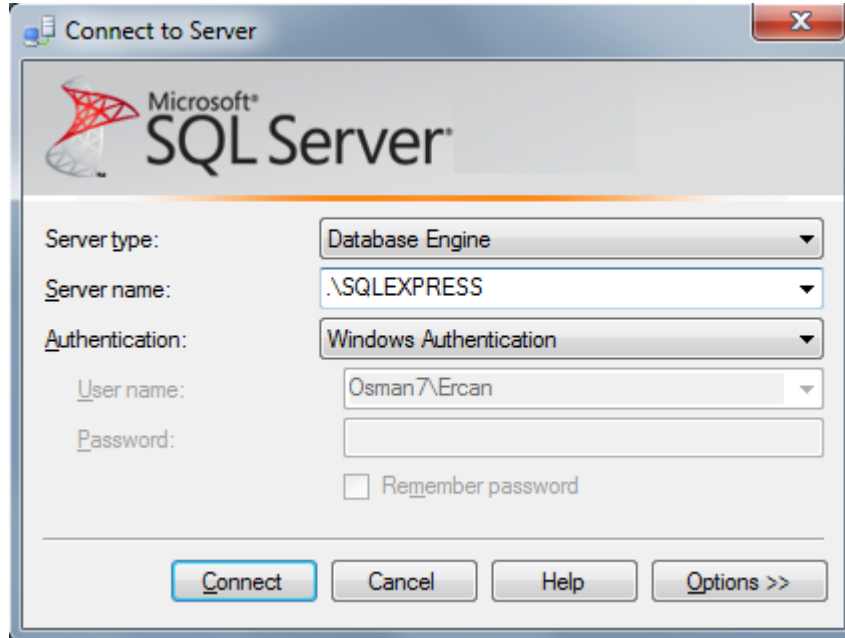
- Veri tabanı ve tablolar üzerinde yapılabilecek tüm sorgu işlemlerinin gerçekleştirebileceği, kodları renklendirilerek sorguların daha anlaşılır olmasını sağlayan metin editörünü kullanmak.
- Çalışan sorguların sonuçlarını görmek.
- Hazırlanan sorguların test edilmesi. Sorguda bir hata olması durumunda hatanın nerede olduğunu ve neden kaynaklandığını kolayca görebilmek.
- Bağlı olunan veri tabanı üzerindeki tabloları ve diğer nesneleri arayüz üzerinden görebilmek. (Object Browser aracılığı ile)
- Birçok SQL sorgusunun hazır olarak bulunduğu şablonları (template) kullanmak. Bu şablonlar yardımıyla kolay bir şekilde sorgular hazırlayabilmek.



SQL Server Management Studio, Microsoft tarafından SMO (SQL Server Management Object) kütüphanesinden yararlanılarak **Visual Studio** ile .NET Framework kullanılarak geliştirilmiştir.

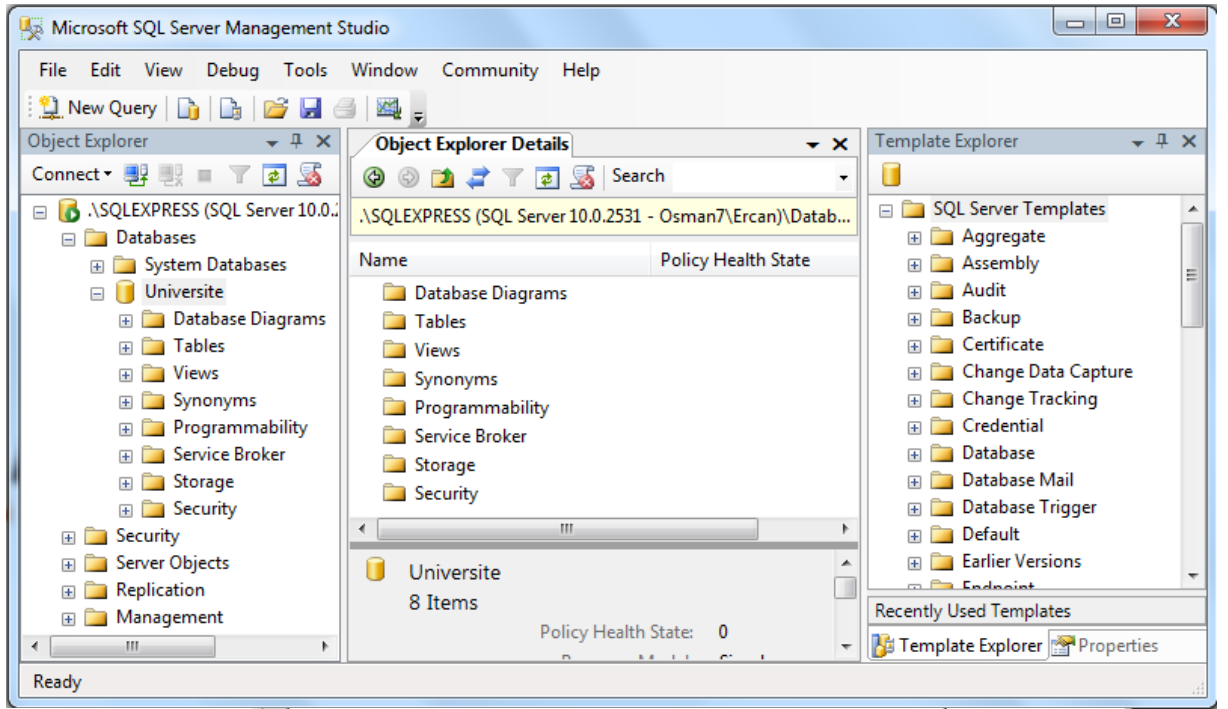
SQL Server Management Studio'yu açmak için Türkçe işletim sistemlerinde **Başlat > Programlar > Microsoft SQL Server** menüsünden **SQL Server Management Studio** uygulamasını başlatmalısınız.

Management Studio açıldığında karşımıza ilk olarak bir SQL Server sunucusuna ulaşmamız için bir bağlantı penceresi gelecektir. Bu pencereden **Server Type**, **Server Name** ve **Authentication** seçeneklerinden bağlanacağımız sunucunun özelliklerine göre uygun seçenekleri girerek bağlantı yapabiliriz. Kendi bilgisayarımızdaki SQL Server sunucusuna bağlanmak istediğimizde, Server Name kısmında makina adımızı ve bilgisayarımızda kayıtlı olan SQL Server'ın ismini (instance name) yazmamız gerekir. Uzak bir bilgisayar üzerindeki SQL Server sunucusuna bağlanmak için ise Server Name kısmına bağlanılacak bilgisayarın IP numarası veya uzak sunucudaki SQL Server'ın alan adı (domain name) yazılır. Windows üzerinde o an giriş yapmış olan kullanıcının hesabı ile bağlantı yapmak için **Windows Authentication** seçeneğini seçmemiz yeterli olacaktır. Bağlanılacak SQL Server üzerindeki bir kullanıcı ile giriş yapmak için ise **SQL Server Authentication**'ı seçip sunucuya bağlanma yetkisi olan kullanıcı adı ve şifreyi girmemiz gerekecektir. **Connect** butonuna tıkladığımızda girilen bilgiler doğru ise SQL Server sunucusuna bağlanılacaktır.



SQL Server Management Studio bağlantı ekranı

Management Studio ile bir SQL Server sunucusuna bağlandığımızda karşımıza gelen arayüzde **Object Explorer**, **Template Explorer** ve **Object Explorer Details** pencereleri yer alabilmektedir. (Bu pencerelerden herhangi biri açık değilse, **View** menüsünden açılıp tekrar kapatılabilir.)



#### Object Explorer

Bağılı bulunan SQL Server içerisindeki veritabanlarını ve diğer tüm nesneleri ağaç yapısı içerisinde görüntüler

#### Object Explorer Details

Object Browser'da seçili olan nesnenin içerisinde bulunan yapıları görüntüler

#### Template Explorer

Birçok SQL sorgusunun hazır bulunduğu şablonlar

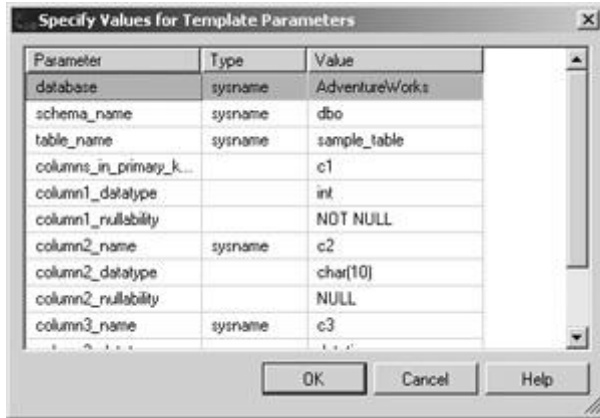
SQL Server Management Studio genel görünümü

**Object Explorer:** Bağlı bulunan SQL Server sunucusu üzerinde bulunan tüm nesneleri (veritabanları, tablolar, kullanıcılar vb.) ağaç yapısı biçiminde görüntülememizi ve bu yapıların içerisinde gezinmemizi sağlar. Yine Object Explorer üzerinden yeni veri tabanı oluşturma, tablo ekleme, varolan nesneleri silme gibi işlemler yapılabilmektedir. Görüntülenen nesneler üzerinde filtreleme işlemleri yapılabilmektedir. Management Studio ile birden fazla SQL Server sunucusuna bağlanabilir, bağlı bulunduğu tüm sunucuları Object Explorer penceresi içerisinde görüntüleyebilir ve üzerlerinde işlemler yapabilirsiniz. Object Explorer üzerinde yapabileceğiniz bazı temel işlemler şunlardır:

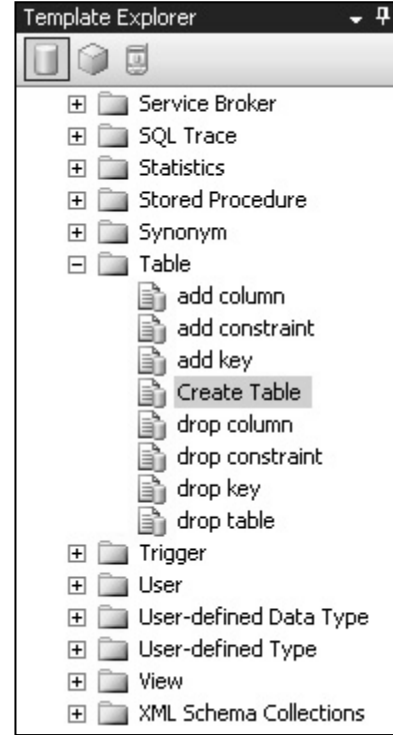
- Yeni bir SQL Server sunucusuna bağlanmak için **Object Explorer** penceresinin sol üst köşesinde yer alan **Connect** butonu aracılığı ile aynı anda birden fazla SQL Server sunucusuna bağlanılabilir.
- Bağlı olduğun SQL Server'da sorgu çalıştırmak için, sunucu adının üzerine sağ tıklayıp **New Query** seçeneği seçilir.
- Bir veritabanı üzerinde sorgu çalıştırmak için o veritabanının üzerine sağ tıklayıp **New Query** seçeneği seçilir.
- Veritabanına yeni bir tablo eklemek için veritabanı içindeki **Tables** kısmında sağ tıklayarak **New Table** seçeneği seçilir.
- Bir tablonun yapısını incelemek ve değişiklikler yapmak için tablo üzerine sağ tıklayıp **Design** seçeneği seçilir.
- Bir tablo içerisindeki kayıtları görmek için tablo üzerine sağ tıklayıp **Select Top 1000 Rows** seçeneğini kullanmak gerekir.

**Object Explorer Details:** Bu ekranda ise Object Explorer içerisinde o an için seçilmiş olan nesnenin içeriği görüntülenmektedir. Pencerenin kendi içerisinde bir üstteki veya bir alttaki yapılara ulaşılabilir. Veri tabanı içerisinde gezinme işlemlerini daha kolay hale getirmek için bu pencere kullanılabilir.

**Template Explorer:** SQL Server'ın içerisinde gelen ve bazı işlemleri basit bir şekilde yapmamızı sağlayan **sorgu şablonları(template)** bulunmaktadır. Bu şablonlar içerisinde birçok işlevi yerine getiren SQL sorgu cümleleri bulunur. Template Explorer içerisindeki sorgu şablonlarını açıp gerekli değişiklikleri yaparak sorguları çalıştırılabilir ve hızlı bir şekilde işlemler gerçekleştirebilirsiniz. Şablon içerisinde değiştirilmesi gereken parametreler, **Query** menüsünden **Specify Values for Template Parameters** seçeneği ile çıkan pencerede gerekli parametreler girilerek düzenlenebilir.



Specify Values for Template Parameters ekranından template içerisindeki parametreler kolay bir şekilde değiştirilebilir



Template Explorer içerisinde bulunan hazır SQL sorgu şablonları

### Management Studio'da SQL Komutları ile Çalışmak

Management Studio içerisinde SQL komutları çalıştırmak için komutların çalıştırılacağı veritabanı üzerine sağ tıklayıp New Query seçeneğini seçmelisin. Açılan pencerede, SQL cümleleri yazıp, test edebilir ve çalıştırılabilirsin. Yazılan SQL sorgularını test etmek için **Query** menüsünden **Parse** seçeneğini kullanabilir veya **CTRL+F5** tuş kombinasyonunu kısayol olarak kullanabilirsin. Test etme işlemi sadece yazılan kodların doğru olup olmadığını kontrol etmek için yapılan bir işlemdir. Test edilen kodlar çalıştırılmaz, yani veri tabanı veya tablolar üzerinde herhangi bir işlem gerçekleşmez. Test işlemi sonucunda eğer yazmış olduğun SQL ifadelerinde bir hata varsa, alt kısımda açılan bir pencerede hatanın neden kaynaklandığı ve kodun neresinde oluştuğunu görebilirsin. Hazırladığın SQL ifadelerini çalıştırmak için ise **Query** menüsünden **Execute** seçeneğini seçebilir veya **F5** tuşunu kısayol olarak kullanabilirsin. Test işleminden farklı olarak çalıştırma işleminde veritabanı hazırlanan sorgu sonuçlarından etkilenecektir.

Sorgu çalıştırma ekranında birden fazla SQL ifadesi yazabilir ve bunlardan sadece istediğin kısmı çalıştırabilirsin. Çalıştırmak istediğin SQL kodlarını seçerek Execute (F5) işlemini yaparak, sadece seçili alandaki kodları çalıştırabilirsin.

```
INSERT INTO Dersler (DersAd, DersBolumId)
VALUES ('Fizik-2', 114)

SELECT * FROM Dersler
```

Sadece seçili alandaki kodlar çalıştırılacaktır

Query penceresi içerisinde sadece işaretli alandaki sorgular çalıştırılır

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Temel Veri Tabanı Kavramları**



Microsoft Türkiye

**Açık Akademi**

## SQL (Yapısal Sorgulama Dili)

**SQL**, yani **Structured Query Language (Yapısal Sorgulama Dili)** tüm ilişkisel veritabanlarında standart olarak kullanılan bir dildir. SQL veritabanı ile kullanıcı arasındaki iletişimi sağlar. Bu dil aracılığıyla hazırladığınız sorguları kullanarak veritabanında depolanan veriler üzerinde bütün işlemleri yapabilirsiniz. SQL dilinin standartları ANSI (American National Standart Institute) ve ISO (International Standarts Organization) tarafından sağlanmakla birlikte, günümüzde en yaygın olarak ANSI standartları kullanılmaktadır.

## T-SQL (Transact SQL)

**T-SQL**, **Transact SQL** adı verilen bir SQL dilidir. SQL dilinin **Microsoft SQL Server** üzerinde kullanılan sürümüdür. Daha iyi performans sağlaması için SQL dili üzerine eklentiler ve fonksiyonellikler eklenerek oluşturulmuştur. Veri tabanından bağımsız olarak, bir programlama dili aracılığıyla kullanıcıdan gelen T-SQL sorgularının sonuçları ilişkisel veri tabanı yönetim sistemi (RDBMS) tarafından oluşturularak kullanıcıya gönderilir. Bu şekilde kullanıcı, veri tabanı ile birebir uğraşmaksızın, sadece sorgular yazarak veri tabanı üzerinde işlemler yapabilir, veri tabanından gelen sonuçları program veya bir web sayfası üzerinde görüntüleyebilir.

## T-SQL İfade Tipleri

T-SQL içerisinde 3 farklı ifade tipi bulunmaktadır.

- **Veri Tanımlama Dili (Data Definition Language)**
- **Veri Kontrol Dili (Data Control Language)**
- **Veri İşleme Dili (Data Manipulation Language)**

Bu üç ifade tipini de eğitimimiz içerisinde yeri geldikçe inceleyeceğiz.

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Veri Tanımlama Dili (DDL)**



Microsoft Türkiye

**Açık Akademi**

## Veri Tanımlama Dili (Data Definition Language - DDL)

Veri tabanında nesneler oluşturmak için gerekli olan ifadeleri sağlamaktadır. Bu ifadeler, veri tabanı veya tablo gibi yeni bir nesnenin oluşturulması, var olan nesne üzerinde değişiklikler yapılması veya nesnenin yok edilmesi için kullanılır. Üzerinde işlem yapılan nesnenin ne gibi özellikleri ve alanları olacağı bu ifadeler içerisinde belirlenir.

T-SQL dilinde 3 adet veri tanımlama ifadesi bulunmaktadır. Bunlar:

- **CREATE**
- **ALTER**
- **DROP**

ifadeleridir. CREATE nesne oluşturmak, ALTER var olan bir nesne üzerinde değişiklikler yapmak, DROP ise varolan bir nesneyi kaldırmak için kullanılmaktadır.

### CREATE


Veri tabanı üzerinde nesne oluşturmak ya da tanımlamak için kullanılan komuttur. Oluşturulacak nesnenin özelliklerine göre farklı parametreler alabilmektedir. Genel kullanımı şu şekilde olmaktadır:

#### CREATE NESNETİPİ Nesne Adı (Nesneye ait gerekli tanımlamalar)

Aşağıda CREATE ifadesi ile ilgili inceleyebileceğin örnek bir kullanım bulunmaktadır.

```
CREATE DATABASE Üniversite
ON
(
    NAME = Üniversite,
    FILENAME = 'C:\universite.mdf',
    SIZE = 4mb,
    MAXSIZE = 10mb,
    FILEGROWTH = 1mb
)
```

Yukarıdaki ifadede **Üniversite** adında yeni bir veri tabanı oluşturuyoruz. **ON ( )** kısmında oluşturulan veri tabanının özellikleri belirleniyor. **NAME** değeri ile oluşan birincil veri tabanı dosyasının adını, **FILENAME** değeri ile veri tabanı dosyasının nereye ve ne isimle yazılacağını, **SIZE** değeri ile dosyanın başlangıçta diskte ne kadar yer kaplayacağını, **MAXSIZE** ile dosyanın diskte en fazla ne boyuta kadar büyüyebileceğini, **FILEGROWTH** ile de dosyanın ne kadarlık boyutlarla büyüyeceğini belirlemektedir.

 Sadece **CREATE DATABASE Üniversite** ifadesini kullanarak ta Üniversite isimli bir veri tabanı oluşturabiliriz. Bu durumda oluşturacağımız veri tabanı sistem tarafından atanmış olan varsayılan ayarlamalara göre oluşturulacaktır.

### ALTER

Varolan bir nesne üzerinde değişiklikler yapmak için kullanılır. CREATE komutunda olduğu gibi değiştireceği nesneye göre farklı parametreler alabilmektedir. Genel kullanımı aşağıdaki gibidir.

#### ALTER NESNETİPİ Nesne Adı Yapılacak Değişiklik

Aşağıdaki örneklerde ALTER ifadesi kullanılarak varolan bir tablo üzerinde nasıl değişiklikler yapılabileceği gösterilmiştir.

```
ALTER TABLE Ogrenci  
ALTER COLUMN AdSoyad NVARCHAR(30) NOT NULL
```

Yukarıdaki örnek ALTER ifadesine aşinalık sağlamak amacıyla verilmiştir.

## DROP

Veri tabanındaki herhangi bir nesneyi silmek için kullanılır. Silinen nesne ile ilgili olarak içerisinde tuttuğu tüm bilgiler de silinmektedir. Örneğin bir tablo silindiğinde içerisindeki tüm bilgiler de veri tabanından silinecektir. Kullanımında dikkat edilmesi gereken bir sorgu ifadesidir. Genel kullanımı aşağıdaki gibidir.

### DROP NESNETİPI Nesne Adı

Aşağıdaki örneklerde **DROP** ifadesi kullanılarak varolan tabloların nasıl silinebileceği gösterilmiştir.

Üzerinde çalışmış olduğunuz veritabanında herhangi bir veri kaybı yaşamamak için, önce geçici olarak işlevi olmayan bir veritabanı oluşturup, sonra bu nesneyi **DROP** sorgusu ile nasıl silinebileceğimizi görelim.

```
CREATE DATABASE TestVeritabani  
GO
```

Yukarıdaki ifade ile **TestVeritabani** adında bir veri tabanı oluşturulur. **GO** komutu öncelikle **CREATE DATABASE TestVeritabani** sorgusunun çalışmasını sağlar.

**GO** ifadesi bir sorguyu çalıştır anlamında kullanılmaktadır. Yukarıdaki sorguda iki farklı sorgu cümlesi bulunmaktadır ve buradaki ikinci cümle için öncelikle TestVeritabani isimli bir veri tabanının oluşturulması gerekmektedir. **GO** ifadesi önce ilk sorgunun çalışmasını sağlamaktadır.

### DROP DATABASE TestVeritabani

**DROP DATABASE** ifadesi ile **TestVeritabani** isimli veri tabanı silinmektedir.



**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Veri Kontrol Dili (DCL)**



Microsoft Türkiye

**Açık Akademi**

## Veri Kontrol Dili (Data Control Language - DCL)

T-SQL'de veri kontrol ifadeleri veri tabanı üzerindeki kullanıcılara ve rollere yetki vermek için kullanılmaktadır. T-SQL'de 3 adet veri kontrol komutu bulunmaktadır. Bu komutlar şunlardır:

### GRANT

Kullanıcıya veritabanına erişebilmesini veya T-SQL ifadeleri çalıştırabilmesini sağlayacak yetkileri verir.

### DENY

Kullanıcının belirli bir alana erişimini engellemek veya belirli T-SQL ifadelerini çalıştıramamasını sağlamak amacıyla kullanılır.

### REVOKE

Daha önceden GRANT veya DENY ile verilmiş yetki veya engelleri kaldırmak için kullanılır.

Veri kontrol ifadelerini çalıştırmak için veri tabanına bağlı olan kullanıcının **sysadmin**, **dbcreator**, **db\_owner** veya **db\_securityadmin** rollerinden birine sahip olması gerekmektedir.

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Veri İşleme Dili (DML)**



Microsoft Türkiye

**Açık Akademi**

## Veri İşleme Dili (Data Manipulation Language - DML)

Veri işleme ifadeleri, veri tabanı üzerinde depolanan veriler üzerinde yapılması gereken işlemler için kullanılan ifadelerdir. Veri seçme/getirme, veri ekleme, veri güncelleme ve veri silme gibi işlemlerin yapılmasını sağlarlar. Toplam 4 adet veri işleme ifadesi bulunmaktadır.

- **SELECT**
- **INSERT**
- **UPDATE**
- **DELETE**

İsimlerinden de anlaşılacağı gibi, **SELECT** tablolardan veri seçme/getirme işlemini, **INSERT** tablolara veri ekleme işlemini, **UPDATE** varolan veriler üzerinde değişiklik yapma, güncelleme işlemini, **DELETE** ise varolan verilerin silinmesi işlemini gerçekleştirmektedir.

### SELECT

Bir tablo içerisindeki verilerin tamamını getirir veya belirli şartlara göre bir kısmını filtreleyerek seçme işlemlerini gerçekleştirir. Genel kullanım şekli aşağıdaki gibidir.

```
SELECT SeçilecekAlan1, SeçilecekAlan2 ... FROM TabloAdı
```

Örnek Kullanım: Öğrenci tablosundaki tüm kayıtların ÖğrenciID ve AdSoyad bilgileri aşağıdaki sql ifadesi ile elde edilebilir.

```
SELECT ÖğrenciID, AdSoyad FROM Öğrenci
```

OğrenciID	AdSoyad
115874	Ercan Bozkurt
101502	Melih Koray
100470	Bülent Sözge
496196	Burak Batur

Sorgu sonucunda seçilen öğrencilerin numaraları ve ad-soyadları

### INSERT

Bir tablo içerisine yeni bir veri eklemek için kullanılır. Genel kullanımı aşağıdaki gibidir.

```
INSERT INTO TabloAdı (VeriEkleniecekAlan1, VeriEkleniecekAlan2, ...)  
VALUES (EkleniecekDeğer1, EkleniecekDeğer2, ...)
```

Örnek Kullanım: Öğrenci tablosuna, öğrenci numarası 115874 olan 101 numaralı bölümden Ercan Bozkurt isimli öğrenci, aşağıdaki sql ifadesi yardımıyla eklenebilir.

```
INSERT INTO Öğrenci (ÖğrenciID, AdSoyad, BölümID)  
VALUES (115874, 'Ercan Bozkurt', 101)
```

### UPDATE

Bir tablo içerisinde bulunan verilerin değiştirilmesi için kullanılır. Genel kullanımı aşağıdaki gibidir.

**UPDATE TabloAdı Set GuncellenecekAlan1 = YeniVeri1,  
GuncellenecekAlan2 = YeniVeri2 WHERE Koşul veya koşullar**

Örnek Kullanım: Öğrenci tablosunda öğrenci numarası 115874 olan öğrencinin bölüm kodunu 102 olarak değiştirmek için aşağıdaki sql ifadesinden yararlanılabilir.

**UPDATE Öğrenci SET BolumID = 102  
WHERE ÖğrenciID = 115874**

## **DELETE**

Tablo içerisinde bulunan bir kaydı veya kayıtları silmek için kullanılır. Genel kullanım şekli aşağıdaki gibidir.

**DELETE FROM TabloAdı WHERE Koşul veya koşullar**

Örnek Kullanım: Öğrenci tablosundaki 115874 numarasına sahip öğrencinin kaydını silmek için aşağıdaki sql ifadesinden yararlanılabilir.

**DELETE FROM Öğrenci WHERE ÖğrenciID = 115874**

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Veri Tabanı Oluşturmak**



Microsoft Türkiye

**Açık Akademi**

DML içerisindeki create komutunu kullanarak veritabanı oluşturabileceğin gibi, Management Studio arayüzünü kullanarak da veritabanı oluşturabilirsin.

#### **Kod Kullanarak Veritabanı Oluşturmak**

Management Studio ile veritabanına bağlantı sağladıktan sonra sol üst bölümde yer alan **New Query** seçeneği ile sorguları yazabileceğin sayfayı açabilirsin. Burada “**create database University**” ifadesini yazıp, **F5** kısayolu ile çalıştırarak, varsayılan ayarlarla University isimli bir veritabanı oluşturabilirsin.

#### **Arayüz Üzerinden Veritabanı Oluşturmak**

Management Studio içerisindeki **Object Explorer** bölümünde yer alan **databases** alanına sağ tıklayıp **New Database** seçeneğini seçerek, yeni veritabanı oluşturma penceresini açabilirsin. Sonrasında bu penceredeki **Database name** alanına oluşturmak istediğin veritabanının adını verip, **OK** butonuna tıklaman yeterli olacaktır.

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Attach – Detach İşlemleri**



Microsoft Türkiye

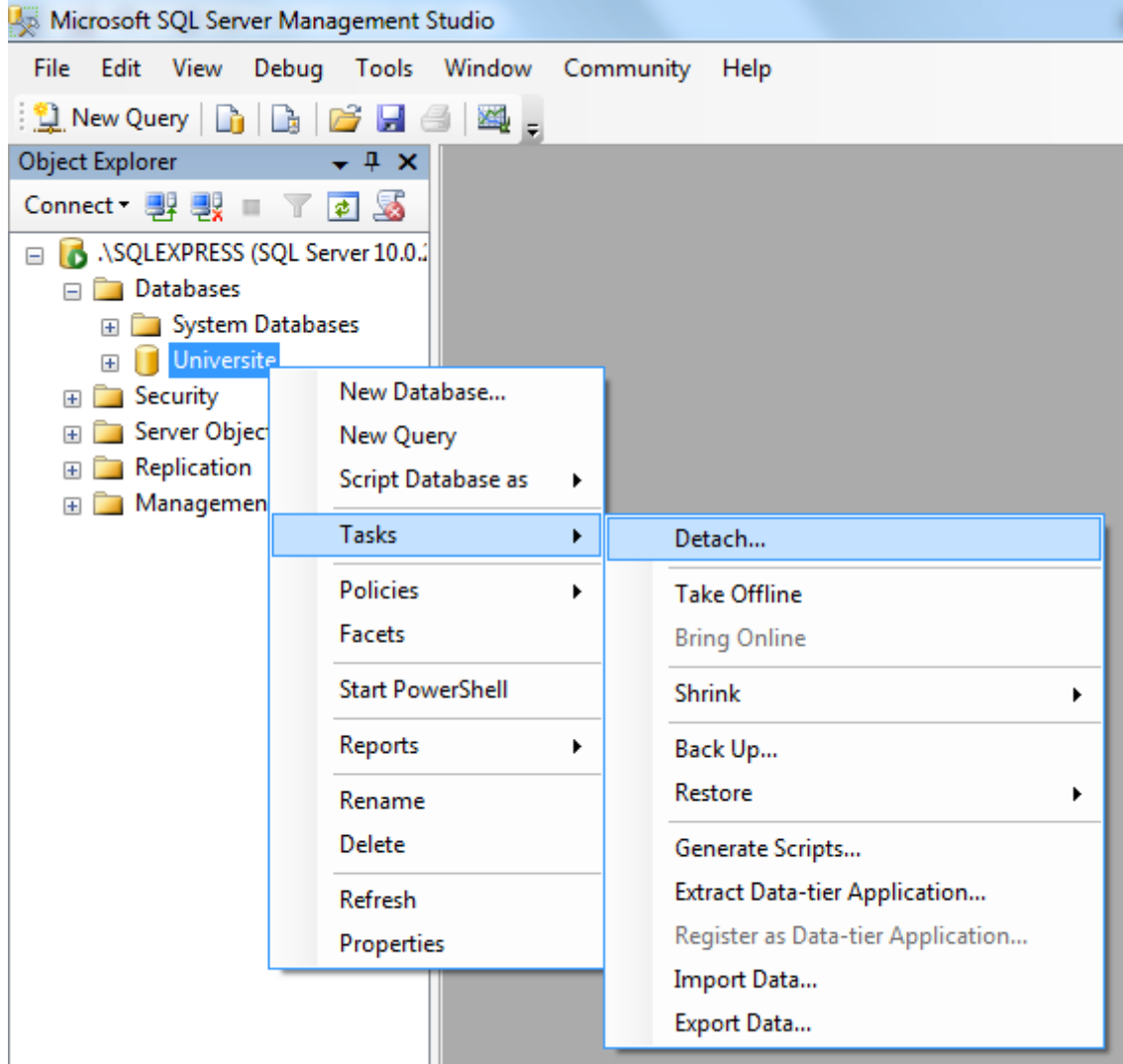
**Açık Akademi**



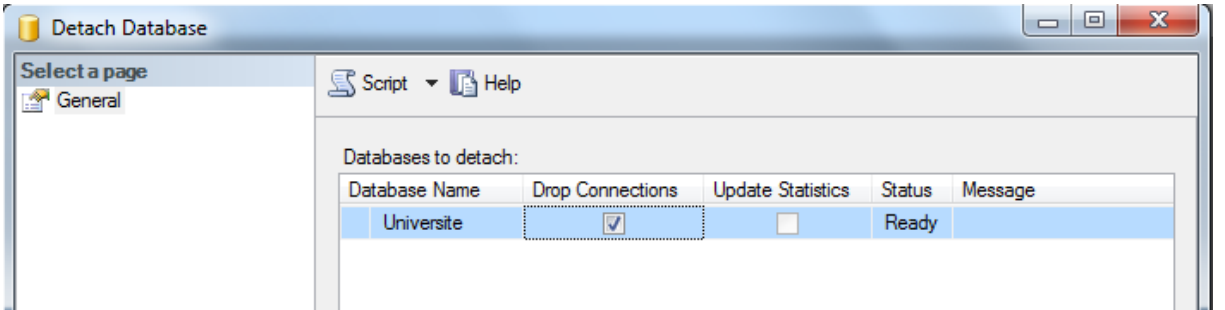
ATTACH ve DETACH komutları bir SQL Server'ın yönetimindeki veritabanını SQL Server'dan ayırmak (DETACH) ya da SQL Server üzerine daha önceden oluşturulmuş bir veritabanını eklemek (ATTACH) amacıyla kullanılırlar.

#### Detach İşlemi

Bir veritabanını SQL Server'dan ayırmak için, Management Studio içerisinde veritabanı adının üzerine sağ tıklayıp, **Tasks** bölümünden **Detach...** seçeneğini seçmelisin.



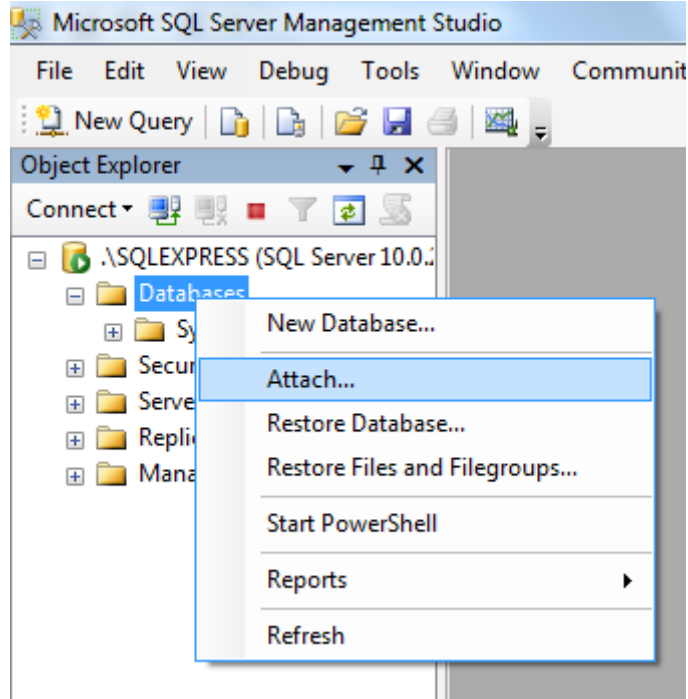
Sonrasında, açılan **Detach Database** ekranında **Drop Connections** seçeneğini işaretleyip **OK** butonuna tıklayarak işlemi tamamlayabilirsin.



Detach Database Ekranı

#### Attach İşlemi

Mevcut bir veritabanı dosyasını SQL Server'a eklemek için, **Object Explorer** içerisindeki **Databases** bölümüne sağ tıklayıp **Attach** seçeneğini seçmelisin.



Sonrasında açılan **Attach Databases** ekranında **Add** butonunu kullanarak, yeni gelen pencerede ekleyeceğin veritabanını seçmelisin. Ardından **OK** tuşu ile mevcut pencereyi kapatıp ilk ekrana dönüyor olacaksın. Burada bir kez daha **OK** tuşunu kullanarak işlemleri tamamlayabilirsin. Eklenen veritabanı, **Object Explorer** içerisinde **Databases** başlığı altında görülebilir.

Şimdi, örneklerde uygulama sırasında kullanılmak üzere hazırlanmış, Üniversite ve Sirket veri tabanlarını SQL Server'a eklemelisin (attach). Bu veri tabanını aşağıdaki adresten indirebilirsin.

<http://www.myenocata.com/enocata/3563/attachments/Universite.zip>

<http://www.myenocata.com/enocata/3563/attachments/Sirket.zip>

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Temel Veri Tipleri**



Microsoft Türkiye

**Açık Akademi**

## T-SQL'de Veri Tipleri

Tablo oluştururken tablo içerisindeki her alanın hangi tipte veri taşıyabileceğini belirleyen bazı veri tipleri bulunmaktadır. T-SQL'de bulunan temel veri tiplerinden bazıları şunlardır:

### Metinsel Veri Tipleri

Tip	Değer Aralığı
<b>char(n)</b>	ASCII türünden ve sabit boyutta veri saklar. En fazla 8000 karakter tutulabilir. (n) alabileceği en fazla karakteri belirler.
<b>nchar(n)</b>	Unicode türünden ve sabit boyutta veri saklar. En fazla 4000 karakter tutulabilir.
<b>varchar(n)</b>	ASCII türünden ve değişken uzunlukta veri saklar. En fazla 8000 karakter tutulabilir.
<b>nvarchar(n)</b>	Unicode türünden ve değişken uzunlukta veri saklar. En fazla 4000 karakter tutulabilir.
<b>varchar(MAX)</b>	varchar veri tipi ile aynı özelliklere sahiptir ve 2 GB'a kadar veri tutabilmektedir.
<b>nvarchar(MAX)</b>	nvarchar veri tipi ile aynı özelliklere sahiptir ve 2 GB'a kadar veri tutabilmektedir.
<b>text</b>	ASCII türünden metin saklamak için kullanılır. 2 GB'a kadar sınırı vardır.
<b>ntext</b>	Unicode türünden metin saklamak için kullanılır. 2 GB'a kadar sınırı vardır.

#### Sql metin veri tipleri

### Sayısal Veri Tipleri

Tip	Değer Aralığı
<b>int</b>	Yaklaşık -2 milyar ile +2 milyar arasındaki tamsayı değerlerini tutar.
<b>bigint</b>	Yaklaşık $-2^{63}$ ile $+2^{63}$ arasındaki tamsayı değerleri tutar.
<b>smallint</b>	Yaklaşık -32 bin ile +32 bin arasındaki tamsayı değerlerini tutar.
<b>tinyint</b>	0-255 arasındaki tamsayı değerlerini tutar.
<b>float(n)</b>	Kayan noktalı sayı değerlerini tutar. $-1.79e+308$ ile $1.79E+308$ arasında değer tutabilir. n, 1 ile 53 arasında değer alabilir. 1 ile 24 arasında olduğunda 7 haneye kadar hassasiyet ve 4 byte yer ayrılması söz konusudur. 25 ile 53 aralığı için ise 15 haneye kadar hassasiyet ve 8 byte yer ayrılması söz konusudur. Varsayılan olarak n değeri 53'tür.
<b>real</b>	$-3.40e+38$ ile $3.40e+38$ arasında değerler alabilir. 7 haneye kadar hassasiyet sunar ve 4 byte yer kaplar. Bu veri tipi float(24)'ün karşılığıdır. Eğer 7 haneye kadar hassasiyet gerekiyorsa real tipi varsayılan float tipi yerine tercih edilebilir.
<b>money</b>	Yaklaşık -922 milyar ile +922 milyar arasındaki değerleri tutar. Bu tip genelde parasal değerlerin tutulacağı alanlarda kullanılır.

#### Sql sayısal veri tipleri

## Tarihsel Veri Tipleri


Tip	Değer Aralığı
<b>datetime</b>	01.01.1753 ile 31.12.9999 arasındaki tarih ve zaman bilgisini tutar.
<b>smalldatetime</b>	01.01.1900 ile 06.06.2079 arasındaki tarih ve zaman bilgisini tutar.
<b>date</b>	01.01.0001 ile 31.12.9999 arasındaki tarih bilgisini tutar.
<b>time</b>	00:00:00.0000000 ile 23:59:59.9999999 arasındaki zaman bilgisini tutar.

### Sql tarihsel veri tipleri

## Diğer Veri Tipleri

Tip	Değer Aralığı
<b>bit</b>	Boolean değerler tutmak için kullanılan veri tipidir. Sadece 1 veya 0 değerlerini alabilir. 1 True, 0 False değerlerini temsil eder.
<b>image</b>	Resim dosyalarının veri tabanında tutulması için kullanılan veri tipidir. 2 GB'a kadar resim dosyası tutabilmektedir.
<b>xml</b>	XML dosyalarını ve XML kodlarını saklayabilen veri tipidir. 2 GB'a kadar veri taşıyabilir.
<b>binary(n)</b>	Sabit uzunluktaki binary veriyi tutmak için kullanılır. Maksimum uzunluğu 8000 byte' tır. Varsayılan uzunluğu ise 1 byte' tır.
<b>varbinary(n)</b>	Değişken uzunlukta binary veriyi tutmak için kullanılır. Maksimum uzunluğu 8000 byte' tır. Varsayılan uzunluğu ise 1 byte' tır.
<b>varbinary(MAX)</b>	Maksimum 2 GB binary veriyi tutabilen veri tipidir.

### Sql diğer veri tipleri

	Microsoft, Sql Server'ın 2005 sürümünden itibaren, image, text ve ntext veri tipleri yerine varbinary(MAX), varchar(MAX), nvarchar(MAX) türlerinin kullanılmasını tavsiye etmektedir.
---	---

Veri tipleri ile ilgili daha detaylı bilgi almak için aşağıdaki MSDN linkini kullanabilirsiniz.  
<http://msdn.microsoft.com/en-us/library/ms187752.aspx>

**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Tablo Oluşturmak**



Microsoft Türkiye

**Açık Akademi**

## Tablolar (Tables)

Tablolar verilerin saklanması sağlayan, **alanlar (sütun-column)** ve **satırlardan (row)** oluşan birimlerdir. Tablo, ilişkisel veritabanlarında temel veri depolama nesnesidir ve bilgiler tablolarda saklanmaktadır.

Tablo içerisinde her satır bir kaydı temsil etmektedir. Alanlar ise kayıtlara ait özellikleri taşımaktadırlar. Örneğin bir öğrenciye ait kayıta okul numarası, öğrenci adı ve öğrenci soyadı bilgilerini temsil eden “115874”, “Ercan”, “Bozkurt” gibi bilgiler tutulabilmektedir. Burada öğrenci numarası, öğrenci adı ve soyadı birer alanı, bu üç verinin tamamı ise bir satırı, yani bir kaydı oluşturmaktadır. (Aşağıdaki şekilde bir tablo üzerinde satır ve alan gösterilmiştir.) Bir tabloda *en az bir tane alan* bulunması gerekmektedir. Tabloda satırlar olabileceği gibi hiçbir satır olmayabilir de. Yani tablomuzda hiç veri taşımayacağımız durumlar olabilir. Tablo içerisinde kayıtlara ait bilgileri tutacak olan alanların hangi veri tipinde veri saklayabileceği de belirlenebilmektedir. Yukarıdaki öğrenci örneğindeki kayıtlarda, öğrenci adı ve soyadı bilgileri metinsel bir ifade, öğrenci numarası da rakamsal bir ifade içersin gibi...

OğrenciID	Ad	Soyad	BolumID
100470	Bülent	Sözge	102
101502	Melih	Koray	103
115874	Ercan	Bozkurt	101
496196	Burak	Batur	101

Bir tablo üzerindeki satırlar ve alanlar

Yukarıdaki şekilde 1 numara ile belirtilmiş kısım satır, 2 numara ile belirtilmiş kısım ise alan olarak adlandırılır.

1. **Satır (Row):** Tabloda bulunan bir kayıt.
2. **Alan (Sütun-Column):** Bir kayda ait özellik. Örneğin; Kayıtlı öğrencinin adı.

Tablolar tasarlanırken içerecekleri bilgi türlerine göre gruplandırılmalıdır. Daha önce kısaca ele aldığımız örneği düşünecek olursak, bir üniversitenin veri tabanı Fakülte, Bölüm ve Öğrenci gibi tablolardan oluşabilir. Bölüm ve öğrenci bilgileri birbirinden farklı şeyler olduğu için, ayrı tablolarda tutulması hem veri bütünlüğü açısından, hem de veriye erişimin daha kolay ve hızlı olması açısından önemlidir. İsterseniz şimdi örnek olarak Bölüm ve Öğrenci tablolarını oluşturalım. Ardından da bu tabloların yapılarını ve içeriklerini inceleyelim.

Bolum Tablosu

BolumID	BolumAd
10	Bilgisayar Mühendisliği
10	Bilgisayar Öğretmenliği
10	İç Mimarlık
10	Fizik

Oğrenci Tablosu

OgrenciID	Ad	Soyad	BolumID
100470	Bülent	Sözge	102
101502	Melih	Koray	103
115874	Ercan	Bozkurt	101
496196	Burak	Batur	101

**Bolum ve Ogrenci tabloları ve içerdği bilgiler**

**Bolum** tablosu 2 alan (sütun) ve 4 satırdan (kayıt), **Ogrenci** tablosu ise 4 alan ve 4 satırdan oluşmaktadır. Bu şekilde birbirinden farklı içerikleri olan bölümler ve öğrenciler farklı iki tabloda depolanmış oldu. Yine **Ogrenci** adlı tablodaki **BolumId** alanına dikkat edilecek olursa, öğrencinin hangi bölümde bulunduğu bilgisinin **Bolum** tablosundaki **BolumId** ile ilişkilendirilmiş olduğu görülebilir. (Buradaki 101, 102 şeklindeki ifadeler aslında **Bolum** tablosundaki Bilgisayar Mühendisliği, Bilgisayar Öğretmenliği gibi bölümleri temsil etmektedir.) **Bolum** tablosunda BolumId ve BolumAd dışında, bölüm adresi, bölüm başkanı, bölümün bağlı olduğu fakülte gibi bilgileri de tuttuğumuzu varsayarsak, bu bilgileri öğrencilerin bilgileri ile aynı tabloda tutmak hem gereksiz yere verilerin tekrarlanmasına, hem de veri karışıklığına sebep olacaktı.

İlişki diyagramlarında bir tablo ile diğer tablo arasındaki ilişki aşağıdaki şekilde olduğu gibi belirtilir.



**Bir öğrenci ve bir bölüm arasındaki ilişkinin şekille gösterimi**

Tablo ve alan isimlendirilirken dikkat etmemiz gereken bazı hususlar vardır.

- Microsoft SQL Server 2005'ten önceki versiyonlarda aynı veri tabanı içerisinde aynı isme sahip sadece bir tablo bulunabilirken, Microsoft SQL Server 2005 ile gelen şema (schema) kavramı sayesinde bu kısıt aşılabilmektedir. Bununla birlikte, bir tablo içerisinde aynı isme sahip sadece bir alan bulunabilir.
- Tablo ve alan isimleri içerisinde rakam veya harfler dışındaki karakterlerin kullanılmaması tavsiye edilmektedir. (\*)
- İsimlerin rakam ile başlamaması tavsiye edilmektedir. (\*)

(\*) Tavsiye edilmeyen durumlarda kullanılacak isimlerin ancak [ ] işaretleri arasına alınarak kullanılmasına izin verilmektedir.

## Tablo oluşturmak

Management Studio arayüzünü kullanarak tablo oluşturmak için **Databases** başlığı altındaki **Tables** bölümüne sağ tıklayıp **New Table** seçeneğini seçebilirsin. Açılan ekranda kolon adları , veri tipleri ve ilgili kolona değer girilmemesine (null) izin verilip verilmemesi gibi tanımlamaları yaptıktan sonra , üstteki menüden **Save** (disket ikonu) seçeneği ile tabloyu oluşturabilirsin. Save butonuna bastığında açılan ekrana tabloya vermek istediğin adı yazmalısın.



**EĞİTİM :**

**VERİ TABANINA GİRİŞ VE  
TEMEL VERİ TABANI  
KAVRAMLARI**

**Bölüm :**

**Veri Tabanına Giriş**

**Konu :**

**Anahtarlar**



Microsoft Türkiye

**Açık Akademi**

## Anahtarlar (Keys)

Bir kayıt içerisinde farklılıkları ve nitelikleri gösteren belirleyicilere **anahtarlar (keys)** denir. Farklı içeriklere sahip olacak verileri farklı tablolarda depolayarak yapabileceğimiz birçok işi kolaylaştırabiliyorduk. Benzer şekilde, tablodaki kayıtları da birbirinden ayırt edebilmek için tablo içindeki alanlara belirli anahtarlar atayarak birçok işlemi kolaylaştırabilmekteyiz. Bir tablo içerisinde bulunabilecek anahtarlar, **birincil anahtar** (primary key), **tekil anahtar** (unique key), **referans anahtar** (foreign key) ve **birleşik anahtar** (composite key)

**Birincil anahtar (Primary key):** Bir tablo içerisindeki satırları birbirinden ayırt eder. Birincil anahtar olan bir veri aynı tablo içerisinde tekrarlanamaz. Yine bu alandaki veri boş bırakılamaz, yani **NULL** değeri alamaz. Tek bir alan birincil anahtar olabileceği gibi bazı tablolarda birden fazla alanın birleşmesiyle birincil anahtar oluşabilir. (Bu aslında az sonra göreceğimiz birleşik anahtardır)

**Tekil anahtar (Unique key):** Tablonun tekil anahtar olarak tanımlanmış bir alanına aynı değer sadece bir kez girilebilir. Birincil anahtardan farklı olarak, tabloda bu alana ait sadece bir kayıt **NULL** değeri alabilir. Birincil anahtar aynı zamanda tek anahtar olarak sayılabilir fakat tek anahtarlar birincil anahtar değildirler.

**Referans anahtar (Foreign key):** Tablodaki bir veriyi başka tablodaki bir veri ile ilişkilendirir. İki tablo arasında yapılan bu ilişkilendirme ile referans anahtar olarak tanımlanmış alana sadece ilişkilendirdiği tablonun alanındaki veriler eklenebilir.

**Birleşik anahtar (Composite Key):** Birden fazla alanın birleştirilmesiyle birincil anahtar görevini üstlenecek tanımlamalar yapılabilir. Bunlar birleşik anahtar olarak adlandırılır.

Ogrenci Tablosu					
OgrenciID	Ad	Soyad	BolumID	Sehir	EMail
100470	Bülent	Sözge	102	Ankara	bulent.sozge@tcm.com.tr
101502	Melih	Koray	103	İstanbul	ben@melihkoray.com
115874	Ercan	Bozkurt	101	İstanbul	ebozkurt@amigadunyasi.com
496196	Burak	Batur	101	Denizli	NULL
↓			↓		↓
Primary key			Foreign key		Unique key

**Ogrenci tablosu üzerinde bulunan anahtarlar**

**OgrenciID** her öğrenci için tek ve belirleyici bir unsur olacağı için **Ogrenci** tablosunda *birincil anahtar* olarak belirlenmiştir. Yine **EMail** her öğrenci için tek olacağı için bu alan tablo için *tekil anahtar* olacaktır. Daha önceki örneğimizden hatırlayacağınız gibi **BolumID**, **Bolum** tablosunun **BolumID** alanı ile ilişkilendirilmiştir. **Ogrenci** tablosunun bu alanına sadece **Bolum** tablosundaki değerleri alabilmektedir. Başka bir tablodaki anahtar ile ilişkilendirildiği için tablonun *referans anahtarı* olmuştur.