Veri Merkezli Uygulamalar ve ADO.NET'e Genel Bakış*

Öğr.Gör.Erkan HÜRNALI

*Çeşitli kaynaklardan derlenmiştir.

T.C. ANKARA ÜNİVERSİTESİ
MYO
elmadağ meslek yüksekokulu

Veri (Data)

Sayısal veya mantıksal her türlü değer bir veridir. Verinin işlenmiş ve bir anlam ifade eden hâline ise bilgi adı verilir. Örneğin, saatte yelkovanın 12 üzerinde olması bir veridir. Aynı şekilde akrebin 4 üzerinde olması bir veridir. Bu iki verinin yorumlanmasıyla elde edilen "saat dörttür" ifadesi bir bilgidir.

Bir bilgi farklı durumlarda veri olarak kullanılabilir. Örneğin, bir arkadaşınızla buluşma saatinizin üç veya dört olması birer veriyken "Buluşmaya 1 saat geç kaldım" ifadesi bir bilgidir. Görüldüğü gibi "saatin dört olması" bir önceki örnekte bilgiyken bu örnekte bir veridir.

Bir kişi veya uygulama için gerekli olan bir veri veya bilgi diğer kişi veya uygulamalar için gerekli olmayabilir. Örneğin, bir alışveriş sitesi birçok kişisel bilgiye ihtiyaç duyarken bir sohbet sitesi sadece rumuz isteyip diğer bilgileri istemeyebilir.

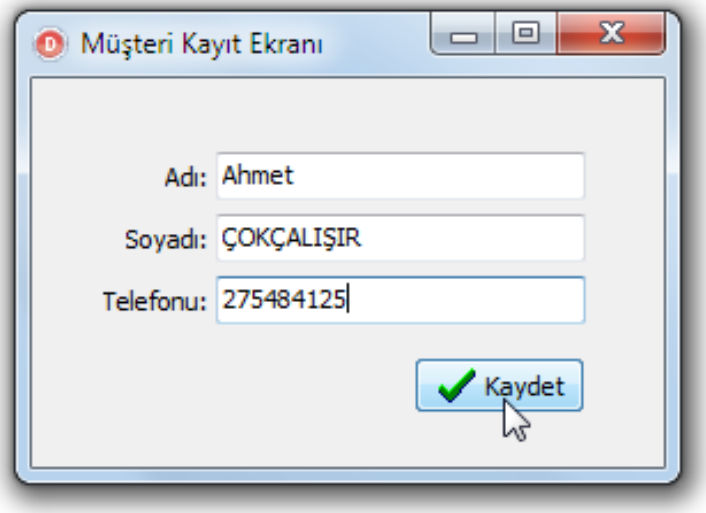
Veri yapıları farklı şekillerde oluşturulabilir. Bir uygulamada ad soyad bilgileri birlikte tutulurken diğer bir uygulamada iki bilgi de ayrı ayrı tutulabilir. Bir başka örnek ise adres bilgilerinin tutulmasıdır. Bir uygulamada adres tek bir alanda tutulurken diğer bir uygulamada cadde, sokak, apartman nu, daire nu, ilçe, il bilgileri ayrı alanlarda tutulabilir.

Neden Veritabanı?

Verileri .... saklamak

- Güvenli
- Performanslı
- Kalıcı

Neden Veritabanı?

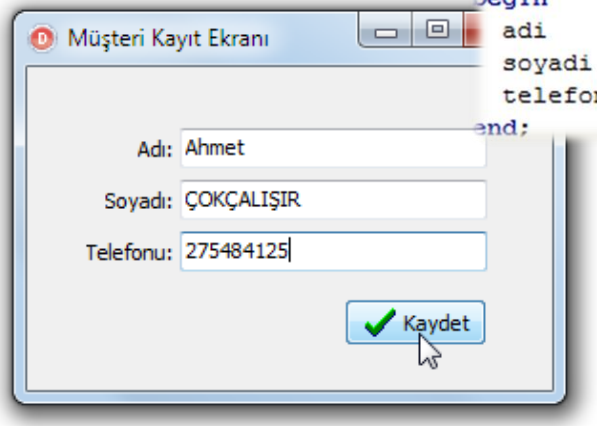


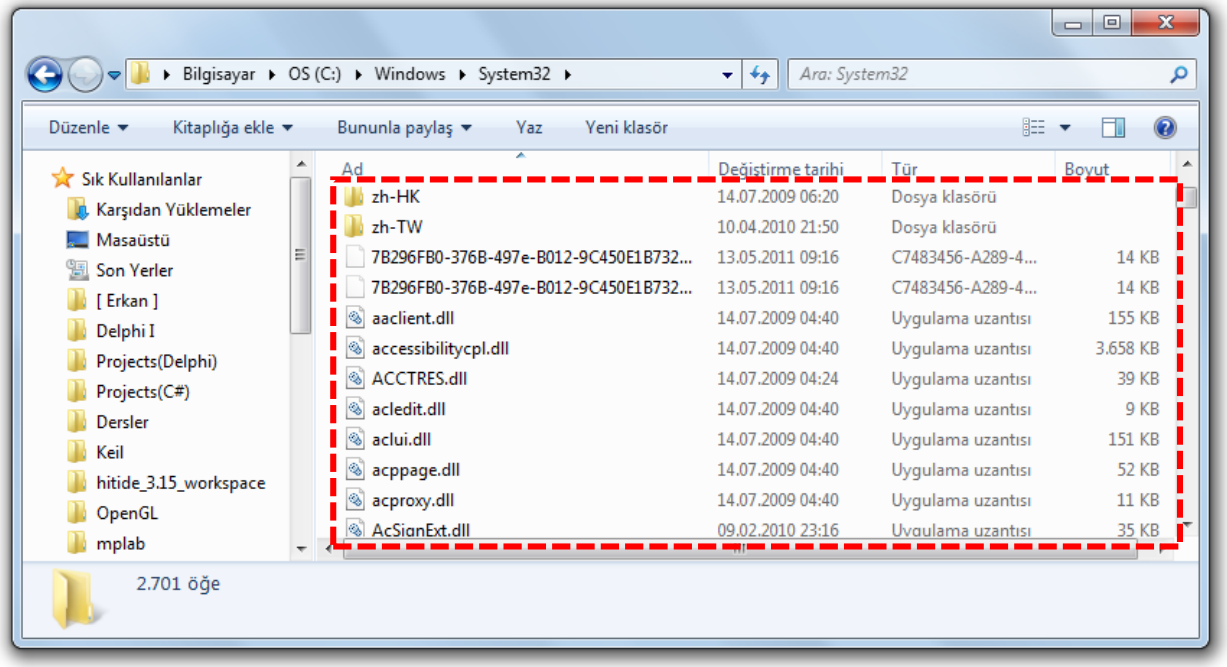

```
procedure TForm1.btnKaydetClick(Sender: TObject);
var
  adi, soyadi, telefonu: string;
begin
  adi       := Edit1.Text;
  soyadi    := Edit2.Text;
  telefonu  := Edit3.text;
end;
```

Bu şekilde de veriler güvenli ve performanslı bir şekilde saklanmış olmaz mı?

Neden Veritabanı?

```
procedure TForm1.btnKaydetClick(Sender: TObject);
var
  adi, soyadi, telefonu: string;
begin
  adi       := Edit1.Text;
  soyadi    := Edit2.Text;
  telefonu  := Edit3.text;
end;
```





VTYS (DBMS)

Veritabanı

Veri tabanları birbirleriyle ilişkili bilgilerin depolandığı alanlardır. Bilgi artışıyla birlikte bilgisayarda bilgi depolama ve bilgiye erişim konularında yeni yöntemlere ihtiyaç duyulmuştur. Veri tabanları; büyük miktardaki bilgileri depolamada geleneksel yöntem olan ‘’dosya-işlem sistemine’’ alternatif olarak geliştirilmiştir.

Telefonlarımızdaki kişi rehberi günlük hayatımızda çok basit bir şekilde kullandığımız veri tabanı örneği olarak kabul edilebilir. Bunların dışında internet sitelerindeki üyelik sistemleri, akademik dergilerin ve üniversitelerin tez yönetim sistemleri de veri tabanı kullanımına örnektir. Veri tabanları sayesinde bilgilere ulaşır ve onları düzenleyebiliriz. Veritabanları genellikle bireysel olarak satın alınamayacak kadar yüksek meblağlara sahip olmasına karşın; ücretsiz kullanıma açılan akademik veri tabanları da bulunmaktadır. Akademik veri tabanları aracılığıyla bazen bibliyografik bilgi bazen de tam metinlere erişmek mümkündür. Veri tabanları, veri tabanı yönetim sistemleri aracılığıyla oluşturulur ve yönetilir. Bu sistemlere; Microsoft Access, MySQL, IBM DB2, Informix, Microsoft SQL Server, PostgreSQL, Oracle, Interbase ve Sysbase örnek olarak verilebilir.

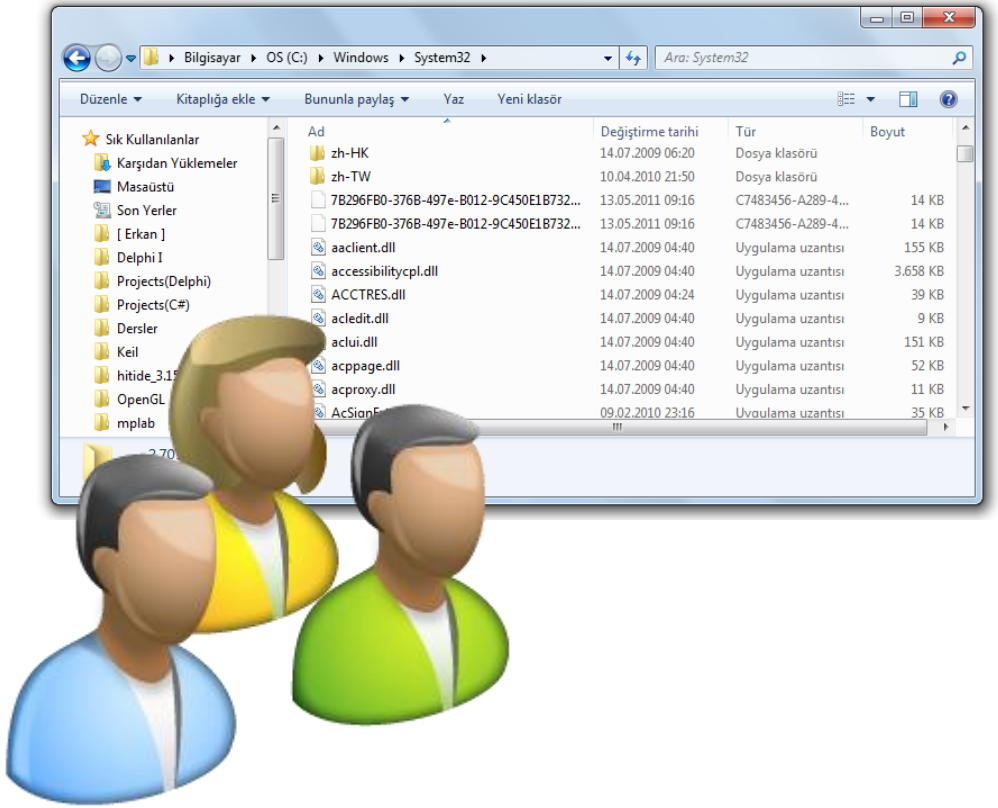
Veritabanı Yönetim Sistemleri

Veritabanı Yönetim Sistemleri

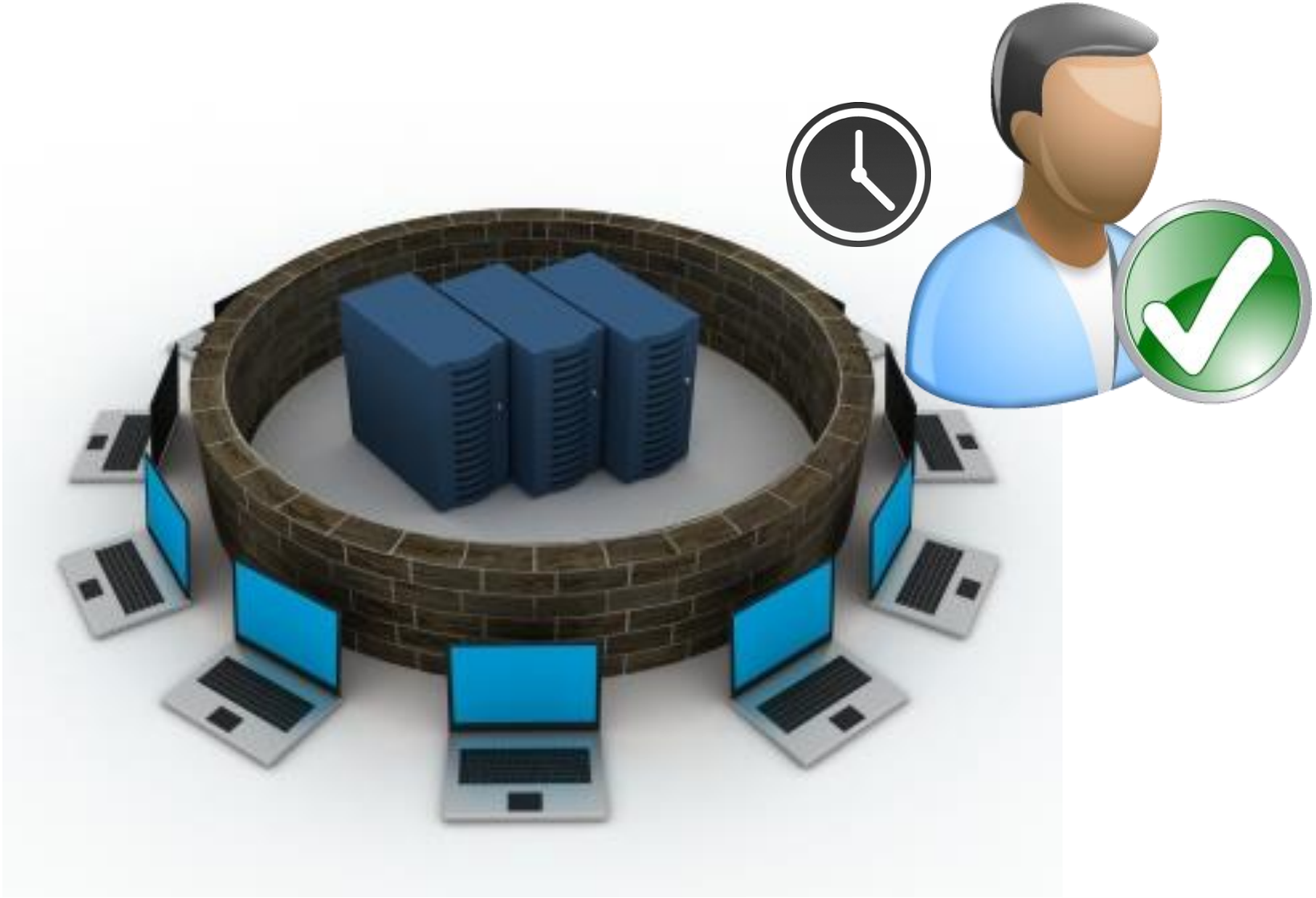
Veri tabanı yönetim sistemi (**VTYS**, İngilizce: *Database Management System*, kısaca **DBMS**), veri tabanlarını tanımlamak, yaratmak, kullanmak, değiştirmek ve veri tabanı sistemleri ile ilgili her türlü işletimsel gereksinimleri karşılamak için tasarlanmış sistem ve yazılımdır.

Neden Veritabanı Yönetim Sistemleri?

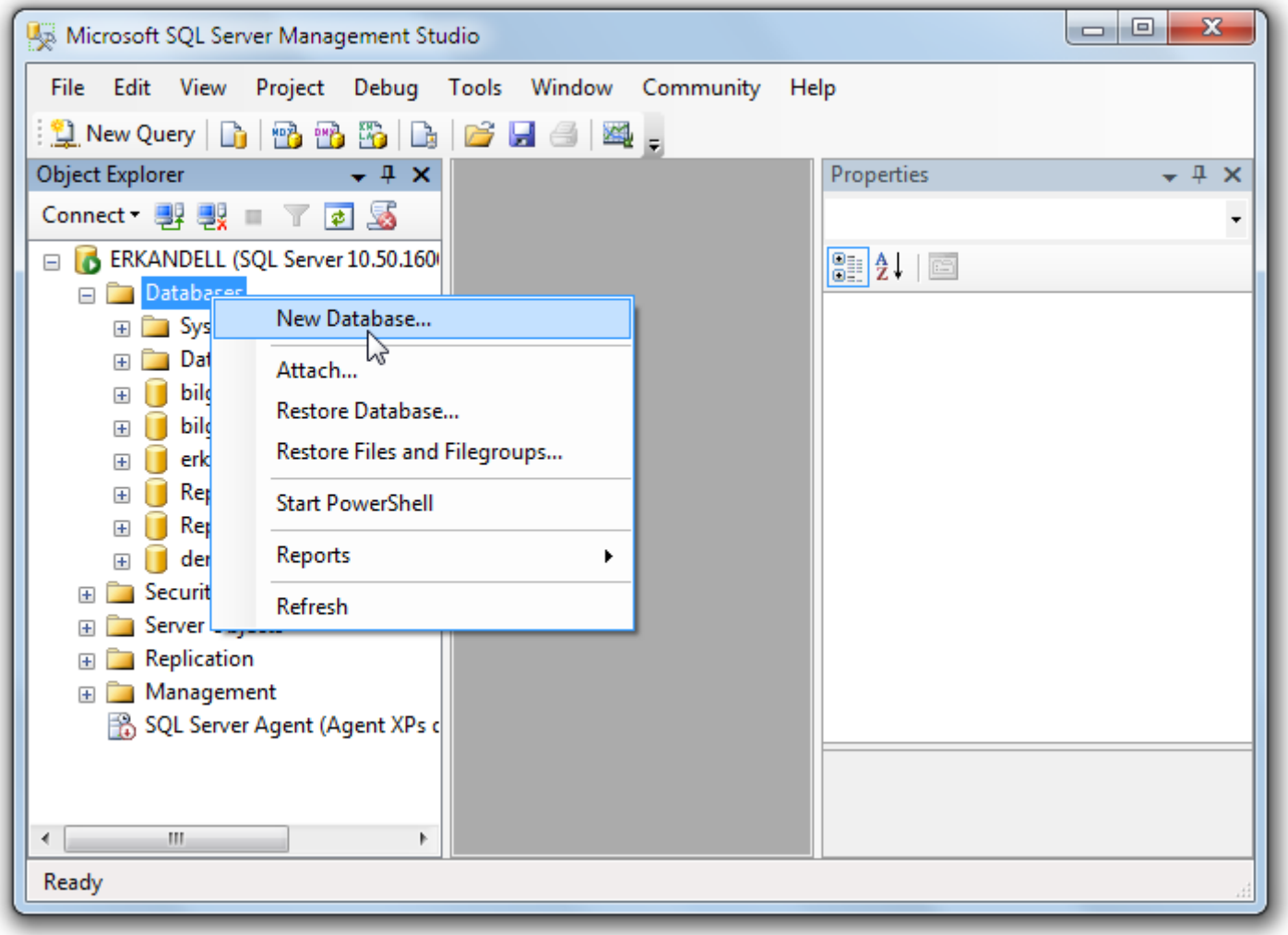
Neden Veritabanı Yönetim Sistemleri?



Neden Veritabanı Yönetim Sistemleri?

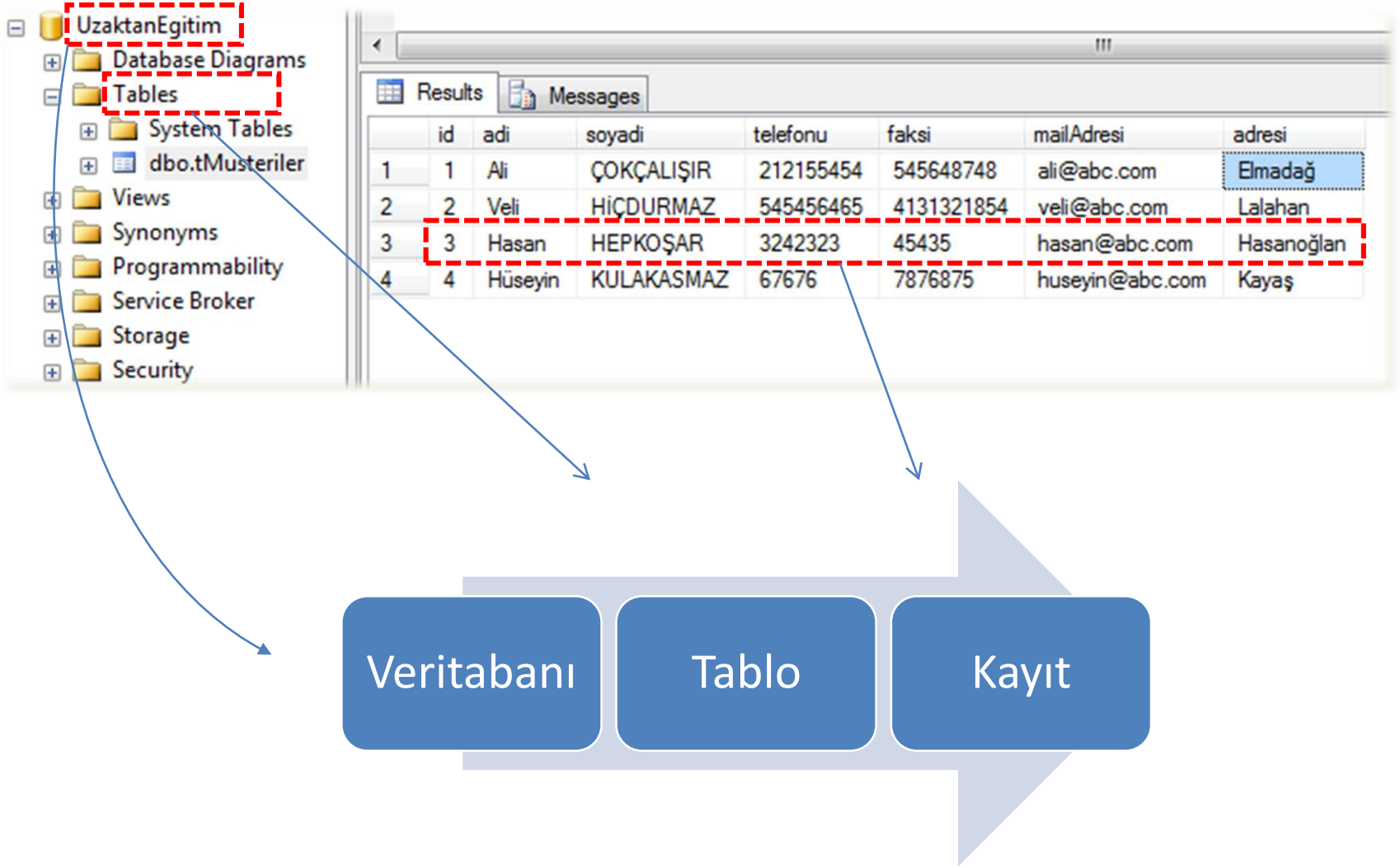


SQL Server

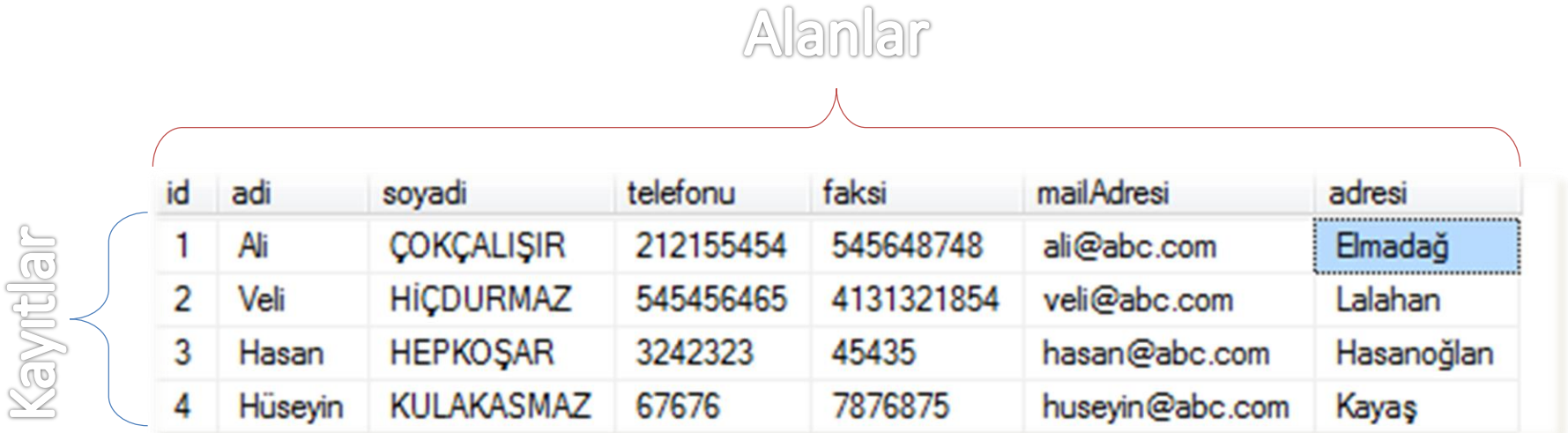

Microsoft SQL Server Management Studio
File Edit View Project Debug Tools Window Community Help
New Query
Object Explorer
Connect
ERKANDELL (SQL Server 10.50.160
Databases
New Database...
Attach...
Restore Database...
Restore Files and Filegroups...
Start PowerShell
Reports
Refresh
Replication
Management
SQL Server Agent (Agent XPs
Properties
Ready

Nasıl Bir Organizasyon?

Nasıl Bir Organizasyon?



Nasıl Bir Organizasyon?



Toplam 4 kayıt bulunmaktadır.
Son kaydın «soyadi» alanında ...? yazmaktadır.

Veri Merkezli Uygulamalar

Günümüz uygulamaları ister masaüstü ister web tabanlı olsun mutlaka bir veri tabanı (database) yazılımı kullanmak zorundadırlar. Çünkü çağımızda bilgi ve bilgiye erişim çok önemli bir rol oynamaktadır. Bir amaç etrafında verileri topluyoruz, sorguluyoruz, bir sonuca ulaşmak için onları işleyip raporlar elde ediyoruz. Çünkü geleceği tahmin etmek istiyoruz. Bu sebepten dolayı bilgisayar programları da, çeşitli bilgileri saklamak, düzenlemek ve tekrar saklamak zorundadır.

Uygulama geliştiriciler ise bu veriler etrafında her zamankinden daha hızlı, esnek, ölçeklenebilir, yüksek erişilebilirlik değerlerine sahip ve performanslı çözümler oluşturmak durumundadırlar. Hazırladıkları yazılımların günün şartlarına uygun olması, kurumsal platformlarda aranan daha fazla esneklik ve daha yüksek performans özellikleri gösteren mimariye sahip olması ilk akla gelen "olmazsa olmaz" şartlardan sadece birkaçıdır. Bunlara ek olarak hazırlanan uygulamanın günümüz standartlarına hitap etmesi iyi bir yazılım olma adına önemli bir gereksinimdir.

Veri Merkezli Uygulamalar

İşte bu gereksinimlerin karşılığı olarak .Net uygulama geliştiricilere ADO.NET adı verilen ve uygulamanız ile sahip olduğunuz verilerin saklandığı veri tabanı sisteminize erişimde bulunmanıza ve üzerinde işlem yapmanıza olanak sağlayan bir teknoloji sunmaktadır. ADO.NET (ActiveX Data Object) eski nesil ADO veri erişim teknolojisinin .NET için yeniden tasarlanmış halidir. .NET Framework'un bir parçası olarak 1.0 sürümünden beri önemli bir yere sahiptir.

Veri erişim yöntem ve teknikleri değişen teknoloji ve yeni nesil programlama paradigmaları ile birlikte sürekli olarak değişim göstererek MS SQL Server, Oracle, MySQL gibi pek çok ilişkisel veri tabanı yönetim sistemine (RDBMS) bağlanabilmeyi ve nesne yönelimli mimarisi ile verilerinizi uygulamanız etrafında kolayca yönetebilmeyi ve bütün bunları XML standartlarında gerçekleştirebilmeyi mümkün kılmaktadır.

Veri Merkezli Uygulamalar

Geçmişteki veri erişim teknolojileri ile ODBC (Open Database Connectivity), DAO (Data Access Objects), RDO (Remote Data Objects), OLE DB (Object Linking and Embedding DataBase), ADO (ActiveX Data Objects), teknolojilerini sunan Microsoft günümüz gereksinimlerini karşılamakta ADO.NET'i uygulama geliştiricilerin kullanımına sunmaktadır. Ancak uygulamaların giderek artan kod satırları, Windows uygulamaları, web servisleri, web uygulamaları gibi konularda programcılara daha üstün hizmet vermek ve daha hızlı üretimlerde bulunmalarını sağlamak maksadıyla uygulama geliştiricilere ADO.NET temelinde .NET kodunu SQL deyimlerine dönüştürebilen DLINQ (Linq to SQL) ve sonrasında veri tabanınızı nesne yönelimli olarak yönetebilmenize olanak sağlayan "EntityFramework" çatısını uygulama geliştiricilere bir araç olarak sunmaktadır.

Veri Merkezli Uygulamalar

Aslında veri kaynağından bilgi gösterme ADO.NET ile gelen bir yenilik değildir, bu tip uygulamalar oldukça eskiye dayanmaktadır. İlk uygulamaların yazıldığı 60'lardan itibaren yazılmış çoğu uygulamada en az bir adet veri kaynağı kullanıldığını görebiliriz.

Veri kaynaklarında veri tutmanın avantajı adından da anlaşılacağı gibi uygulamayı kullanan firma için çok değerli olan bilgilerin bir kaynakta uzun zaman saklanabilmesidir. Hatta veri kaynağı olarak ileri seviye veritabanı sunucuları kullanarak verilerin kaybolması ihtimali sıfıra yakın olabilmektedir.

Ne yazık ki sıfır veri kaybı ihtimali yoktur. En güvenli sistemlerde bile veri kaybolma olasılığı vardır

Veri Merkezli Uygulamalar

Veri kaynaklarında veri tutmanın diğer bir avantajı; veri ve uygulama katmanı ayrı olduğu için verilerin başka uygulamalar da kullanabilmesidir (Reusability). Ayrıca ilişkisel veritabanların özelliği olan triggers , stored procedure ve relations'larla kod ile yazması oldukça zahmetli olan bazı işlemler veritananı katmanında yapıldığı için yazılan kod miktarı oldukça azalmaktadır. Bu da uygulamanın bakım işini kolaylaştırmaktadır.

Veri Depolama

Günümüzde verileri saklamak için çeşitli teknikler kullanılır. Örneğin bir emlakçı emlak alım, satım bilgilerini dosya kâğıtları üzerinde depolayabilir. Bu yöntem veri arama ve listeleme işlemlerinin karmaşık hale gelmesine ve arama süresinin uzamasına sebep olur. Hatta daha büyük organizasyonlarda işlemlerin yavaşlamasına ve durmasına sebep olabilir.

Artan ihtiyaçlar doğrultusunda veri depolamak ve depolanan veriye erişmek için çeşitli veri depolama yöntemleri geliştirilmiştir. Bu yöntemler:

Veri Depolama

- **Yapısal Olmayan:** Bu yöntem ile depolanan veriler için belirli bir sınıflandırma ve sıralama yoktur. Veriler düz bir şekilde kaydedilir. Örneğin basit not dosyaları.
- **Yapısal:** Bu yöntem ile depolanan veriler çeşitli gruplara ayrılarak saklanır fakat bu gruplar arasında bir alt-üst ayrımı yapılmaz. Örneğin virgülle ayrılmış dosyalar (csv), Excel belgeleri.
- **Hiyerarşik:** Hiyerarşik depolama yöntemini ağaç yapısına benzetebiliriz. Bu yöntemde veriler çeşitli kategorilere bölünerek depolanır. Her bir kategorinin içerisinde alt kategorilerde olabilir. Örneğin XML (eXtensible Markup Language) dosyalar.

Veri Depolama

- **İlişkisel Veritabanı:** İlişkisel veritabanlarında veriler tablolar üzerinde depolanır. Tablo içerisindeki her bir satır kaydı, her bir sütun ise veriyi ifade eder. Örneğin SQL Server, Oracle, Access.
- **Nesne Yönelimli Veritabanı:** En gelişmiş veri depolama yöntemidir. Bu yöntemde veriler; ihtiyaca göre gruplandırılarak, nesneler içerisinde saklanır. Örneğin Versant, AOL

ADO.NET bu depolama tekniklerinin tümünü destekler.

Bağlantılı (Connected) Veri Ortamları

- Avantajları
  - En güvenli veri ortamı
  - Erişimler eş zamanlı
- Dezavantajları
  - Sabit bir ağ bağlantısı gerektirir.
  - Ağ trafiğinin yoğunluğunu arttırır.

Bağlantılı (Connected) Veri Ortamları

Bağlantılı veri ortamları, uygulamaların veri kaynağına sürekli bağlı kaldığı ortamlardır. Bu ortamlarda veri alma ve değiştirme işlemleri uygulama ile veri kaynağı arasında bağlantı kurulduktan sonra gerçekleştirilir. Bağlantılı veri ortamlarında, veri işlemleri gerçekleştiği sürece bağlantı açık kalır.

İlk bilgisayar üretiminden bugüne en çok tercih edilen yöntem bağlantılı veri ortamları olmuştur. Bağlantılı ortamlar veriye erişmek için birçok avantaj sağlar.

Bağlantılı (Connected) Veri Ortamları

Avantajları:

- En güvenli veri ortamıdır.
- Veri kaynağına yapılan eş zamanlı erişimlerde, veri kaynağının kontrolünü kolaylaştırır.

Dezavantajları:

- Uygulama ile veri kaynağı arasında gerçekleşen bağlantıyı koruyabilmek için sabit bir ağ bağlantısının olması gerekir.
- Uygulama ile veri kaynağı arasındaki bağlantı ağ üzerinden gerçekleştiği için, ağ trafiğinin yoğunluğunu artırır.

Bağlantısız (Disconnected) Veri Ortamları

Bağlantı, veri alış verişi yapılırken açılır, işlem bittikten sonra kapatılır .

- Avantajları
  - Taşınabilir aygıtlarla girilen veriler, istenilen zamanda veri ortamlarına aktarılabilir.
  - Uygulama performansını arttırır.

- Dezavantajları
  - Verinin güncelliği sağlanmalıdır.
  - Veri çakışmaları önlenmelidir.

Bağlantısız (Disconnected) Veri Ortamları

Bağlantısız veri ortamı, uygulamanın veri kaynağına sürekli bağlı kalmadığı veri ortamıdır. Uygulama ile veri kaynağı arasında bağlantı, veri alış verişi yapılırken açılır ve işlem bittikten sonra kapatılır. Bu veri ortamları çevrimdışı çalışmak için kullanılır.

Teknolojinin ilerlemesi ve veri depolayan araçların taşınabilirliğinin sağlanması ile tüm dünyada çevrimdışı ortamlara duyulan ihtiyaç artmıştır. Laptop, Notebook ve Pocket PC gibi araçların yaygınlaşması ile günümüzde uygulamanın veri kaynağına bağlı olmadığı durumlarda bile veri girişi yapılabilir.

Uygulamada sadece çevrimiçi veya çevrimdışı ortamlardan birini seçmek yeterli olmayabilir. Gelişmiş uygulamalarda her iki ortamın avantajlarını birleştiren bir çözüm tercih edilebilir.

Bağlantısız (Disconnected) Veri Ortamları

Avantajları:

- Laptop, Notebook ve Pocket PC gibi araçlarla girilen veriler, istenilen zamanda veri ortamlarına aktarılabilir.
- Çevrimdışı ortamlar sayesinde, verilerin depolandığı uygulama üzerindeki yük hafifletilir. Bu durum performans artışını sağlar.

Dezavantajları:

- Bağlantısız veri ortamlarında, verilerin güncel kalmasına dikkat edilmelidir. Bu ortamlarda veri güncelleme işlemleri farklı zamanlarda gerçekleştirilebilir. Veri üzerinde yapılan bu değişimlerin, diğer kullanıcılara gösterilebilmesi için çeşitli çözümler geliştirilmelidir.
- Bağlantısız veri ortamları içerisinde farklı kullanıcılar eşzamanlı güncelleme işlemleri gerçekleştirebilir. Bu durumda oluşacak veri çakışmalarının engellenmesi gerekir.

Veri Erişim Teknolojileri

Veri Kaynağı?

Veri Kaynağı (Data Source)

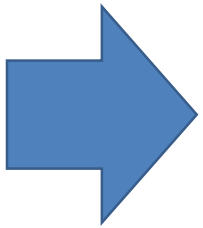
Uygulamalar kullanacakları verileri çeşitli veri kaynaklarından alır. Bu veri kaynağı bir metin (**text**) dosyası, Access gibi bir veritabanı programı veya XML belgesi olabilir. Her veri kaynağı, verileri kendine özel bir düzen içinde saklar. Bir metin dosyasında veriler satır ve sütunlardan oluşan bir düzen içinde tutulur.

Veri Kaynağı (Data Source)

Yukarıdaki veriler bir XML belgesinde aşağıdaki gösterildiği gibi daha farklı bir düzen içinde tutulmaktadır.

Veri Kaynağı (Data Source)

Ad	Soyad	Meslek
Seda	Çiçek	Avukat
Eda	Güneş	Muhasebeci
Simge	Mavi	Mühendis
Nilsu	Şirin	Doktor
Yadigar	Çiçek	Jeolog
Cansu	Deniz	Öğretmen
Gözde	Umut	Mühendis
Özge	Derya	Hemşire



```
<musteri_bilgileri>
        <kisiler>
            <ad>Seda</ad>
            <soyad>Çiçek</soyad>
            <meslek>Avukat</meslek>
        </kisiler>
        <kisiler>
            <ad>Eda</ad>
            <soyad>Güneş</soyad>
            <meslek>Muhasebeci</meslek>
        </kisiler>
        <kisiler>
            <ad>Simge</ad>
            <soyad>Mavi</soyad>
            <meslek>Mühendis</meslek>
        </kisiler>
        ……
</musteri_bilgileri>
```

Veri Kaynağı (Data Source)

Veritabanı programlarında ise veriler satır ve sütunlardan oluşan bir düzen içinde tutarlar. Ancak, veritabanlarının kendilerine has biçimsel bir yapıları vardır. Veritabanı programları, büyük miktardaki veriler üzerinde her türlü ilişkilendirmeyi, sorgulamayı, düzenlemeyi, yönetmeyi kolaylaştıran özelliklerinden dolayı tercih edilmektedir.

Veri Erişim Teknolojileri

Uygulamalarda veriye erişmek için birçok veri erişim teknolojisi geliştirilmiştir. Bu teknolojilerden bazıları aşağıdaki gibi sıralanabilir;

- **ODBC (Open Database Connectivity)**

Birçok kuruluşun katılımıyla geliştirilen *ODBC* teknolojisi ile birçok veri kaynağına bağlanılabilir. *ODBC* uygulama ortamlarında API (Application Programming Interface – Uygulama Programlama Arayüzü) sunmaktadır. Hem yerel (Local) hem de uzaktaki (Remote) veri kayaklarına erişmeye olanak sağlar.

- **DAO (Data Access Object)**

ODBC'nin kullanımının zor olması ve yeni dillerle kullanılamaması üzerine bu teknolojinin geliştirilmesiyle oluşturulmuş bir teknolojidir.

Veri Erişim Teknolojileri

- **RDO (Remote Data Object)**

Uzak veri kaynaklarına erişimde *ODBC*'nin performansının geliştirilmesiyle oluşturulan daha yeni bir teknolojidir.

- **OLE DB (Object Linking and Embedding DataBase)**

COM arayüzünü kullanarak birçok sisteme bağlantı sağlayan bir veri erişim teknolojisidir. Bu özelliği ile en çok kullanılan teknolojilerden biridir.

- **ADO (ActiveX Data Object)**

Yüksek seviyeli programlama dillerinde tercih edilen OLE DB teknolojisi kullanan ve veriye erişim kolaylaştıran bir teknolojidir.

Veri Erişim Teknolojileri

- **ADO.NET**

.NET uygulamalarında her türlü veriye erişim için veri tiplerine sahip, COM desteği gerektirmeyen, XML standardı üzerine kurulmuş ve .NET platformu özelliklerini kullanabilen, ADO teknolojisinin gelişmiş versiyonudur.

ADO.NET Mimarisi

ADO.NET, .NET platformunda kullanılan ortak bir katmandır. .NET ile geliştirilen tüm uygulamalar, veriye erişimde ADO.NET tiplerinden faydalanmaktadır.

ADO.NET Mimarisi

ADO.NET ile farklı veri tabanları ve veri tabanı yönetim sistemleri kullanılabilmektedir. Bu nedenle .NET platformu geliştirilirken farklı standartları destekleyen tipler yazılmış ve ayrı isim alanları (Namespace) oluşturulmuştur. Bu isim alanlarından bazıları şunlardır;

- SQL için, *Sql Server Veri Sağlayıcısı (Sql Server .NET Data Provider)*
- Oracle için, *Oracle Veri Sağlayıcısı (Oracle Data Provider)*
- OLEDB için, *OleDb .NET Veri Sağlayıcısı (OleDb .NET Data Provider)*
- ODBC için, *ODBC :NET Veri Sağlayıcısı (ODBC .NET Data Provider)*

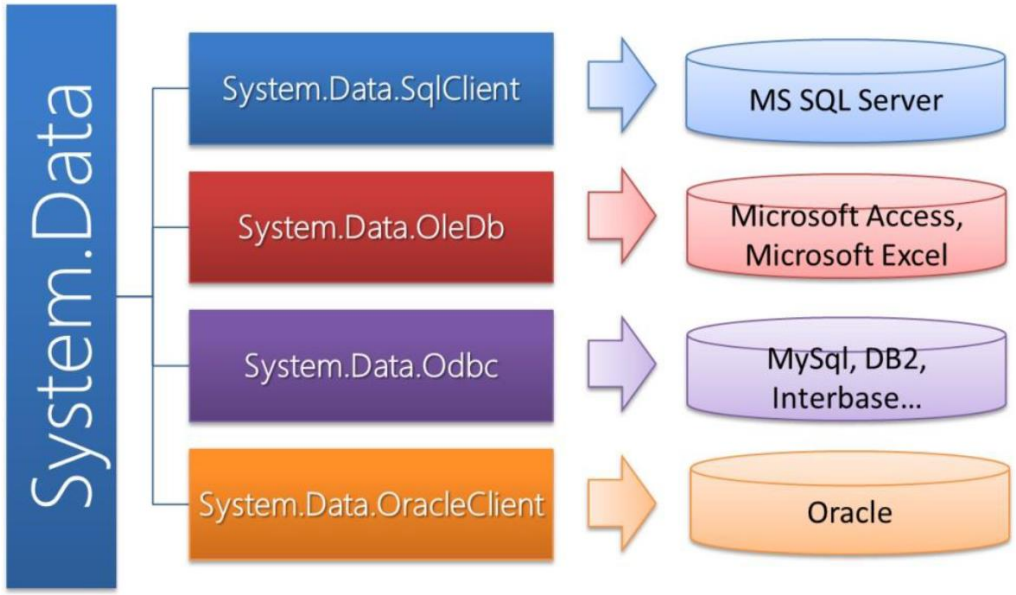
Tüm bu isim alanları .NET platformunda *System.Data* isim alanı altında yer almaktadır.

ADO.NET Mimarisi

Veri tabanına bağlanmada ADO.NET size "System.Data" isim alanı içinde bir takım hizmet sınıfları sunmaktadır. "System.Data" isim alanı aynı zamanda oldukça geniş bir sağlayıcı kümesini de içinde barındırır. Bu sayede farklı platformlara ait (Oracle, MySQL, Access vb.) veri kaynaklarına bağlanabilme imkânı sağlar. Böylece farklı veri kaynaklarını tek bir uygulamada toplayabilir ve hatta birbirleri ile konuşturabilirsiniz.

Projenizde hangi veri tabanı ile bağlantıya geçmek istiyorsanız ilgili sağlayıcı isim alanını üzerinde bulunan sınıf hizmetlerini kullanmalısınız.

ADO.NET Mimarisi



Veri tabanı ile bağlantı kurulacağı zaman *System.Data* isim alanı ile kullanılan veri erişim teknolojisi *(System.Data.KullanılanTeknolojiİsimAlanı)* referans olarak uygulamaya eklenmelidir.

ADO.NET Mimarisi

- **System.Data.SqlClient**

MS SQL Server ile çalışabilmek için yazılmış tiplerin yer aldığı bu isim alanı, SQL Server 7.0 ve sonraki versiyonları ile kullanılabilmektedir.

- **System.Data.OleDB**

MS SQL Server 6.5 ve önceki sürümleri, Microsoft Access, Oracle, Microsoft Excel dosyaları gibi OleDB arayüzüne sahip tüm sistemler ile bağlantı kurmak için gerekli hizmet sınıflarını barındırmaktadır.

ADO.NET Mimarisi

- **System.Odbc**

ODBC (Open Database Connectivity) standartlarını destekleyen ve ODBC sürücüsü bulunan sistemlere bağlantı kurmayı sağlayan hizmet sınıflarını barındırmaktadır.

- **System.Data.Oracle**

.Net 2.0'dan sonra Oracle veri yönetim sistemine daha güçlü destek sağlanmıştır. Ancak Oracle veri sağlayıcısını kullanabilmek için öncelikle "System.Data.OracleClient" referansının proje referansları arasına dahil edilmesi gerekmektedir.

Veri Kaynağından Veri Gösterimi Hayat Döngüsü

Veri kaynağından veri gösterme 3 aşamadan oluşur. Bu 3 aşama genellikle tüm programlama dillerinde bulunur:

1. Veri kaynağına bağlantı yapılır.
2. Veri kaynağına sorgu cümleleri gönderilir.
3. Veri kaynağı üzerindeki sorgulama sonucunda elde edilen kayıtların veya sorgulama sonucunda etkilenen kayıtların uygulamada kullanıcıya gösterilmesi vb. işlemlerin yapılmasıdır.

Veri Kaynağına Bağlantı

Veri kaynağını bir banka ve uygulamayı bankanın müşterisi olarak düşünürsek, bağlantı yapılması müşterinin bankanın adresini bulup bankaya gitmesi, görevliye kendini tanıtıcı belgeleri göstermesi (hesap cüzdanı, nufus cüzdanı gibi) ve görevlinin gelen kişinin müşteri olarak kabul etmesine kadar olan aşamalar olarak düşünebiliriz.

Bu aşamada olabilecek hatalar, kişinin gittiği adreste bankanın bulunmaması (yanlış bağlantı yolu verilmesi) veya kişinin banka müşterisi olmamasıdır. (veri kaynağının kullanıcı adı ve şifresinin yanlış yazılması).

Sorgu Cümleleri

Banka ve müşteri örneğinden devam edersek sorgu göndermek banka görevlisine yapmak istediğimiz işlemleri söylemektir.

Bankada yapılan bazı işlemlerde - havale yapmak veya para yatırmak gibi- müşteriye sadece işlemin yapıldığını belirten bir belge döndürülür. Bazı işlemlerde ise - para çekmek gibi- müşteriye belgenin yanında para gibi bazı nesneler verilir.

Veri kaynakları ile yapılan işlemlerde de durum oldukça benzerdir. Bazı işlemlerde - veri yazma, silme, güncelleme- uygulamaya sadece işlemin başarıyla yapılıp yapılmadığı bilgisi döndürülür. Bunun yanında bazı işlemler -veri okuma gibi- uygulamaya istenen verileri de döndürür.

Sorgu Cümleleri

Bu aşamada karşılaşılabilecek hatalar, müşterinin hatalı bir işlem yapmak istemesi durumu (hesabında olan paradan daha fazla para çekmek istemesi gibi) veya müşterinin yapmak istediği işlemi yanlış belirtmesidir (havale yapacağı hesap numarasını yanlış belirtmesi gibi). Aynı şekilde veri kaynağından da benzer sebeplerden dolayı hatalar gelebilir.

Sorgulama Sonucunun Değerlendirilmesi

…ve son aşama veri kaynağından alınan bilgilerin teker teker okunup uygulamada kullanıcıya gösterilmesidir.

Banka örneğinden devam edersek alınan paranın harcanması olarak düşünülebilir. Veri kaynağından veri gelmediği durumlarda ise genellikle kullanıcıya işleminiz başarıyla tamamlanmıştır gibi bir mesaj verilir.

ADO.NET Bağlantı Yöntemi ile Veriye Erişim

ADO.NET Microsoft tarafından geliştirilen ve bir veri kaynağı ile iletişimi sağlayan kütüphanedir. Az önce bahsedildiği gibi veri kaynağı bağlantısı üç aşamadan oluşmaktadır. ADO.NET içerisinde bulunan veri sağlayıcılarında bulunan sınıf ve metodlar, bu üç aşamayı yerine getirmek için kullanılır.

ADO.NET sınıflarının başında "ADO" gibi bir belirteç yazılmadığından ilk defa veri tabanı programı yapan geliştiriciler ADO.NET kütüphanesini hiç kullanmadığını sadece uygulamaların kendi kütüphanelerini kullandığını düşünebilirler. Fakat bu çok büyük ihtimalle yanlış bir düşüncedir ve uygulamada kullanılan veri tabanı bağlantısı yapan veya sorgu göndermeye yarayan sınıflar ADO.NET'e ait sınıflardır.

ADO.NET Bağlantı Yöntemi ile Veriye Erişim

ADO.NET 'in getirmeye çalıştığı; tüm veri kaynaklarına standart bir yöntemle bağlanmaktır. Yani veri kaynağı oracle, mssql gibi veri tabanları olsun veya düz metin dosyası olsun, ADO.NET benzer yollarla farklı veri kaynaklarında benzer işlemleri yapmaya olanak sağlar. Bu işlem birazdan anlatılacak olan data provider'lar ile sağlanmaktadır.

Kısa Bir Tarihçe

Aslında veri kaynaklarına ulaşımı standartlaştırma çalışmalarının 20 yıla yakın bir geçmişi vardır. Önceden her bir veri kaynağı için birbirinden alakasız sınıf ve metodları kullanmak gerekliydi. Standartlaşma adına ilk adım ODBC (Open Database Connectivity) ile getirildi. ODBC, veri kaynaklarına ulaşmak için geliştirilen ortak bir dildir ve ilk olmasının avantajını kullanarak şu ana kadar bir çok uygulamada kullanıldı ve bir çok veritabanı odbc uyumlu kütüphanelerini yazdı.

ODBC'nin dezavantajı oldukça alt seviye bir standart olmasıydı. Daha kolay bir standart oluşturmak için Microsoft OLE DB standartını getirdi. OLE DB kullanması daha kolay bir standarttır. Günümüzde bir çok veritabanı firması OLE DB standartında yazılmış kütüphanelerini piyasaya sürmüştür.

Yeni standartlar çıktıkça bu standartları kullanan ortak standart yeni kütüphaneler geliştirme gereği duyuldu. Bunun için Microsoft ilk önce UDA (Universal Data Access) kütüphanesini geliştirdi. Bunun ardından ADO kütüphanesini geliştirdi. Son olarak Microsoft ADO'yu oldukça geliştirerek ADO.NET kütüphanesini .NET framework ile çıkardı.

ADO.NET ile veri tabanına ister ODBC ile başlanılsın , ister OLE DB ile başlanılsın veya istenirse Oracle veya MSSQL'de olduğu gibi kendi veritabanı bağlantı kütüphaneleri kullanılsın, aynı standart metodlar ve sınıflar ile veri kaynağına veri işlenir veya kaynaktan okunur.

Data Providers (Veri Sağlayıcıları)

Önceden bahsedildiği gibi hangi veri tabanı olursa olsun, veri kaynağı ile yapılan işlemler 3 aşamadan oluşuyor. Bunlar bağlantı açma, sorgu gönderme ve sorgu sonucunu işlemedir. ADO.NET'te bulunan her bir data provider'da bu üç aşama için hazırlanan standart sınıf ve metodlar vardır.

Örnek olarak ODBC veri sağlayıcısında, veri tabanı bağlantısı yapan System.Data.Odbc.OdbcConnection sınıfı bulunur. Aynı zamanda OLE DB data provider'ında, veri tabanı bağlantısı yapan System.Data.OleDb.OleDbConnection sınıfı vardır. ADO.NET 'te tüm işlemler standart metod isimleriyle yapıldığından, her 2 sınıfın bağlantı açan Open() metodu ve bağlantı kapatan Close() metodları vardır.

Hazır Data Providerlar (Veri Sağlayıcıları)

Microsoft .NET framework kütüphanesinde hazır olarak gelen bazı veri sağlayıcılar vardır. Her bir data provider bir adet name space (ad alanı) ve onun içerisindeki sınıflar ile belirlenir. ADO.NET'te hazır olarak gelen data provider'lardan bazıları aşağıdaki gibidir.

Veri Sağlayıcı Adı	Açıklama	Desteklenen Bazı Veritabanları
System.Data.Odbc	ODBC desteği olan veritabanlarına bu veri sağlayıcı ile bağlanabilinir.	•DB2 •My Sql •Paradox •Access •Excel
System.Data.OleDb	OLEBD desteği olan veritabanlarına bu veri sağlayıcı ile bağlanabilinir.	•Interbase •Access •MS Sql server •Oracle
System.Data.OracleClient	Sadece Oracle veritabanına başlanmak için Microsoft tarafından çıkarılan veri sağlayıcı.	•Oracle
System.Data.SqlClient	Sql server'a (SQL Server CE hariç tüm versiyonlar) başlanmak için geliştirilen veri sağlayıcı.	•MS SQL Server
System.Data.SqlServerCe	SQL Server CE'ye (mobil araçlar için geliştirilen Sql Server sürümü) başlanmak için geliştirilen veri sağlayıcı.	•SQL Server CE

Hazır Data Providerlar (Veri Sağlayıcıları)

Veri sağlayıcılarını sadece bir veri kaynağı için yapılan özel veri sağlayıcıları ve bir çok veri kaynağına hitap eden genel veri sağlayıcıları olarak 2'ye ayrılır. Özel veri sağlayıcıları sadece belli bir veritabanı için optimize olarak geliştirildiğinden genellere göre oldukça hızlıdır. Bu yüzden öncelikle -eğer varsa- kullanacağımız veri kaynağının özel veri sağlayıcısını araştırmamız gerekir, eğer yoksa kullanacağımız veri kaynağının OLE DB veya ODBC desteklediğini araştırıp bunlara ait genel veri kaynaklarını (System.Data.OleDb, System.Data.Odbc) kullanmamız gerekir.

İPUCU:

ADO.NET içerisinde bulunan 5 adet veri sağlayıcının yanında isteyen kendi veri sağlayıcısınıda yazabilir. Bazı firmalar (Oracle gibi) ADO.NET için kendi veri sağlayıcılarını yazmıştır. Bunun için yapılması gereken bazı ADO.NET interface sınıflarının (IConnection, ICommand gibi) türetip, bu interface sınıflarında bulunan metodları istediğimiz veri tabanına özel olarak geliştirmektir. Tüm veri sağlayıcıların sınıfları IConnection, ICommand gibi standart interface sınıflarından türetildiği için, bu sınıflardaki metod isimleri hep aynıdır.

DİKKAT:

Her ne kadar Microsoft tarafından Oracle için yazılmış veri sağlayıcısı var ise de Oracle geliştiricilere kendi veri sağlayıcısını (Oracle Data Access Component) kullanmasını tavsiye ediyor. Oracle'ın kendi veri sağlayıcısı ADO.NET içerisinde hazır olarak bulunmuyor. Oracle'ın sitesinden indirilip projelere gerekli dll dosyalarının referans olarak eklenmesi gerekmektedir.

Veri Sağlayıcılarında Bulunan Ortak Sınıflar

Daha önce bahsedildiği gibi veri kaynağı ile yapılan işlemler 3 adımdan oluşmaktadır. 1. aşama olarak veri kaynağıyla bağlantı yapılır, 2. aşamada veri kaynağına sorgu gönderilir ve son aşamada veri kaynağından gelen bilgiler okunur. Tüm veri sağlayıcılarında bu üç aşama için sınıflar vardır. Bu sınıflardan;

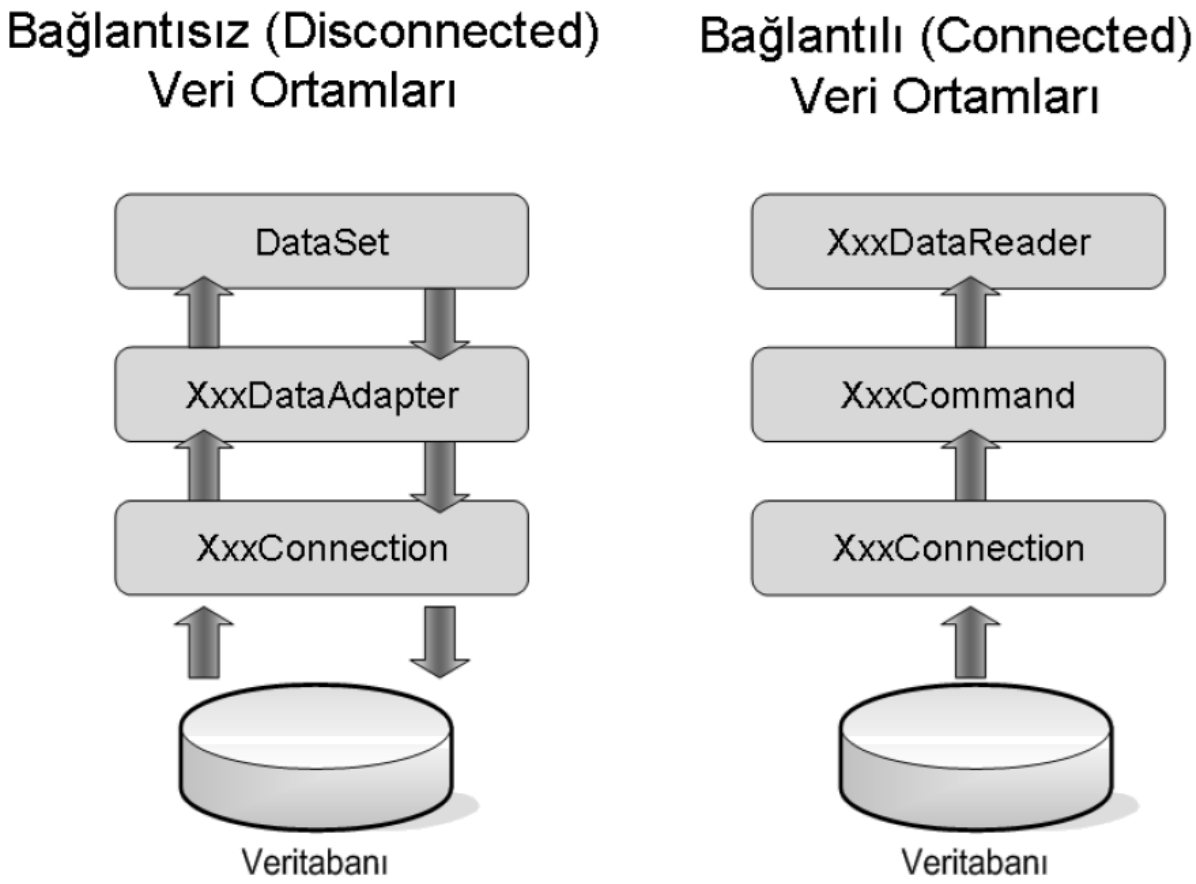
Connection sınıfı 1. aşama için kullanılır. Bu sınıf ile veri kaynağına bağlantı yapılır.
Command sınıfı 2. aşama için kullanılır. Bu sınıf ile veri kaynağına sorgu cümleleri gönderilir. Sorgu cümleleri parametre alabilirler, bu parametreler Parameter sınıflarıyla eklenir.
DataReader sınıfı 3. aşama için kullanılır. Sorgudan dönen sonuç satır satır bu sınıf sayesinde okunur.
DataAdapter sınıfı da 3. aşama için kullanılır. Bu sınıfın DataReader'dan farkı bu sınıftan dönen veriler satır satır okunmaz, onun yerine veriler DataSet adında bir sınıfa konulur. DataSet sınıfından istenilen zamanda veriler okunabilinir. DataSet sınıfı veri sağlayıcılarının bir sınıfı değildir. Verileri sunucu hafızasında tutmak için kullanılır. DataSet nesnesi diğer makalelerde detaylı şekilde anlatılacaktır.
Exception sınıfları veri sağlayıcısı sınıflarında yapılan işlemlerde bir hata olduğunda veri sağlayıcısı tarafından fırlatılır.

Veri Sağlayıcılarında Bulunan Ortak Sınıflar

Bu ortak sınıflar tüm veri sağlayıcı ad alanlarında bulunur. Genellikle isimlerinin başında veri sağlayıcısının adı bulunur.

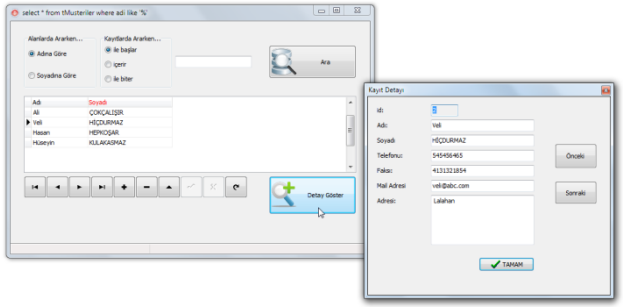
Veri Sağlayıcı	Connection Sınıfı	Command Sınıfı	DataReader Sınıfı	DataAdapter Sınıfı
System. Data. Odbc	OdbcConnection	OdbcCommand	OdbcDataReader	OdbcData Adapter
System. Data. OleDb	OleDbConnection	OleDbCommand	OleDbDataReader	OleDbData Adapter
System. Data. OracleClient	OracleConnection	OracleCommand	OracleDataReader	OracleData Adapter
System. Data. SqlClient	SqlConnection	SqlCommand	SqlDataReader	SqlData Adapter
System. Data. SqlServerCe	SqlCeConnection	SqlCeCommand	SqlCeDataReader	SqlCeData Adapter

ADO.NET'de Veri Ortamları



ADO.NET'de Bağlantı Cümleciği

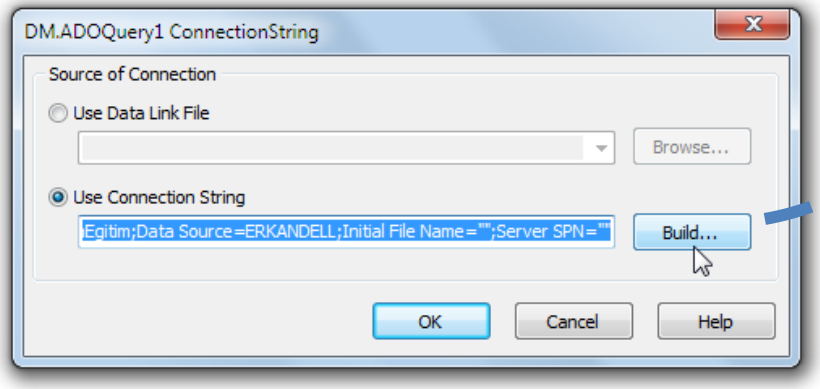
Connection String

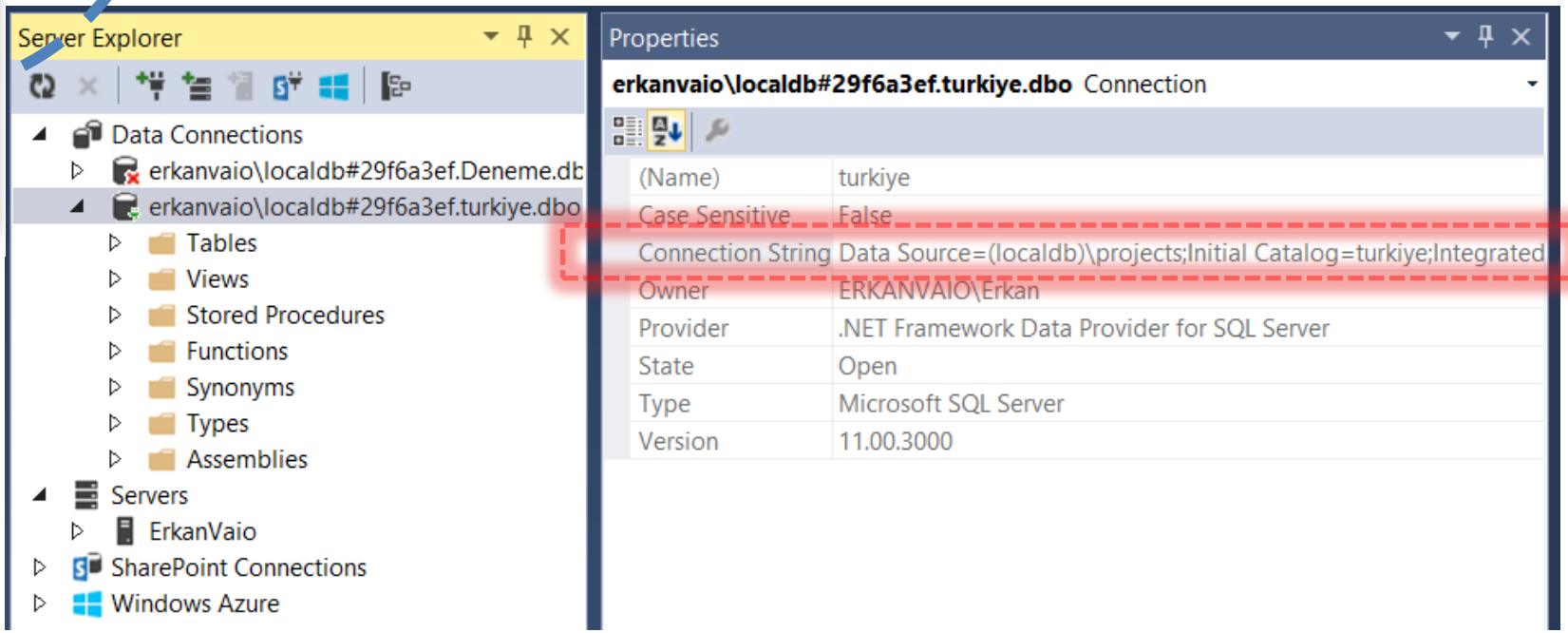


```
Provider=SQLNCLI10.1;Integrated Security=SSPI;Persist
Security Info=False;User ID="";Initial
Catalog=UzaktanEgitim;Data Source=ERKANDELL;Initial File
Name="";Server SPN=""
```

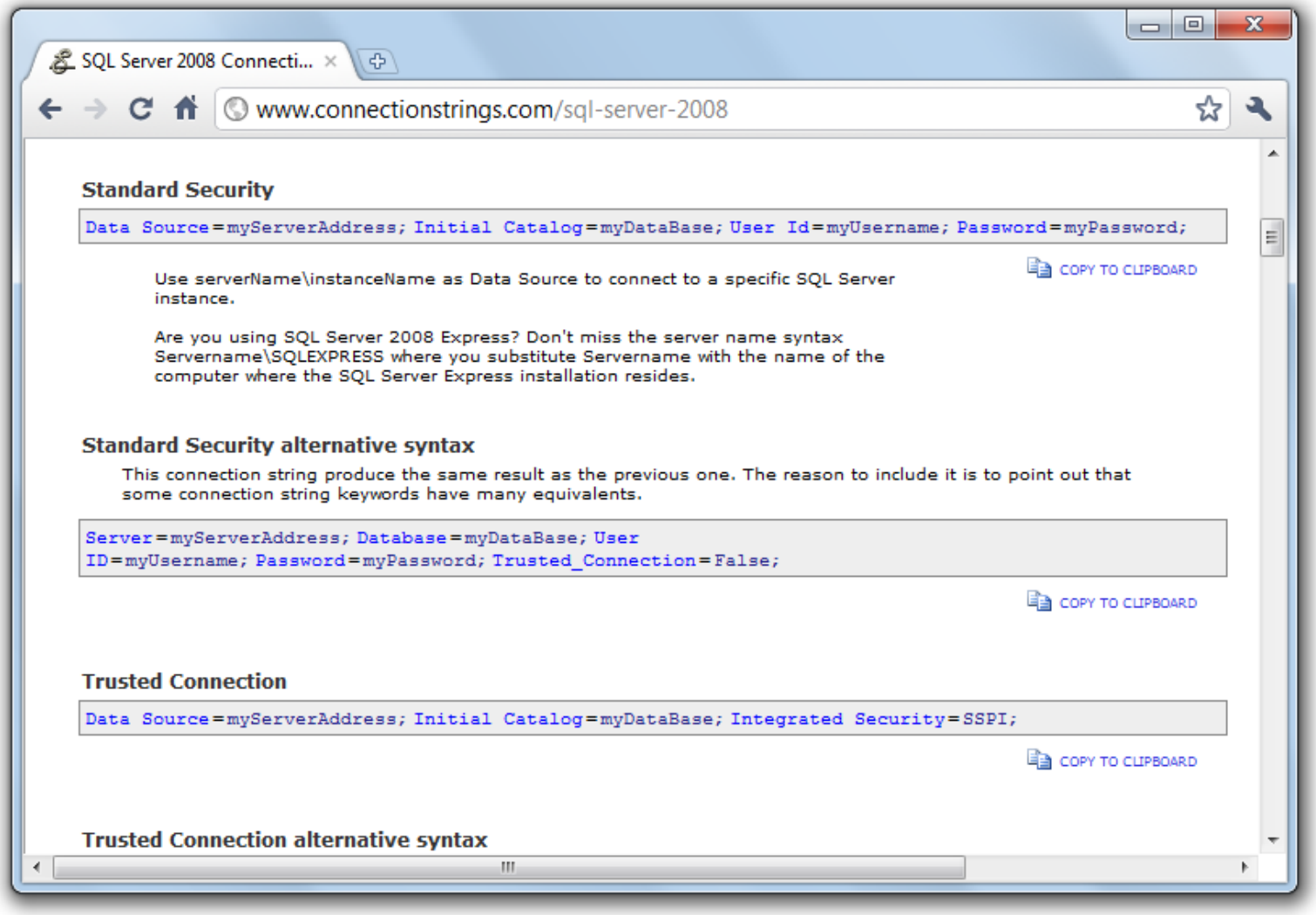
Connection String

```
Provider=SQLNCLI10.1;Integrated Security=SSPI;Persist Security
Info=False;User ID="";Initial Catalog=UzaktanEgitim;Data
Source=ERKANDELL;Initial File Name="";Server SPN=""
```





Connection String



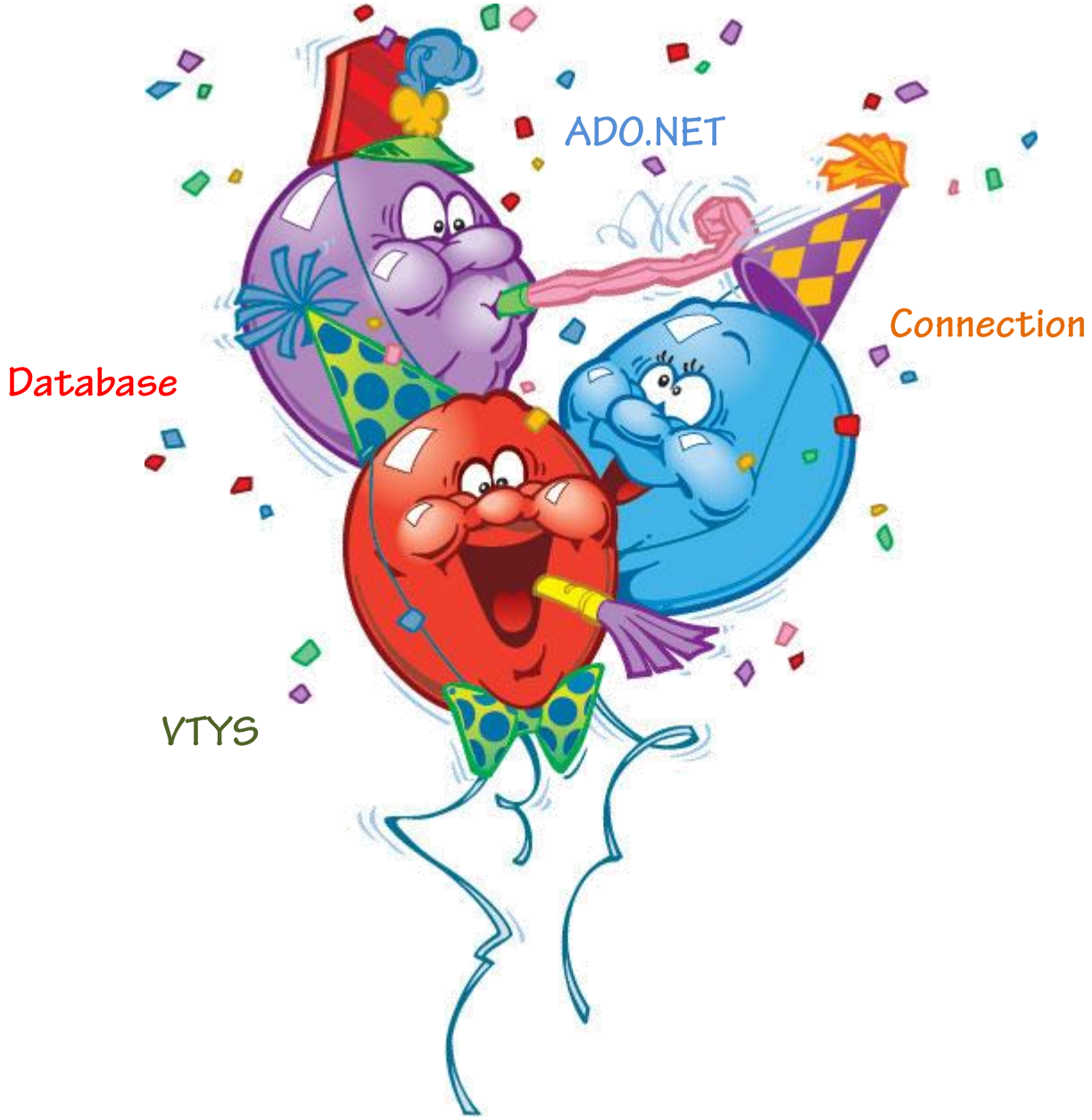
Data Source Kontrolleri

Data Source kontrolleri ADO.NET 2.0 ile beraber gelmiştir. Data Source kontrollerinin özelliği, veri kaynağından bilgi göstermek için gerekli 3 aşamayı kendi içerisinde yapmasıdır. Yani Data Source kontrolü kendi içerisinde veri tabanına bağlantı yapar, veri tabanına sorgu gönderir ve sorgudan gelen sonucu Data Set'e veya DataReader sınıfına aktarır.

Data Source kontrollerinin amacı geliştiricileri "sıfır-kod" 'a yakınlaştırmaktır. Sıfır-kod Microsoft'un uygulamaları en az kod ile geliştiricilere yazdırmaya çalışmasıdır. Data Source kontrolleriyle daha önceden 3-5 satır kod ile yapılan işlemler, 1 satırda yapılabilmektedir.

Data Source kontrolleri , veri sağlayıcılarla karıştırmamak gerekir. Data Source kontrolleri aslında kendi içerisinde veri sağlayıcıları kullanır. Veri sağlayıcıların içerisindeki Connection, command ve diğer sınıfları kullanır fakat bu işlemler uygulama geliştirici tarafından görülmez. Ayrıca data source kontrollerine öğrenip veri sağlayıcıları öğrenmemek ileride sıkıntı oluşturabilir.

Havada Kalmasın ☺



Teşekkürler…

bize katlandığınız için!