

# 5

## Amaçlarımız

- Bu üniteyi tamamladıktan sonra;
- 🕒 SQL ve DDL arasındaki ilişkiyi açıklayabilecek,
  - 🕒 DDL ile gerçekleştirilecek işlemleri sıralayabilecek,
  - 🕒 Veri tanımlama diliinde tablo ve indeks ile ilgili komutları açıklayabilecek,
  - 🕒 Görünüm ve saklı yordam ile ilgili DDL komutlarını tanımlayabilecek bilgi ve becerilere sahip olacaksınız.

## Anahtar Kavramlar

- Yapılandırılmış Sorulama Dili
- Veritabanı Yönetim Sistemi
- Veri Tanımlama Dili
- Veritabanı Nesneleri

## İçindekiler

Veritabanı Sistemleri

Veri Tanımlama

- GİRİŞ
- VERİ TANIMLAMA DİLİ
- ÖRNEK BİR VERİTABANI YÖNETİM SİSTEMİ KURULUMU
- VERİ TANIMLAMA DİLİ TABLO İŞLEMLERİ
- VERİ TANIMLAMA DİLİ INDEKS İŞLEMLERİ
- VERİ TANIMLAMA DİLİ DİĞER İŞLEMLER

# Veri Tanımlama

## GİRİŞ

Veritabanı yönetim sistemlerinde verilerin depolanması, veri gösterimi, güvenlik yönetimi, veri bütünlüğü yönetimi vb. fonksiyonlar; tasarımı yapılan şemalar ve oluşturulan nesneler üzerinden yapılmaktadır. Bunların oluşturulması veri tanımlama olarak adlandırılmasından dolayı, veritabanı yönetim sistemi üzerinde gerçekleşmesi ise **Veri Tanımlama Dili** (DDL-Data Definition Language) ile mümkündür.

DDL ayrı bir dil olmayıp, veritabanı nesnelerinin oluşturulması ve düzenlenmesi işlevlerini yapan Yapılandırılmış Sorgu Dilinin (SQL-Structured Query Language) alt komut grubudur. SQL komutlarının kullanım amaçlarına göre olmuşmuş diğer bazı gruplar ise Veri İşleme Dili (DML) ve Veri Kontrol Dilidir (DCL). Burada, DML, veri girmek, değiştirmek, silmek ve verileri almak için kullanılan SQL komut grubu; DCL ise veritabanı kullanıcısı veya rolü ile ilgili izinlerin düzenlenmesini sağlayan komut grubudur. Örnek olarak DDL'de, tabloların oluşturulması, silinmesi ve bazı temel özelliklerinin düzenlenmesini sağlamak üzere sırası ile **CREATE**, **CREATE**, **DROP** ve **ALTER** komutları kullanılır. DML örnek olarak, veri seçmek için **SELECT**, veri eklemek için **INSERT**, veri silmek için **DELETE**, veri güncellemek için **UPDATE** verilebilir. DCL'de ise, kullanıcıya yetki tanımlama için **GRANT**, kullanıcı yetkilerini engellemek için **DENY** ve daha önce yapılmış olan yetki ve izinleri kaldırmak için **REVOKE** komutları kullanılır.

SQL'ın gelişim süreci ile beraber DDL komut setlerinde de gelişmeler olmuştur. Bilindiği gibi SQL'ın ilk sürümü 1970'li yıllarda SEQUEL olarak ortaya çıkmış olup 1980'li yıllarda ANSI standartlarında tanımlamalar yapılmıştır. Daha sonra 1992 yılına kadar, bazı ufak güncellemeler ile değişik ara sürümler çıkmıştır. 1992 yılında ise SQL için ISO/ANSI standartları tanımlanmıştır. Bu tarihteki standartlar içinde, veritabanı şemalarının yönetimi mümkün kılan DDL özellikleri de eklenmiştir. 1999 yılında bazı nesne tabanlı özellikler ve gömülü SQL özellikleri eklenmiştir. 2003 ve sonraki yıllarda ise XML(Extensible Markup Language) ile ilişkili özellikler SQL standartlarına eklenmiştir. XML için benzeri veri tanımlama dili özellikleri XSD (XML Schema Definition) ile gerçekleşmektedir. Bu yillardan sonra güncellemeler devam etmekle beraber yakın zamanda olan 2008 ve 2011 güncellemelerinde eski sürümlerdeki özellikler korunarak yeni özellikler eklenmiştir.

SQL standartlarında, belli veri kümesi girdi olarak alınıp başka bir veri kümesi üretilir. Ara işlemler için yordamsal dil özelliği zayıftır veya yoktur. Yordamsal dil özelliklerini güçlendirmek üzere SQL'in farklı Veritabanı Yönetim Sistemine (VTYS) özelleşmiş sürümleri de bulunmaktadır. Örnek olarak, T-SQL Microsoft SQL Server ve Sybase, PL/SQL ise Oracle için geliştirilmiştir. Bu özelleşmiş sürümler SQL standartları yanı sıra veritabanı yönetim sistemi içinde, akış kontrolü, döngü vb. yordamsal dil özellikleri yanı sıra yukarıda belirtilen veri tanımlama ile ilgili de önemli ek özellikler sunmaktadır.

**Veri Tanımlama Dili** ile veritabanında tablolar, indeksler, görüntümler vb. oluşturulabilir, silinebilir veya bazı temel özellikleri düzenlenebilir.

## VERİ TANIMLAMA DİLİ

DDL komutlarının en temel üç ifadesi **CREATE**, **ALTER**, **DROP** komutlarıdır. Bu komutlar veritabanı nesneleri üzerinde sırasıyla oluşturma, düzenleme ve silme işlemlerini yerine getirir.

Veri Tanımlama Dili, ilişkisel veritabanı sistemlerinde varlık-iliski veri modeline karşılık gelen ilişkisel şemalarının (relational schemas) seçilen VTYS üzerinde oluşturulması aşamasında kullanılır. Bu aşamada, veritabanından beklenen performans, veri bütünlüğü, veri gösterimi vb.'ne bağlı olarak VTYS üzerinde farklı nesneler oluşturularak tasarım beklenileri karşılanır.

Bu amaçla DDL, veri tabanı tablo ve görünüm (view) oluşturulması, silinmesi ve değiştirilmesi, veritabanı tabloları üzerinde tanımlamalar ile bütünlük kısıtları (integrity constraint) oluşturulması vb. amaçlı kullanılır. Günümüzdeki VTYS sistemlerindeki güncel komutlara bakıldığından tablo, görünüm vb. oluşturma yanı sıra indeks oluşturma, tetikleyici (trigger) oluşturma, servis oluşturma vb. birçok işlem de DDL altında yapılmaktadır. Veri tanımlamada kullanılan en temel komut olan **CREATE** için MS SQL Server'da bazı güncel kullanım alanları Tablo 5.1'de verilmektedir.

**Tablo 5.1**  
MS SQL Server  
Tarafından Desteklenen  
Bazı Ddl Komutları ve  
Açıklamaları

KOMUT ADI	AÇIKLAMASI
<b>CREATE SCHEMA</b>	Bu komut ile mevcut veritabanında bir şema içinde tablolar ve görünümler oluşturulabilir. Bu şema üzerinde farklı kullanıcılara ait yetkiler tanımlanabilir.
<b>CREATE CERTIFICATE</b>	SQL sunucudaki bir veritabanına sertifika ekler.
<b>CREATE SEQUENCE</b>	Dizi üretimi için bir nesne ve özelliklerini oluşturur. Tanımlanan değerlere bağlı olarak farklı değerlerde nümerik diziler üretir.
<b>CREATE ASYMMETRIC KEY</b>	Veritabanında asimetrik anahtar oluşturur.
<b>CREATE INDEX</b>	Tablo veya görünüm üzerinde ilişkisel indeks oluşturur. Özellikle sorgu performansının artırılması için farklı yapıarda oluşturulabilir.
<b>CREATE TABLE</b>	Veritabanında yeni bir tablo oluşturur.
<b>CREATE TRIGGER</b>	DML, DDL vb. için tetikleyici oluşturur. Veritabanı sunucusunda belli bir olay gerçekleştiğinde ilgili saklı yordam çalıştırılabilir. DML için tetikleyici olaylar INSERT, UPDATE veya DELETE iken, DDL için tetikleyici olaylar CREATE, ALTER ve DROP'tur.
<b>CREATE VIEW</b>	İçeriği sorgular ile belirlenen sanal bir tablo oluşturmaktadır.

İNTERNET



Tablo 5.1'de bazıları verilen DDL komutlarının hepsine <https://msdn.microsoft.com/en-us/library/ff848799.aspx> adresinden ulaşabilirsiniz.

## Şema Oluşturma

SQL, 1992 yılından önceki sürümlerinde veritabanındaki tüm tabloların aynı şemaya ait olmasını gerektirmekteydi. Veri tanımlama dili açısından, 1992'de oluşturulan standart ile aynı VTYS'de olan bazı tablolar ve ilgili nesneler gruplanarak bir şema oluşturma imkânı olmaktadır. SQL şemasını nitelendirmek üzere, şema ismi, şema sahibi ve şemadaki tüm elemanların tanımlanması yapılabilmektedir. Şema içinde, tablolar, görünümler, alanlar ve yetkilendirmeler şemayı tanımlamaktadır. Şema Tablo 5.1'de verildiği gibi **CREATE SCHEMA** komutu ile oluşturulmaktadır. **CREATE DATABASE** de birçok VTYS için aynı komut işlevini görmektedir.

VTYS'de tüm kullanıcıların şema ve şema elemanı oluşturma yetkisi olmayabilir. Veritabanı yöneticisi tarafından ilgili kullanıcılar bu yetkilerin tanımlanması gereklidir. Şema kavramına ek olarak, SQL'de ayrıca katalog kavramı da bulunmaktadır. Katalog ise belli şemaların bir araya gelmesi ile oluşturulmaktadır. Bir katalog her zaman bilgi sağlayan **INFORMATION\_SCHEMA** şemasına sahiptir. Bütünlük kısıtları, eğer aynı katalog içindeki ilişkilerde tanımlı ise kullanılabilir.

## Tablo ve Kısıtların Oluşturulması

Bu bölümde, ilişkisel örnek(relational instance) için tablo, tuple için satır veya kayıt, öznitelik(attribute) için sütün veya alan terimleri değişimli olarak kullanılacaktır. Veri tanımlama dilinde **CREATE TABLE** komutu ile VTYS'de yeni bir tablo, öznitelikleri ve kısıtların tanımlanması yapılabilir. Tablo oluşturmada alanlar ve veri tipleri ilk olarak belirlenir. Bunun yanı sıra alan kısıtları, birincil anahtar kısıtları ve bütünlük kısıtları da tanımlanabilir. Veri tanımlama dilinde birden fazla tablo üzerinde tanımlanabilecek kısıt komutları da bulunmaktadır. Bütün bunlar dikkate alındığında, işletme ile ilgili birçok veri tipi ve ilişkisinin hata yapmayacağı şekilde veritabanına aktarımı için **CREATE TABLE** komutu ve beraberinde kullanılacak seçenekler, yönetim bilişim sistemlerinde veritabanı gerçeklemesi için önemli bir yere sahiptir. Takip eden bölümde bu konuda farklı örnekler uygulamalı olarak verilecektir.

## İndeks Oluşturma

Veritabanı'nın performansının artırılmasında indeks kullanımı önemli bir yere sahiptir. VTYS'nin fiziksel şemasını oluşturmaya yönelik indeksler kullanılır. Eğer işletmelerde belli tip sorgular daha yaygın kullanılmakta ve sorguların cevaplarında gecikmeler olmakta ise yapılacak indeks tanımlamaları ile hızlandırma mümkün olabilir. SQL sorgusunun özelliklerine ve işlem yapılacak tablonun boyutuna bağlı değişik özelliklerde indeks tanımlaması yapılmaktadır.

Yukarıdaki alt başlıklar incelendiğinde VTYS de en üstteki kavramsal şemadan, verilerin saklandığı alt katmandaki fiziksel şemaya kadar çok farklı katmanlardaki nesne tanımlamaları veri tanımlama dili ile mümkün olmaktadır. Takip eden bölümlerde önceki örnek bir veritabanı sistemi kurulumu verilecek olup, sonrasında ise veri tanımlama dili ile ilgili yaygın kullanılan bazı komutlar uygulamalı olarak verilecektir.

## Görünüm Oluşturma

İşletmelerin veritabanı sistemlerinde çok farklı kullanıcılar için veriler bulunmaktadır. Bazı veriler belirli kullanıcıların ortak erişimine açık iken bazı verilerin ise tüm kullanıcılar açık olmaması gereklidir. Bu ise her kullanıcının veritabanını farklı bir erişim veya görünüm ile görmesini gerektirmektedir. VTYS'nin sağladığı görünüm nesnesi ile farklı kullanıcıların erişimi için CREATE VIEW ile sanal veri kümeleri oluşturulmaktadır.

Veritabanı üzerinde Görünüm oluşturmanın faydalarını araştırınız.

SIRA SİZDE



## ÖRNEK BİR VERİTABANI YÖNETİM SİSTEMİ KURULUMU

Yönetim bilişim sistemlerinde farklı VTYS'ler kullanılmaktadır. Microsoft SQL Server bunlardan yaygın olarak kullanılmıştır. Bu kitap kapsamında okuyucunun veri tanımlama dili uygulamalarını tekrar edebilmesi için ücretsiz MS SQL Server 2014 Express ve SQL Management Studio bileşenlerini kurması önerilir. Örnek uygulamalar, SQL Management Studio içinde kopyalanarak tekrar edilecek yapıda hazırlanmıştır.

## Kurulumu Hazırlık

SQL Server 2014 Express yazılımı kurmadan önce bazı hazırlık ve kontrollerin yapılması gereklidir. Bu kapsamda, kurulacak bilgisayardaki donanımın minimum sistem gereksinimleri, işletim sisteminin 32bit ya da 64 bit olup olmadığı bilgilerini kontrol ederek <http://www.microsoft.com/en-us/download/details.aspx?id=42299> adresinden “MS SQL Server 2014 Express” ve “SQLManagement Studio” ürünlerinin kurulum dosyalarının indirilmesi gereklidir. Bu internet bağlantısının değişmesi ya da yeni sürümlerin çıkması durumunda Microsoft web sayfasından MS SQL Express yükleme sayfasına ulaşılması gerekecektir.

DİKKAT



**SQL Server Express olan veritabanı motorunun kurulumu için bilgisayarda .NET 3.5 SP1 veya .NET 4 ‘ün kurulması gereklidir. Tam veya ileri versiyonların kurulumu için .Net Framework 4.0 veya 4.5 ile birlikte ayrıca .Net Framework 3.5 SP1 de yüklü olmalıdır (Yeni versiyonlarda buradaki bazı gereklilikler farklılık gösterebilecektir).**

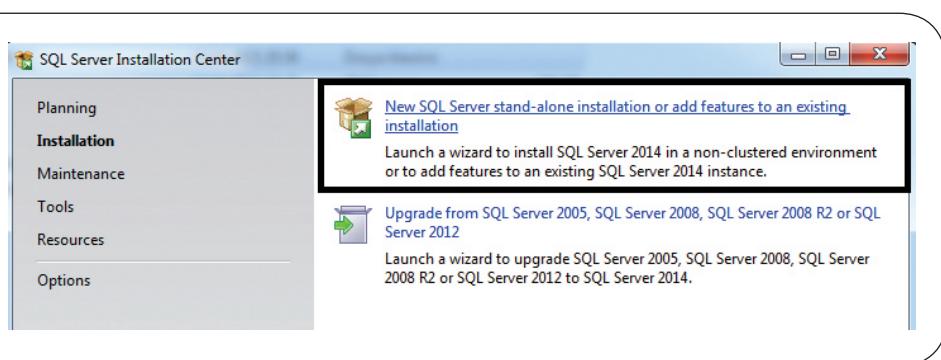
## MS SQL Server 2014 Express Kurulumu

Bu örnekte, daha önce SQL sunucu kurulmamış 64 bit bilgisayar üzerine kurulum yapılacaktır. Bunun için ilgili dosyayı açıp klasördeki (örn: SQLEXPR\_x64\_ENU) kurulum (setup.exe) dosyasının çalıştırılması gerekmektedir. Kurulum yardımcısı ile aşağıdaki adımlar izlenerek kurulum tamamlanacaktır. Bu kitabın hazırlandığı ana kadar SQL Server yazılımının Türkçe sürümü bulunmadığı için kurulum ve yazılım ara yüzleri İngilizce olarak aşağıda adımlar olarak verilmektedir:

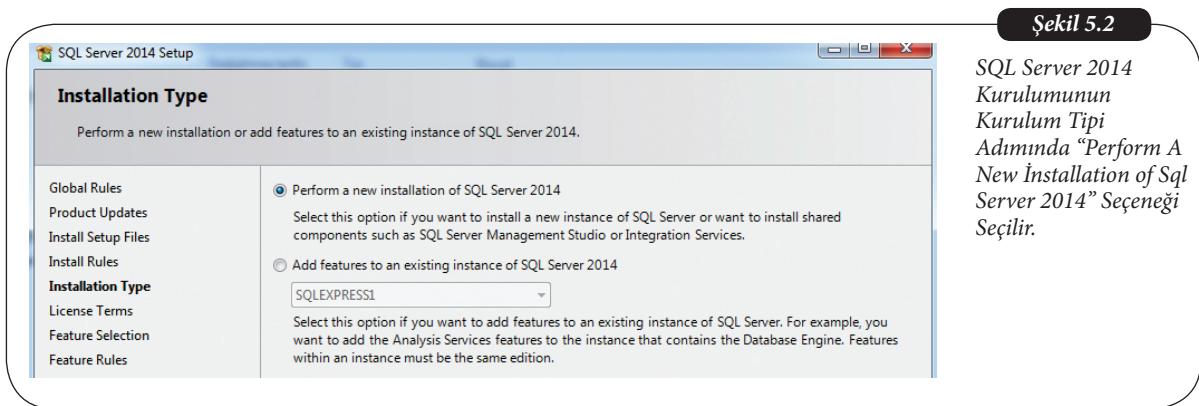
- Kurulum (Installation):** Bu adımda, Şekil 5.1'deki “New SQL Server stand-alone installation or add features to an existing installation” seçeneğinin seçilmesi gereklidir.

**Şekil 5.1**

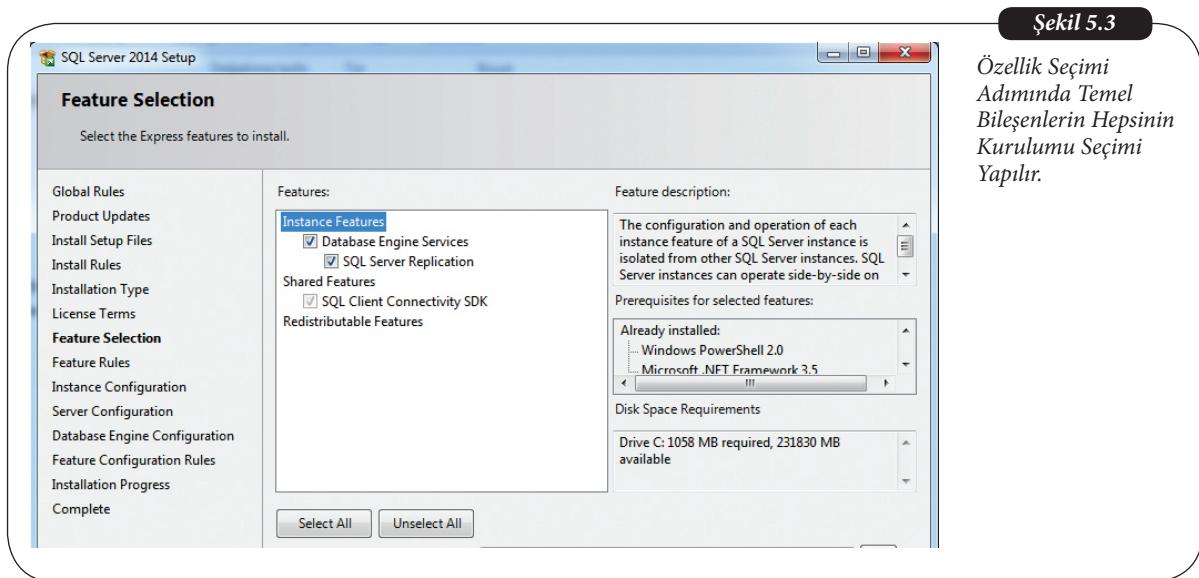
MS SQL Server 2014 Kurulumunda “New SQL Server Stand-Alone Installation or Add Features to An Existing Installation” Seçeneğinin Seçilmesi Gerekir.



- Evrensel Kurallar (Global Rules):** Bu adımda, kurulum yardımcısı tarafından kullanılacak dosyaların kurulumu sırasında bir problemle karşılaşılmaması için bazı kontroller yapılır. Problem yoksa otomatik olarak sonraki adıma geçilir.
- Kurulum Tipi (Installation Type):** Bu adımda Şekil 5.2. deki “Perform a new installation of SQL Server 2014” seçilip devam edilir.



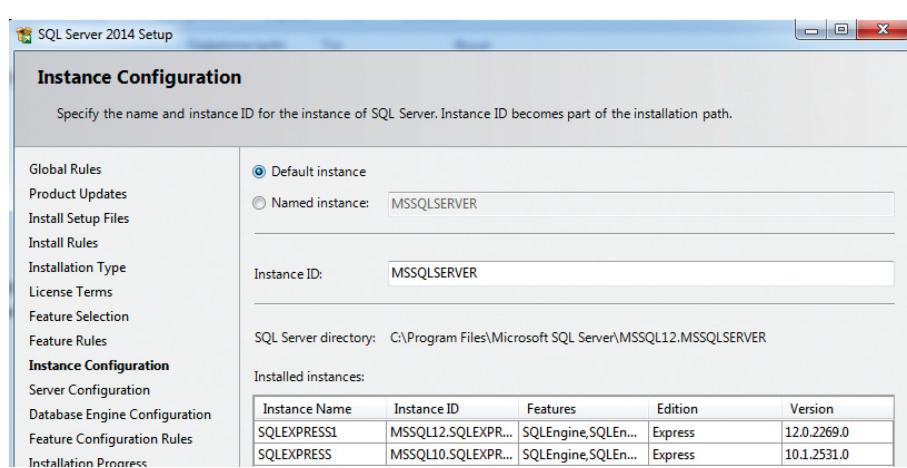
- iv) **Lisanslama (License Terms):** Bu adımda “I accept the license terms” ile lisans anlaşmasının kabul edildiğini işaretlenmelidir.
- v) **Özellik Seçimi (Feature Selection):** Express kurulum adımında, Şekil 5.3'teki temel bileşenlerin hepsinin kurulumu önerilir. Bunlar, MS SQL Server üzerinde hangi alt özelliklerin çalışacağını belirler.



- vi) **Oluşum Yapılandırması (Instance Configuration):** Bu adımda SQL Server 2014 için bir isim verebilir veya Şekil 5.4'deki “Default instance” seçimiyle ilerlenebilir. Daha sonra yeni bir kurulum yapılacaksa ona isim verilmesi gereklidir.

**Şekil 5.4**

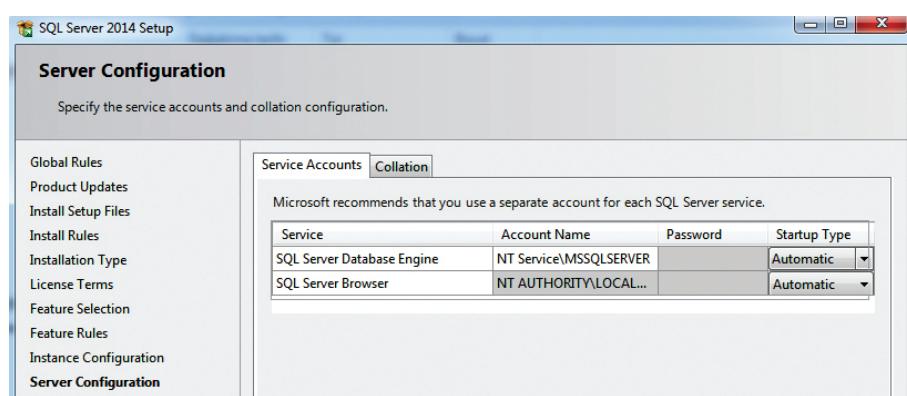
*SQL Server 2014 kurulumunun Oluşum Yapılandırması Adımında “Default Instance” Seçimi Yapılır.*



- vii) **Sunucu Yapılandırması (Server Configuration):** SQL Server veritabanı motoru için daha önce seçilen özellikler de dikkate alınarak Şekil 5.5'teki gibi başlangıç parola belirlemesi yapılabilir.

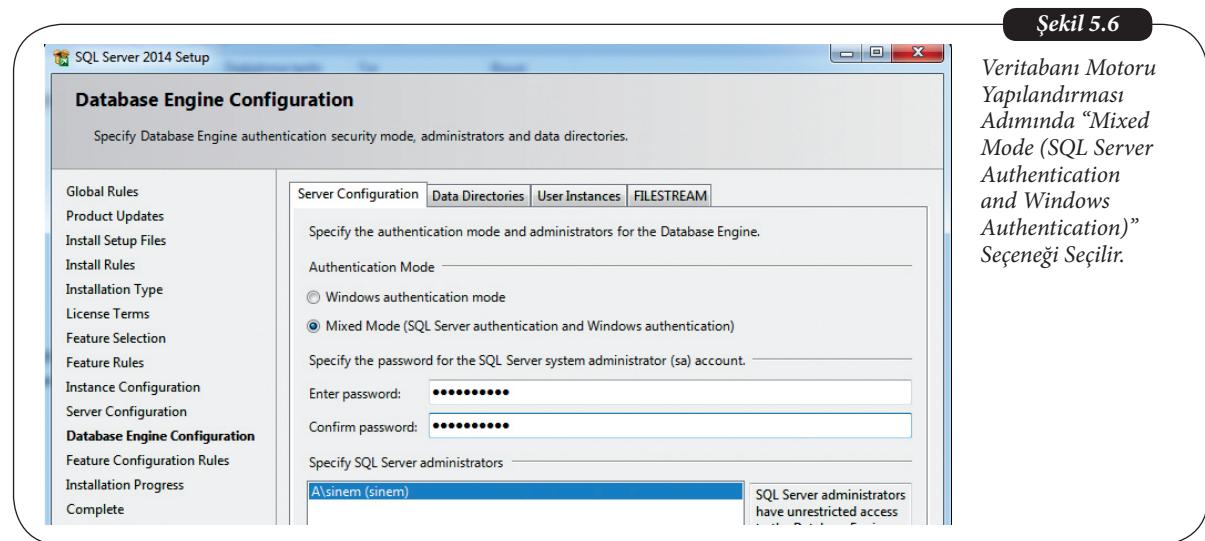
**Şekil 5.5**

*Sunucu Yapılandırması adımında SQL sunucu veritabanı motoru için parola tanımlaması yapılabilir.*



Bir bilgisayarda birden fazla veritabanı oluşturulabilir. Kullanıcıların bir sistemdeki veritabanına ulaşmaları için ilgili bilgisayarın aşağıdaki adresini ve Veritabanı Oluşum Adını (Instance Name) bilmesi gerekmektedir.

- viii) **Veritabanı Motoru Yapılandırması (Database Engine Configuration):** MS SQL Server 2014 veritabanı motoru servisine yönetici olarak erişirken kullanılacak kimlik doğrulama yöntemi belirlenir. Şekil 5.6'da gösterilen “Mixed Mode (SQL Server authentication and Windows authentication)” seçeneği ile devam edilebilir. Bu şekilde “Data Directions” sekmesinde istenilen değişikliklerin yapılabileceği veritabanı, veri yedek gibi içeriklerin bulunacağı varsayılan dizinler listelenir.

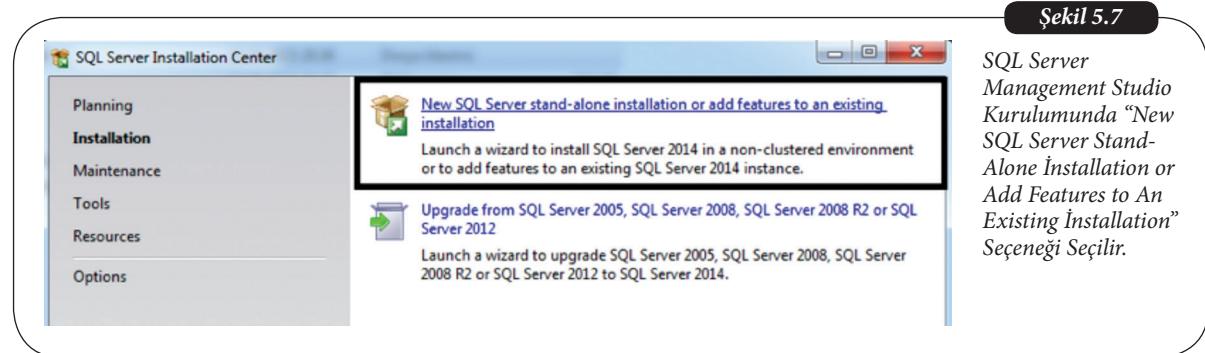


- ix) **Kurulum İşlemi ve Tamamlanma (Installation Progress and Complete):** Bu adımda kurulum işlemi sürdürülür ve tamamlanır. Kurulum yapılan özellikler de bu adımda listelenir.

## SQL Server Management Studio Kurulumu

Bilgisayara indirilen dosyayı açıp klasördeki (örn: SQLManagementStudio\_x64\_ENU) kurulum (setup.exe) dosyasını çalıştırınız. Kurulum yardımcısı sizi aşağıdaki adımları izleterek kurulumu tamamlayacaktır.

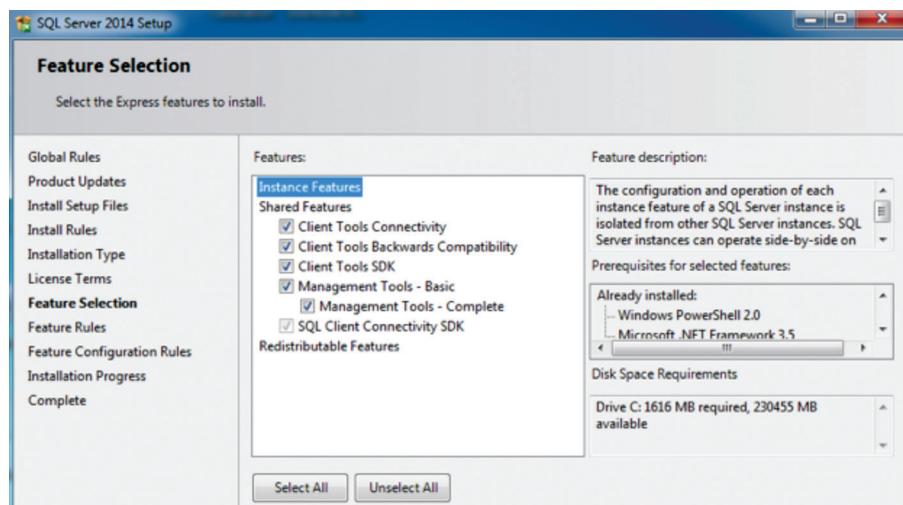
- i) **Kurulum (Installation):** Bu adımda Şekil 5.7'deki “New SQL Server stand-alone installation or add features to an existing installation” seçeneği seçilir.



- ii) **Özellik Seçimi (Feature Selection):** SQL Server Management Studio için Şekil 5.8'deki ekranda seçili temel özelliklerinin hepsinin kurulumu önerilir.

**Şekil 5.8**

*SQL Server Management Studio Kurulumu Özelliğin Seçimi Adımında Temel Bileşenlerin Hepsinin Kurulumu Seçimi Yapılır.*



iii) **Kurulum İşlemi ve Tamamlanma (Installation Progress and Complete):** Bu adımda başarı ile kurulum yapılan özellikler listelenir.

Bu kurulumdan sonra bilgisayarınızdaki “SQL Server Management Studio”yu çalıştırarak SQL Server 2014’e, sağlanan ara yüzler ile erişebilirsiniz.

## Veritabanı İşlemleri

Bu kitap kapsamında verilen uygulamalarda kullanılmak üzere tasarlanmış bir veritabanının oluşturulması bu kısımda verilmektedir. MS SQL VTYS'de veritabanı oluşturmak için SQL kodları veya veritabanı oluşturma yardımcısı kullanılabilir. Bu kitapta Northwind veritabanı (Türkçesi) için kurulum örnekleri verilecektir.

### MS SQL Komutları ile Veritabanı Oluşturma

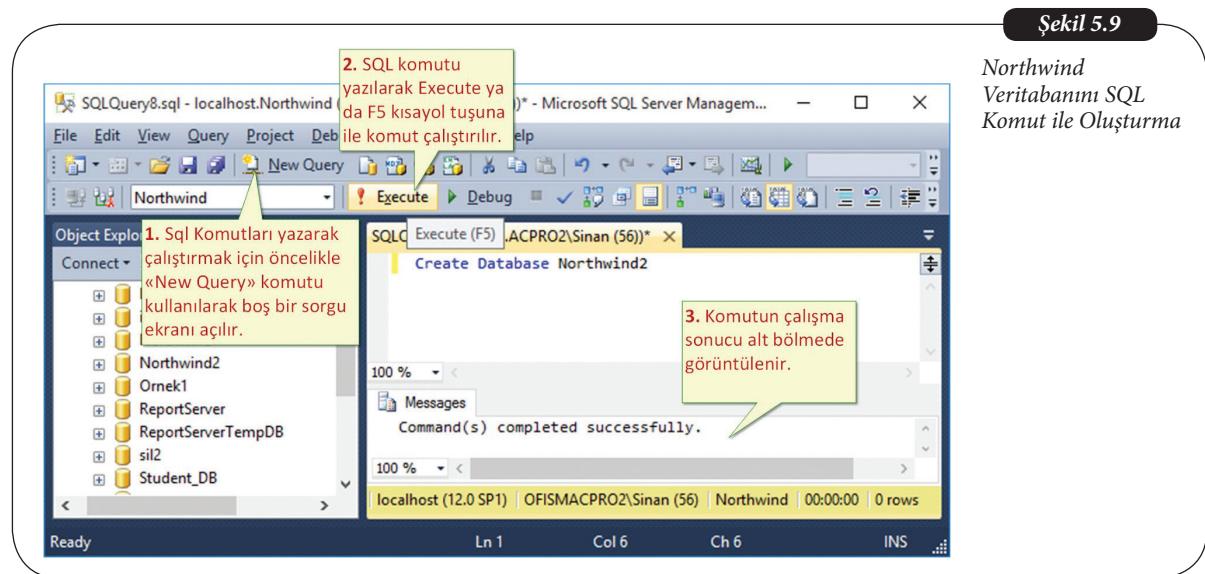
“Northwind” veritabanını komut ile oluşturmak için; “SQL Server Management Studio” programı açılarak ana menüde “New Query” e tıklayarak **SQL Sorğu penceresi** oluşturulur. Şekil 5.9'deki SQL Sorğu penceresine

#### **CREATE DATABASE Northwind2**

komut satırı yazılıp ana menüde “Execute” e tıklanır veya “F5” tuşuna basılır. Veritabanının adını yeni bir isim olan NWYeniİsim'e değiştirmek için

**ALTER DATABASE Northwind2 MODIFY NAME=NWYeniİsim;**  
satırı kullanılabilir.

**SQL Sorğu penceresi:** Ana menüde “New Query” ile açılan SQL Sorğu penceresi sorguların çalıştırılması için kullanılmaktadır. Soru bir veritabanı üzerinde işlem yapacaksız ana menü veritabanı açılır listesinden ilgili veritabanının da seçili olması gerekmektedir. Bu pencerenin altında “Command(s) completed successfully” çıktısı yazdırıldığımız komutun düzgün çalıştığını göstermektedir. Komutun diğer çıktıları da burada görselleşir.



"Object Explorer>Databases" altında oluşturulan veritabanı gözükmüyorsa "Databases" üzerine sağ tuş ile tıklayarak "refresh" e tıklayınız. Veritabanı üzerinde yapılan işlemler gerçekleşmiş olmakla beraber kullanıcı arayüzüne anında yansımayabilir. Bunun için "refresh" ile güncellemenin ara yüze aktarımı sağlanır.



DİKKAT

### MS SQL Komutları ile Veritabanı Silme

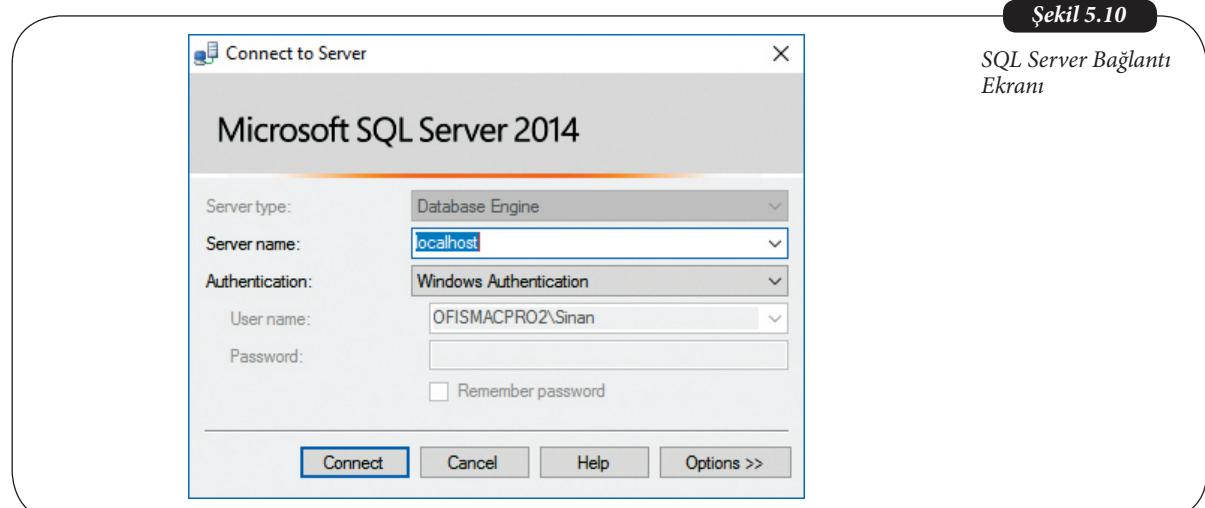
"NWYeniİsim" veritabanını silmek için; "SQL Server Management Studio" programı açılarak ana menüde "New Query" e tıklayarak SQL Sorgu penceresi oluşturulur. İlgili SQL Sorgu penceresine

**DROP DATABASE NWYeniİsim**

komut satırı yazılıp ana menüde "Execute" e tıklanır veya "F5" tuşuna basılır.

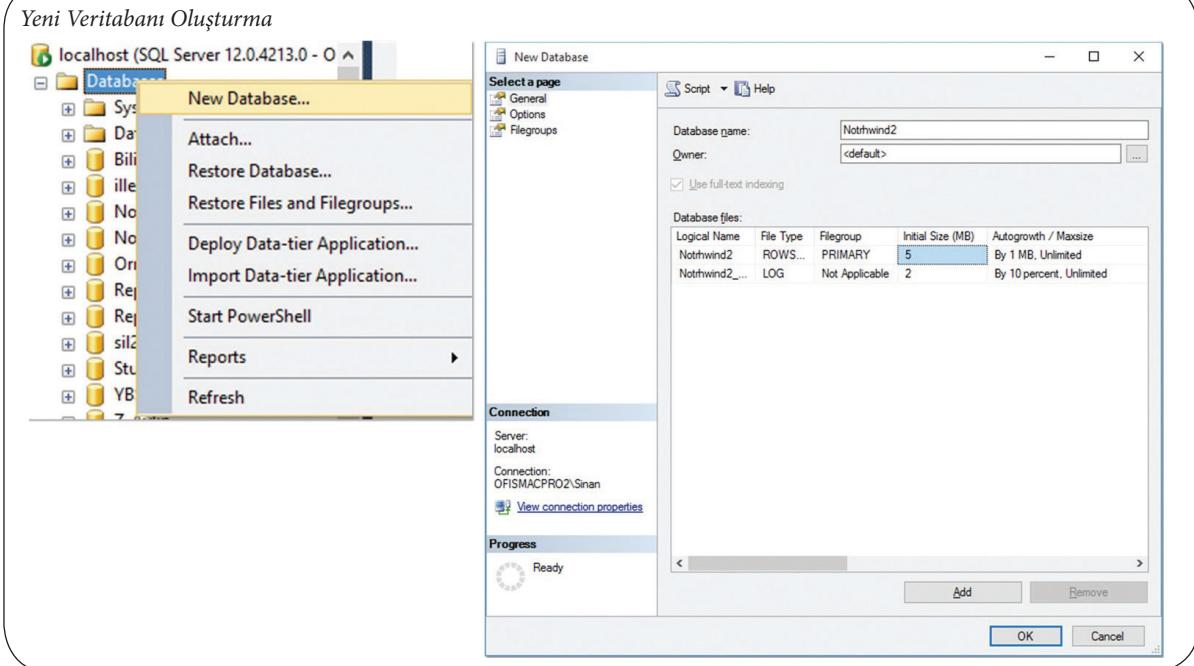
### Ms Sql Server 2014 Yardımcısı ile Örnek Veritabanını Oluşturma

"Northwind" veritabanını görsel ara yüz kullanarak oluşturmak için "SQL Server Management Studio" programı açılarak Şekil 5.10'daki gibi programı yüklerken seçtiğimiz sunucu ismi ile bağlantı işleminin gerçekleştirilir.



Sonrasında Nesne Tarayıcısı (Object Explorer) açık değilse “View > Object Explorer” ile açılıp, Şekil 5.11'deki gibi Databases alanına sağ tıklayıp “New Database” seçeneği tıklanarak yeni veritabanımızın ismi ve ilk özelliklerinin tanımlanacağı form görüntülenir.

**Şekil 5.11**



Görsel arayüz kullanarak veritabanını silmek için Nesne Tarayıcısı (Object Explorer) açık değilse “View > Object Explorer” ile açılıp, Databases alanında ilgili veritabanı üzerinde sağ tıklayıp “Delete” seçeneği tıklanarak mevcut veritabanı silinebilir.

## VERİ TANIMLAMA DİLİ TABLO İŞLEMLERİ

Tablolar, VTYS'lerinin temel yapı taşıları olan nesnelerdir. İşletme süreçlerindeki birçok iş kuralı tablo oluşturma aşamasında farklı parametreleri kullanarak VTYS'de tanımlanabilir. **CREATE TABLE** komutu ile VTYS'de yeni bir tablo ismi, öznitelikleri ve kısıtları tanımlanabilir. Burada kısıtlar, işletmelerdeki süreçler ile ilgili kuralların tanımlamada kullanılacağı önemli bir araçtır. Tablo oluşturmada öznitelikler ile ilgili isim, veri tipi ve kısıtların tanımlanması gereklidir. Tabloda öznitelikler tanımlandıktan sonra, anahtar, varlık bütünlüğü ve referans bütünlük kısıtları tanımlanabilir. Bu kısıtlar gerekirse sonradan, **ALTER TABLE** komutu ile de eklenebilir. Ayrıca, **DROP TABLE** ile de bir tablo veritabanından silinebilir.

### Tablolarda Öznitelik Veri Tipleri ve SQL Kısıt Tanımlaması

Öznitelikler veya alanlar, tablolarda belirlenmesi gereken temel bileşenlerdir. Sayısal, metin-karakter, tarih, para vb. veri tipleri öznitelikler için kullanılabilir. Ayrıca, alanlar üzerinde bazı kısıtlar da tanımlanabilir. Takip eden kısımda, 3. ünitede detay olarak verilen veri tipleri ve kısıtlarının MS SQL özeline uygulanması olarak verilecektir.

**Sayısal Veri Tipleri:** Sayısal veri tiplerinde tamsayı (integer), kayan noktalı sayılar (floating point), nümerik sayılar ve ikili (binary) sayılar vb. için tanımlamalar yapılmaktadır. Depolanacak tamsayının büyüklüğünü göre **tinyint** (1 bayt), **smallint** (2 bayt), **int** (4 bayt), **bigint** (8 bayt) veri tipleri bulunmaktadır. Kayan noktalı sayı veri tipinde **float** (4 bayt) ve **real** (4 bayt) veri türü ya da kullanıcının ihtiyacına göre basamak sayıları tanımlanabilen **decimal** ve **numeric** veri tipleri alanları tanımlamak için kullanılmaktadır. İkili sayılar veri tipinde, sabit uzunlukta **binary** ve değişken uzunlukta **varbinary** veri tipi bulunmaktadır.

**Metin-Karakter Veri Tipleri:** Bu veri tipinde hem ASCII karakter seti, hem de Unicode uluslararası karakter seti kullanımı veya veri uzunluğunun sabit veya değişken olmasına bağlı olarak veri tipi tanımlamaları değişir. ASCII karakter seti için, sabit uzunluklu veri kümesi saklayan **char**, değişken uzunluklu veri kümesi saklayan **varchar** veri tipleri vardır. Unicode karakter seti için, sabit uzunluklu veri kümesi saklayan **nchar**, değişken uzunluklu veri kümesi saklayan **nvarchar** veri tipleri vardır.

**Tarih ve Zaman Veri Tipleri:** Bu veri tipleri içinde tarih, zaman, tarih-zaman vb. veri tipleri bulunmaktadır. Tarihi Yıl-Ay-Gün olarak saklamak için **date**, Zamanı saat:dakika:saniye olarak saklamak için **time**, her ikisini beraber saklayabilen **datetime, smalldatetime** vb. veri tipleri vardır.

**Öznitelik Kısıtlarının Tanımlanması:** Öznitelik verilerinin değerlerinin ne olacağı SQL de tanımlanması gereklidir. Örneğin, bir veri girişi sırasında bazı bilgilerin zorunlu dolması gerekiyorsa bu bilgilere karşılık gelen veritabanı özniteliklerinin de uygun şekilde tanımlanması gereklidir. SQL, öznitelik değeri olarak **NULL** kabul edebilir. Veri tanımlaması gereklili ise ilgili öznitelik için **NOT NULL** kısıtının tanımlanması gereklidir. Öznitelikler için herhangi bir değer girilmediğinde olağan(default) bir değer tanımlanması isteniyorsa **DEFAULT <value>** tanımlanması gereklidir. Eğer herhangi bir **DEFAULT** değer tanımlanmamışsa, **NOT NULL** olan öznitelikler için **NULL** değeri atanır. Öznitelikler ile ilgili diğer bir kısıt tipide aralık tanımlamaya imkân veren **CHECK** komutu ile mümkün olmaktadır. Örneğin, tamsayı değere sahip olan bir özniteligin 0-18 arasında değer alması isteniyorsa; ... **CHECK** (sayi > 0 AND sayı < 18) şeklinde tanımlanabilir. Belirli bir sayıdan başlayıp, belirlediğimiz aralığa göre artan veya azalan bir şekilde sayısal değer üretilmesi isteniyorsa, **IDENTITY** komutu da kullanılabilir. Genellikle birincil anahtar ile birlikte kullanılır.

Bu kitapta kullanılan Northwind örnek veritabanının Siparişler tablosu öznitelikleri de tanımlanarak SQL komutları ile oluşturulursun.

### ÖRNEK 5.1

Siparişler tablosu için ilişkisel şema; Siparişler ([Sipariş No],[Çalışan No],[Müşteri No], [Sipariş Tarihi], [Sevk Tarihi], [Taşıyıcı No], [Sevk Adı], [Sevk Adresi],...) olup, bu MS SQL sunucudaki Northwind veritabanında aşağıdaki SQL kodu ile oluşturulabilir.

```
CREATE TABLE [Siparişler](
    [Sipariş No] [int] IDENTITY(1,1) NOT NULL,
    [Çalışan No] [int] NULL,
    [Müşteri No] [int] NULL,
    [Sipariş Tarihi] [datetime] NULL DEFAULT (getdate()),
    [Sevk Tarihi] [datetime] NULL,
    [Taşıyıcı No] [int] NULL,
    [Sevk Adı] [nvarchar](50) NULL,
    [Sevk Adresi] [nvarchar](max) NULL,
    [Sevk Şehir] [nvarchar](50) NULL,
    [Sevk Eyalet/İl] [nvarchar](50) NULL,
    [Sevk Posta Kodu] [nvarchar](50) NULL,
    [Sevk Ülke/Bölge] [nvarchar](50) NULL,
    [Nakliye Ücreti] [money] NULL DEFAULT ((0)),
    [Vergiler] [money] NULL DEFAULT ((0)),
    [Ödeme Türü] [nvarchar](50) NULL,
    [Ödeme Tarihi] [datetime] NULL,
    [Notlar] [nvarchar](max) NULL,
    [Vergi Oranı] [float] NULL DEFAULT ((0)),
    [Vergi Durumu] [tinyint] NULL,
    [Durum No] [tinyint] NULL DEFAULT ((0)))
```

DİKKAT



**Örnek veritabanını daha önce sisteminize kurmuş iseniz Yukarıdaki örnek kodu çalıştırırken hata alabilirsiniz. Bunun nedeni Veritabanında daha önce var olan bir tablonun tekrar oluşturulamamasıdır. Komutu “CREATE TABLE [Siparişler2]( .. )” şeklinde değiştirilmesi durumunda yeni bir tablo oluşturulabilir.**

## Tablolarda Anahtar ve Diğer Bütünlük Kısıtlarının Tanımlaması

Tablo oluşturma aşamasında veya sonrasında öznitelikler ile ilgili veri tipleri ve kısıtları yanı sıra, anahtar kısıtı, bütünlük kısıtları (integrity constraints) vb. tanımlanabilir. Bu kısıtlar işletme yönetimi ile ilgili önemli iş kuralları ve veri bütünlüğünün sağlanması garantilemektedir.

**Anahtar kısıtı:** Anahtar kısıtinin özel hali olan birincil anahtar kısıtı ile tablolardaki bir veya daha fazla eşsiz değere sahip olan öznitelikler **PRIMARY KEY** komutu ile tanımlanabilir. Eğer **birincil anahtar** sadece bir öznitelikten oluşuyorsa, doğrudan **PRIMARY KEY**den sonra yazılabilir. Birden fazla alanın, beraber anahtar olması durumunda **UNIQUE** komutu da kullanılabilir.

**Bütünlük (integrity) Kıısı:** Bu kısıt, tüm veritabanı tabloları arasındaki özniteliklerin birbirleri ile olan ilişkisinin bütünlüğünün sağlanması için önemlidir. Örnek uygulama 5.1'deki Siparişler tablosundaki [Sipariş No] özniteligiye başka bir tablodan **FOREIGN KEY** kullanılarak referans verildiğinde, Siparişler tablosunda değişiklik yapılarken VTYS belirlenen opsiyonlara bağlı kabul veya ret verebilir. Bu tanımlandıktan sonra, VTYS tablolara yapılan işlemler sırasında bu kısıtlara uyumluluğu kontrol eder. Burada, veritabanı tasarımcısı alternatif davranış seçenekleri olan **SET NULL**, **CASCADE** ve **SET DEFAULT**'u da tasarılayabilir. Bu davranışın tetiklenmesi için veritabanında yapılacak olan olası silme ve güncelleme içi **ON DELETE** veya **ON UPDATE** durumlarına bağlı tanımlanabilir.

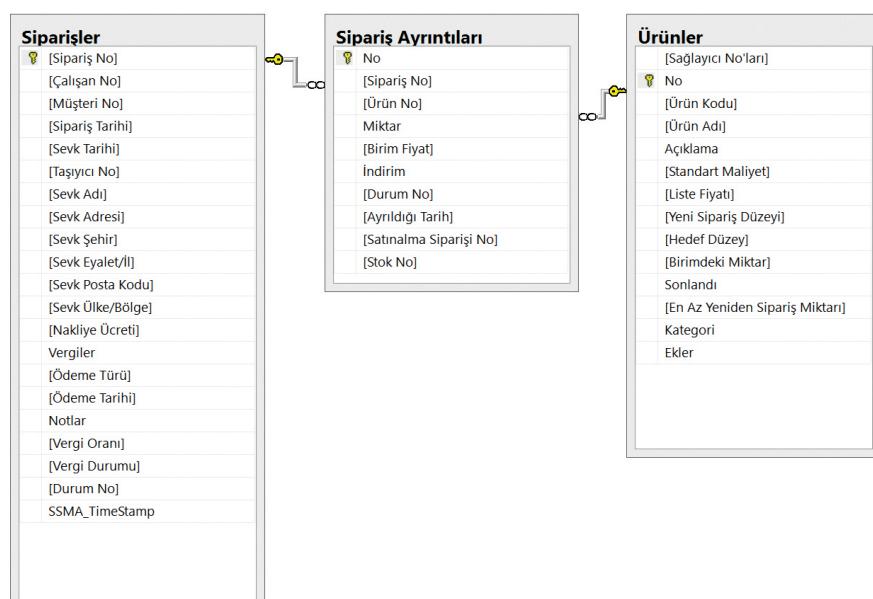
**Birincil Anahtar** (Primary Key) olarak tanımlanan alan ya da alanlar, ilgili tabloda benzersiz değer alırlar. Diğer bir deyişle aynı değerin faklı satırda yer almamasını garanti altına alırlar.

### ÖRNEK 5.2

Northwind örnek veritabanının Şekil 5.12. ile diyagramı verilen [*Ürünler*], [*Sipariş Ayrıntıları*] ve [*Siparişler*] şemaları için ilişkisel kısıtlarda dikkate alınarak tablolar SQL komutlarıyla oluşturululsun.

Şekil 5.12

Northwind Veritabanı  
Örnek Veritabanı  
Diyagramı



Şekil 5.12. ile verilen veritabanı diyagramı aşağıdaki SQL kodu ile oluşturulabilir.

```

/* Siparişler Tablosunu oluşturan SQL komutu */
CREATE TABLE [Siparişler](
    [Sipariş No] [int] IDENTITY(1,1) NOT NULL,
    [Çalışan No] [int] NULL,
    [Müşteri No] [int] NULL,
    [Sipariş Tarihi] [datetime] NULL DEFAULT (getdate()),
    [Sevk Tarihi] [datetime] NULL,
    [Taşıyıcı No] [int] NULL,
    [Sevk Adı] [nvarchar](50) NULL,
    [Sevk Adresi] [nvarchar](max) NULL,
    [Sevk Şehir] [nvarchar](50) NULL,
    [Sevk Eyalet/İl] [nvarchar](50) NULL,
    [Sevk Posta Kodu] [nvarchar](50) NULL,
    [Sevk Ülke/Bölge] [nvarchar](50) NULL,
    [Nakliye Ücreti] [money] NULL DEFAULT ((0)),
    [Vergiler] [money] NULL DEFAULT ((0)),
    [Ödeme Türü] [nvarchar](50) NULL,
    [Ödeme Tarihi] [datetime] NULL,
    [Notlar] [nvarchar](max) NULL,
    [Vergi Oranı] [float] NULL DEFAULT ((0)),
    [Vergi Durumu] [tinyint] NULL,
    [Durum No] [tinyint] NULL DEFAULT ((0)),
    CONSTRAINT [Siparişler$PrimaryKey] PRIMARY KEY CLUSTERED ([Sipariş No] ASC)
)
/* Ürünler Tablosunu oluşturan SQL komutu */
CREATE TABLE [Ürünler](
    [Sağlayıcı No'ları] [varchar](8000) NULL,
    [No] [int] IDENTITY(1,1) NOT NULL,
    [Ürün Kodu] [nvarchar](25) NULL,
    [Ürün Adı] [nvarchar](50) NULL,
    [Açıklama] [nvarchar](max) NULL,
    [Standart Maliyet] [money] NULL DEFAULT ((0)),
    [Liste Fiyatı] [money] NOT NULL DEFAULT ((0)),
    [Yeni Sipariş Düzeyi] [smallint] NULL,
    [Hedef Düzey] [int] NULL,
    [Birimdeki Miktar] [nvarchar](50) NULL,
    [Sonlandı] [bit] NOT NULL DEFAULT ((0)),
    [En Az Yeniden Sipariş Miktarı] [smallint] NULL,
    [Kategori] [nvarchar](50) NULL,
    [Ekler] [varchar](8000) NULL,
    CONSTRAINT [Ürünler$PrimaryKey] PRIMARY KEY CLUSTERED ([No] ASC),
    CONSTRAINT "CK_Ürünler_ListeFiyatı" CHECK ([Liste Fiyatı] >= 0)
)
/* [Sipariş Ayrıntıları] Tablosunu oluşturan SQL komutu */
CREATE TABLE [Sipariş Ayrıntıları](

```

```

[No] [int] IDENTITY(1,1) NOT NULL,
[Sipariş No] [int] NOT NULL,
[Ürün No] [int] NULL,
[Miktar] [float] NOT NULL DEFAULT ((0)),
[Birim Fiyat] [money] NULL DEFAULT ((0)),
[İndirim] [float] NOT NULL DEFAULT ((0)),
[Durum No] [int] NULL,
[Ayrıldığı Tarih] [datetime] NULL,
[Satınalma Siparişi No] [int] NULL,
[Stok No] [int] NULL,
CONSTRAINT [Sipariş Ayrıntıları$PrimaryKey] PRIMARY KEY
CLUSTERED ([No] ASC)
CONSTRAINT [Sipariş Ayrıntıları$New_OrderDetails] FOREIGN
KEY([Sipariş No])
REFERENCES [Siparişler] ([Sipariş No])
CONSTRAINT [Sipariş Ayrıntıları$New_ProductsOnOrders] FOREIGN
KEY([Ürün No])
REFERENCES [Ürünler] ([No])
)

```

Bu örnek incelendiğinde, üç tablo için de tek bir öznitelik kısıt olarak **PRIMARY KEY** ile birincil anahtar tanımlanmıştır. Tabloların yapısı gereği birincil anahtar olarak belirlenen alanlar **IDENTITY(1,1)** komutu ile otomatik olarak değeri belirlenen bir yapıya sahiptir. Sipariş ve Ürünler arasındaki bağlantıyı sağlayan [Sipariş Ayrıntıları] tablosu **FOREIGN KEY** iki adet kısıtı ile diğer tablolar ile ilişkilendirilmiştir. Bazı özniteliklerin olduğu alanlarda ise **NOT NULL** veya **DEFAULT(0)** kısıtları kullanılmıştır. Ayrıca, [Ürünler] tablosunda **CHECK** kısıtı ile [Liste Fiyatı] alanının alacağı değerler sınırlandırılmıştır.

DİKKAT



**MS SQL tablo, alan ve benzeri kullanıcı tanıtı isimleri verirken köşeli parantez ([]) ya da çift tırnak ("") kullanılabilir. Özellikle iki kelimededen ya da Türkçe karakterlerden oluşan ifadeler tanımlanırken adların bu semboller için e alınması derleyicinin ifadeleri doğru yorumlamasını sağlar.**

SIRA SİZDE



**Bir işletme için *Calisanlar* (*CalisanTCNo*, *CalisanAdı*, *CalisanMaas*) ilişkisel şemasını, çalışanlar için 1.300TL ile 15.000TL aralığı dışında maaş verisi girilmeyecek şekilde veritabanında tablo olarak gerçekleyecek SQL kodunu yazınız?**

### Mevcut Tablolarda Değişiklik Yapılması

Veritabanı üzerinde oluşturulan bir tablo ile ilgili, sonrasında adı, öznitelikleri veya alanlar, kısıtlar vb. tüm tablo özellikleri değiştirilebilir veya ek özellikler eklenebilir. Tablolar için, bir alanın eklenmesi veya çıkartılması, bir alanın adının değiştirilmesi, tablodan kısıt çıkartılması veya eklenmesi vb. değişiklikler **ALTER** komutu ile yapılabilir.

#### ÖRNEK 5.3

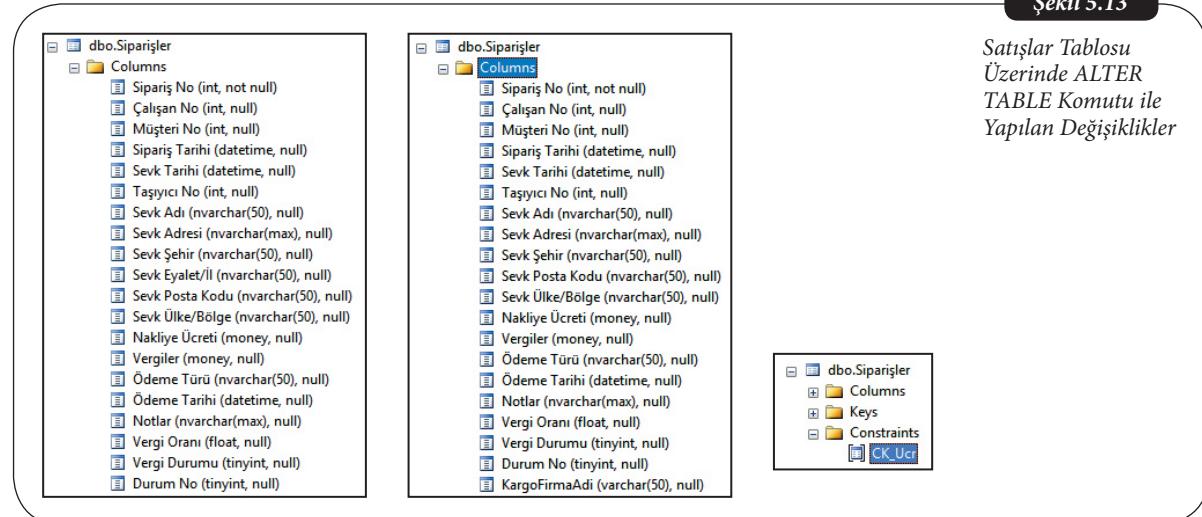
**Örnek 5.1 ile oluşturulan *Siparişler* tablosunda aşağıdaki işlemleri gerçekleştiren SQL Komutlarını oluşturunuz.**

- [*Sevk Eyalet/İl*] alanının silinmesi,
- [*KargoFirmaAdı*] alanının eklenmesi
- [*Nakliye Ücreti*] alanının değeri pozitif olacak şekilde kısıt eklenmesi.

```
USE Northwind
ALTER TABLE Siparişler DROP COLUMN [Sevk Eyalet/İl];
ALTER TABLE Siparişler ADD [KargoFirmaAdı] VARCHAR(50);
ALTER TABLE Siparişler ADD CONSTRAINT CK_Ucr CHECK ([Nakliye
Ücreti] >=0)
```

Bu işlem yapıldıktan sonra, Object Explorer>Databases>Northwind>Tables kısmındaki Satışlar tablosu aşağıdaki Şekil 5.13'deki gibi güncellenecektir. Dikkat edilirse, [Sevk Eyalet/İl] özniteligiine karşılık gelen alan silinip, [KargoFirmaAdı] alanı eklenmiştir. Ayrıca, Object Explorer>Databases>Northwind>Tables altında Constraints kısmına [CK\_Ucr] kısıtı Şekilde görüntülenmiştir.

Şekil 5.13



Tablo üzerinde **ALTER** ve/veya **DROP** işlemi yaparken diğer tablolardan bu tabloya yabancı anahtar ile bağlı bir alan varsa işlem gerçeklenmeyecektir. Öncelikle bu tip ilişkisel kısıtların kaldırılması gereklidir. Buna yönelik düzenlemeler “**ALTER/DROP/CASCADE**” komut seçenekleri ile ilerideki bölümde detaylı verilmektedir. **DROP TABLE [Tablo Adı]** komutu ile bir tablo nesnesi ve içindekiler VTYŞ'den silinir.

**Bir işletme için verilen Malzemeler(MalzemeNo, MalzemeAdı) ve MalzemeSatis(MalzemeNo, Adet, SatisTarih) ilişkisel şemaları, birbirleri ile ilgili bütünlük kısıtlarını koruyarak iki tablo olarak veritabanında oluşturacak SQL kodunu yazınız.**



## VERİ TANIMLAMA DİLİ İNDEKS İŞLEMLERİ

İndeksler veritabanı yönetim sistemlerinde mevcut verilerin dizin hâline getirilerek istenilen tablo alanlarına daha hızlı ulaşmasını sağlayan mekanizmalardır. Bu işlemler için ek depolama alanı ve yazma işlemleri gerektirirler. İndekslerin birincil kullanım amacı veritabanı işlemlerinin performansını artırmaktır uygın kullanılmadıkları takdirde ise performans düşüşüne de yol açarlar. MS SQL Server yazılımında indeks oluşturma **CREATE INDEX**, oluşturulan indeksin silinmesi için **DROP INDEX** komutu kullanılır. İndeks oluşturma komutunun yazım kuralı aşağıdaki gibidir.

```
CREATE [UNIQUE] INDEX <INDX_ADI> ON <TABLO> (<ALAN> [ASC|DESC])
```

Veritabanlarında **indeks** veriye erişim hızını artıran yapılardır.

Komutun yazım şeğlinden de fark edileceği üzere bir indeks oluşturmak için bir tablo üzerinde **indeks** oluşturulacak alanlar belirtilmeli ve oluşturulan indekse bir ad verilmesi gerekmektedir. Bu komutu bir örnek üzerinde pekiştirmek için aşağıda tanımlanan uygulamayı inceleyiniz.

#### ÖRNEK 5.4

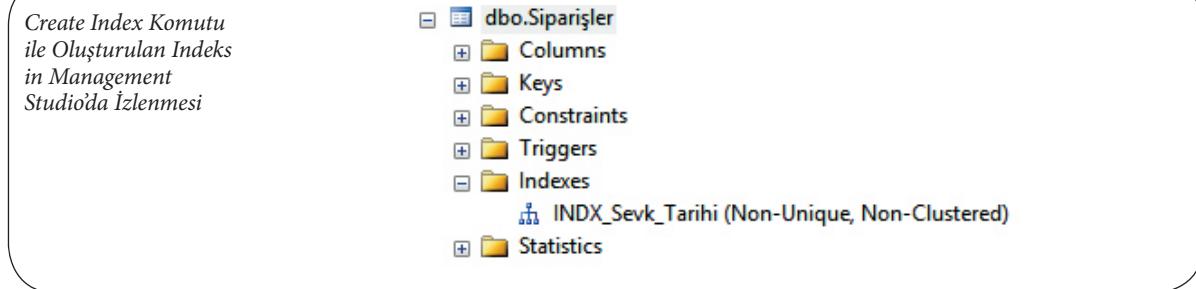
*Örnek uygulama veritabanında yer alan [Siparişler] tablosunun [Sevk Tarihi] alanına göre yoğun olarak sorgulandığı tespit edilmiştir. Siklikla gerçekleştirilen bu sorgulamaların performansını artırmak için bu alana indeks ekleyiniz.*

İstenilen indeksin oluşturulması için Create Index komutu kullanılmalıdır. Create Unique Index komutu ise ilgili alanda değerleri tekrarlanmadığı durumlarda kullanılır ve dolayısıyla aranan değerleri bulmak için daha az süre gerektirir. Siperişlere ilişkin aynı tarihte sevk olabileceği için [Sevk Tarihi] alanına benzersiz bir indeks uygulanamayacaktır. Dolayısıyla aşağıdaki komut uygulanarak indeks oluşturulur.

```
CREATE INDEX INDX_Sevk_Tarihi ON [Siparişler] ([Sevk Tarihi])
```

İndeks oluşturma komutundaki [INDX\_Sevk\_Tarihi] adı kullanıcı tarafından belirlenir. Komut sonucu VTYS'de bu isimli bir nesne olacağı için isminin sistematik bir şekilde verilmesi yerinde olacaktır. Oluşturulan indeks SQL Management Studio SQL sorgu penceresinde Şekil 5.14'deki gibi görüntülenebilir ya da `Select * from sys.indexes` komutu ile tüm sistemdeki indekslerle beraber izlenebilir.

**Sekil 5.14**



Oluşturulan bir indeksin silinmesi için **DROP** komutu kullanılır. Aşağıdaki komut örnek için oluşturulan indeksin silinmesini sağlar.

```
DROP INDEX INDX_Sevk_Tarihi ON [Siparişler]
```

SIRA SİZDE



4

İndeksleri “clustered” (kümelmiş) veya “non-clustered”(kümelmemiş) olarak oluşturmanın sorgu performansına etkisini araştırınız.

## VERİ TANIMLAMA DİLİ DİĞER İŞLEMLER

VTYS'de işlem kolaylığı sağlayacak yardımcı nesneler oluşturulabilmektedir. Görünüm (View) ve Saklı Yordam (Stored procedure) yaygın olarak kullanılan veritabanı nesnelerine ilişkin komutlar bu bölümde verilecektir.

## Görünüm Oluşturma ve Silme İşlemleri

Görünümler, kullanıcıların veritabanındaki nesneleri sorgulayarak istedikleri veri kümelerini elde etmek için kullanılan yapılardır. Bu yapılar veriyi değil veriyi elde edecek sorgu komutlarını saklarlar. Birden fazla tablo ya da veritabanı nesnesinden istenilen alanların ve kayıtların elde edilmesi için görünümler oluşturulabilir. Bu yöntem ile kullanıcıların istenilen veri alanlarına erişimi sağlanabilir. Bu şekilde güvenlik artacağı gibi, karmaşık sorgulardan elde edilen görünümlerin kullanılması uygulamada raporlama ve benzeri amaçlar için kolaylık sağlar. Görünüm oluşturmak için aşağıdaki yazım kuralı kullanılır.

```
CREATE VIEW Görünüm_Adı
AS
<SQL Seçme Sorgusu>
```

Northwind örnek veritabanında ürün kategorilere göre toplam satışlarını listeleyen görünü mü oluşturan SQL komutunu oluşturun.

### ÖRNEK 5.5

Bir görünümü oluşturmak için asıl önemli olan istenilen veri kümесini elde edecek seçme sorgusunun yazılmasıdır. Bir sonraki ünite de seçme sorgularıyla ilgili ayrıntılı bilgilere yer verecek ancak bu soruda bir seçme sorgusu komutu yazılması örneklenecektir. Bazı durumlarda ulaşım istenilen veri kümесinin karmaşık olması istenilen veri kümese birden fazla adımda ulaşılmasını gerektirebilmektedir. Bu durumda birden fazla görünüm oluşturularak nihai görünümme ulaşılır. Buradaki örnekte istenilen veri kümese ulaşabilmek için hangi tabloların kullanılacağı ve bu tabloların birbirileyle olan ilişkileri biliinmelidir. [Ürünler] ve [Sipariş Ayrıntıları] tablosu birleştirilerek istenilen veri kümese ulaşılabilir. Görünümü oluşturmak için aşağıdaki SQL komutu yazılmalıdır.

```
Create View Ornek5_5
As
SELECT Ürünler.Kategori,
SUM([Sipariş Ayrıntıları].Miktar*[Sipariş Ayrıntıları].
[Birim Fiyat]) AS Tutar
FROM Ürünler INNER JOIN
[Sipariş Ayrıntıları] ON dbo.Ürünler.No = [Sipariş Ayrıntıları].
[Ürün No]
GROUP BY dbo.Ürünler.Kategori
GO
```

Görünüm oluşturma komut dizininin çalıştırılması için GO komutu kullanılmıştır. Bunun yerine SQL sorgusu BEGIN... END bloğu arasında da yazılarak çalıştırılabilir.

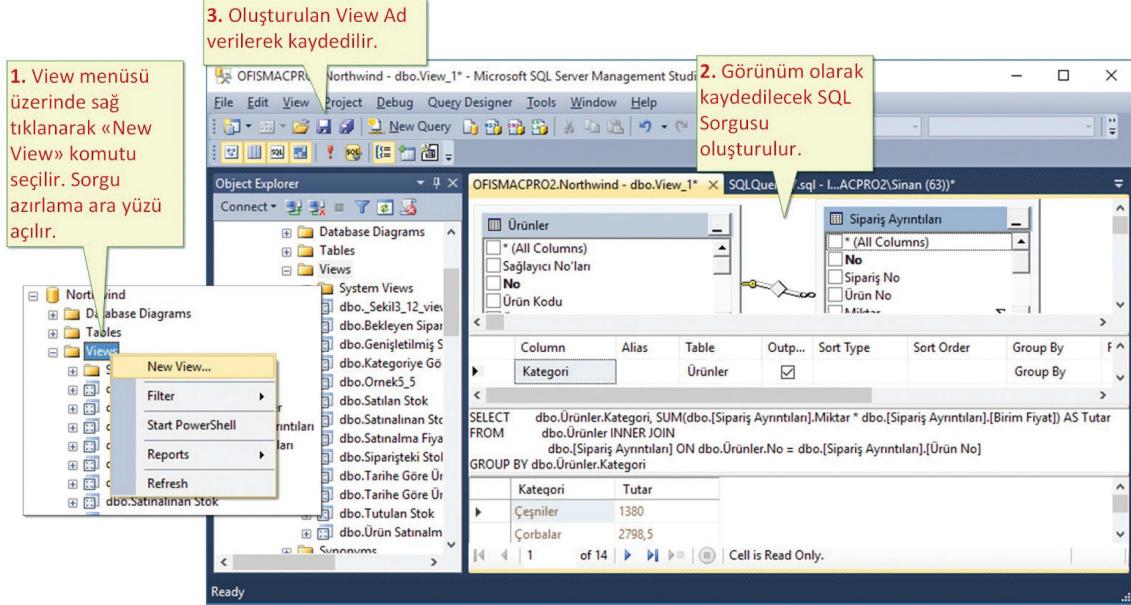


DİKKAT

Görünümler SQL komutu yazarak oluşturulabildiği gibi Management Studio Grafik ara yüzünde de oluşturulabilir. Şekil 5.15'te yer alan adım ve işlemler takip edilerek aynı görünüm oluşturulabilir.

**Şekil 5.15**

Görünümü (View) Grafik Arayüzde Oluşturmak İçin İzlenen Adımlar.



Oluşturulan bir görünümün silinmesi için **DROP** komutu kullanılır. Örneğin, uygulama 5.5'te oluşturulan [Ornek5\_5] görünümünü silmek için aşağıdaki SQL komut satırının **DROP VIEW Ornek5\_5** sorgu penceresinde çalıştırılması gereklidir.

## Saklı Yordam Oluşturma ve Silme İşlemleri

**Saklı Yordam**, sunucu üzerinde tutulan belirli bir görevi yerine getirmek için birden fazla tablo üzerinde işlem yapabilen, program içinden farklı parametreler ile çağrılarak kullanılabilen SQL tabanlı komut kümesidir. Saklı yordamlar veritabanı yönetim sistemi ile istemci yazılımlar arasında veri getirme, veri güncelleme ya da veri tabanındaki bir dizi işlemin gerçekleştirilmesi için yoğun olarak kullanılabilmektedir.

Kullanıcı tarafından oluşturulabilecek saklı yordamlar yerel saklı yordamlar (local stored procedures) olup kullanıcı tabanlı saklı yordamlar olarak da adlandırılırlar. Saklı yordamlar veritabanı üzerinde hızlı işlemler yapmak için kullanılan önemli araçlardan birisidir. Yerel saklı yordam oluşturmak için aşağıdaki yazım kuralı uygulanır. (-- ile başlayan satırlar açıklama satırıdır.)

```

CREATE PROCEDURE Procedureİsmi
    -- Parametre içermeyen Saklı yordamlarda alt satır yer almaz
    <@Param1> veri_türü, <@Param2> veri_türü
AS
BEGIN
    -- Sql Programlama ve seçme komutları
END

```

**Saklı Yordamlar** (Stored Procedures) veritabanlarında ihtiyaç olan tekrarlı işlemler veya ortak kullanım için oluşturulan komut kümeleridir.

Northwind örnek veritabanında tutar olarak en fazla satış yapılan 5 müşteriyi listeleyen saklı yordamı oluşturun.

**ÖRNEK 5.6**

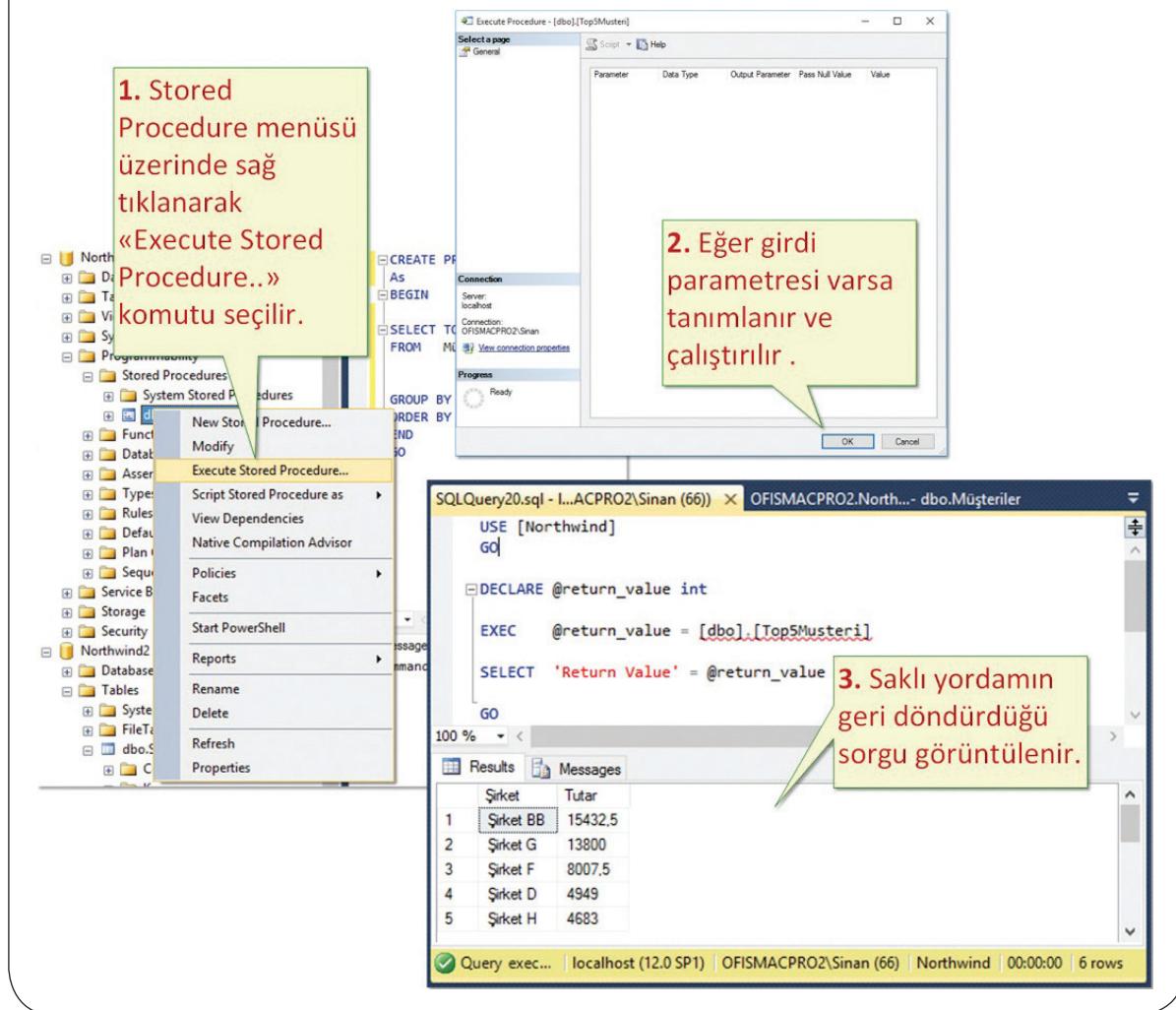
İstenilen veri kümesine ulaşmak için ilgili tablolar (**Müşteriler**, **Siparişler**, [**Sipariş Ayrıntıları**]) ilişkilendirilerek toplam satış miktarına (**SUM([Sipariş Ayrıntıları].Miktar \* [Sipariş Ayrıntıları]).Birim Fiyat**]) göre ters sıralayan (**ORDER BY Tutar DESC**) bir sorgu oluşturulur. Sorgunun sadece üstteki beş satırın görüntülenmesi için “**TOP (5)**” komutu kullanılmıştır. Bu sayede istenilen veriyi görüntüleyecek saklı yordam oluşturma komutu aşağıdaki gibi oluşturulur.

```
CREATE PROCEDURE Top5Musteri
AS
BEGIN
    SELECT TOP (5) Müşteriler.Şirket, SUM([Sipariş Ayrıntıları].
Miktar *
    [Sipariş Ayrıntıları].[Birim Fiyat]) AS Tutar
    FROM Müşteriler
    INNER JOIN Siparişler ON Müşteriler.No = Siparişler.[Müşteri
No]
    INNER JOIN [Sipariş Ayrıntıları] ON Siparişler.[Sipariş No] =
[Sipariş Ayrıntıları].[Sipariş No]
    GROUP BY Müşteriler.Şirket
    ORDER BY Tutar DESC
END
```

Oluşturulan saklı yordamın SQL Server Management Studio'daki görünümü ve çalışma adımları Şekil 5.16'da gösterilmiştir. Bu görüntüde bir saklı yordamın nasıl çalıştırıp sonucunun nasıl alındığı gösterilmiştir ancak bu yapılar veritabanı ve istemci arasında veri taleplerini ve işlemlerini yerine getiren yapıdır. Bir veri uygulaması ya da masaüstü istemci uygulaması veritabanındaki saklı yordamları çalıştırıbilmekte ve sonucunu kullanıcısına görüntülemektedir.

Şekil 5.16

Saklı Yordamın Çalıştırılması



Oluşturulan herhangi bir saklı yordamı silmek için **DROP PROCEDURE** SaklıYorda  
mİsmi komutu kullanılabilir. Örneğin, Örnek 5.7'de oluşturulan saklı yordamın silinmesi  
için aşağıdaki aşağıdaki komut uygulanmalıdır.

**DROP PROCEDURE [top5Musteri]**

Görünüm oluşturmada olduğu gibi saklı yordamlarda da önemli olan hedeflenen veri  
kümesi ya da komutların yazılmasıdır. Görüntüler sadece bir SQL sorgusunu çalıştırırken  
saklı yordamlarda Transactional SQL programlama dili kullanılabilir. Bu programla  
ma dilinde değişkenler oluşturulabilir, döngüler kurulabilir, kayıt kayıt veri tabloları işle  
nebilir hatta kullanıcılara e-posta gönderilebilir.

## Özet



### SQL ve DDL arasındaki ilişkiyi açıklamak

Yapılardırılmış Sorgu Dili (SQL)'in ilk sürümü 1970'li yıllarda SEQUEL olarak ortaya çıkmış olup, 1980'li yıllarda ANSI standartlarında tanımlamalar yapılmıştır. Daha sonra 1992 ve 1999 yıllarındaki bazı ek standart tanımlamaları ile bazı nesne tabanlı özellikler ve gömülü SQL özellikleri eklenmiştir. SQL veri sorgulama dilindeki önemli bileşenlerden birisi Veri Tanımlama Dilidir (DDL-Data Definition Language). DDL ayrı bir dil olmayıp, veritabanı nesnelerinin oluşturulması ve düzenlenmesi işlemlerini yapan SQL'in alt komut grubudur. DDL veritabanında tablolar, indeksler, görünümler gibi nesnelerin oluşturulması, düzenlenmesi ve silinmesi işlemlerini gerçekleştiren komutları içerir. DDL veri tabanında depolanan verilerle değil verileri barındıran, düzenleyen nesnelerle ilgili komutlardır.



### DDL ile gerçekleştirilecek işlemleri sıralamak

DDL dili ile veritabanlarında şema, tablo ve kısıtlar, görünüm (view) ve saklı yordam, tetikleyiciler ve benzeri nesnelerin oluşturulması, düzenlenmesi ve silinmesi sağlanır. Bu görevlerin dışında veritabanı yönetim süreçlerinde tanımlanan birçok işlem bu yapı içerisinde tanımlanmış komutlarla yerine getirilir.



### Veri tanımlama dilinde tablo ve indeks ile ilgili komutları açıklamak

Bir veritabanı yönetim sisteminde tablo oluşturabilmek için veri türlerinin, sınırlama işlevlerinin bilinmesi gereklidir. Veritabanı tablosu oluşturulmak için kullanılan komut **CREATE TABLE** komutudur. Bu komut ile farklı veri türlerindeki (tamsayı, kesirli sayı, ikili sayı, ikili sayı, metin, tarih vb.) alanlar tanımlanmaktadır. Ayrıca tablonun özelliklerine yönelik kısıtlamalar tanımlanabilir. Birincil ve yabancı anahtarlar, varsayılan değerler, alt ve üst değer sınırlamalar alanların tanımlanmasında ve veri tutarlığının sağlanması için kullanılabilmeğtedir. Bu amaçla kullanılan komutlara **DEFAULT, CHECK, IDENTITY, PRIMARY KEY, FOREIGN KEY** ifadeleri örnek verilebilir.

Veritabanı performansı açısından önemli nesne de indekslerdir. İndeksler **CREATE INDEX** komutu ile tablolardaki bir ya da birden fazla alan için oluşturulabilir.



### Görünüm ve saklı yordam ile ilgili DDL komutlarını tanımlamak

Görünümler(view), kullanıcıların veritabanındaki nesneleri sorgulayarak istedikleri veri kümelerini elde etmek için kullanılan yapılardır. Bu yapılar veriyi değil veriyi elde edecek sorgu komutlarını saklayarak çağrılarındakılarda dinamik olarak mevcut veriden istenilen veri kümesini üretirler. Bir görünüm "**CREATE VIEW** Görünüm\_Adı As <SQL Seçme Sorgusu>" söz dizimi ile oluşturulabileceği gibi grafik arayüzde oluşturulabilir. Saklı Yordam, sunucu üzerinde tutulan belirli bir görevi yinele getirmek için birden fazla tablo üzerinde işlem yapabilen, program içinden farklı parametreler ile çağrılarak kullanılabilen SQL tabanlı komut kümesidir. Bu yapıların oluşturulması için **CREATE PROCEDURE** komutu kullanılır.

## Kendimizi Sınayalım

- 1.** Aşağıdakilerden hangisi Veri Tanımlama Dili ile ilişkili **değildir**?
  - a. Tablo oluşturma
  - b. Mevcut bir tablonun bir alanını silme
  - c. Veritabanı üzerinde kullanıcı yetkisi oluşturma
  - d. XML'de veri aktarma
  - e. Saklı yordam silme
  
- 2.** Aşağıdakilerden hangisi veri tanımlama dili kapsamında veritabanında gerçekleşir?
  - a. Sorgu optimizasyonu
  - b. Saklı yordamın çağrılması
  - c. Tabloya ait verilerin listelenmesi
  - d. Kullanıcı haklarının tanımlanması
  - e. Bir tablo ile ilgili sonradan kısıt eklenmesi
  
- 3.** Veri Tanımlama Dili ile ilgili aşağıdaki ifadelerden hangisi **yanlıştır**?
  - a. Görünüm oluşturulabilir.
  - b. Akiş kontrolü yapılabılır.
  - c. ALTER olayını temel alan tetikleyici tanımlanabilir.
  - d. Tablo'nun bir alanı ile ilgili birincil anahtar tanımlanabilir.
  - e. Komutları SQL standartlarındadır.
  
- 4.** MS SQL Server üzerinde Veri tanımlama dili uygulamalarını yapabilmek için aşağıdakilerden hangisi son kullanıcı için imkan sağlar?
  - a. Veritabanı motoru servisleri
  - b. SQL Server Replication
  - c. Windows Power Shell 2.0
  - d. SQL Server Management Studio
  - e. Net Framework
  
- 5.** Bir tabloda herhangi bir alan için mutlaka veri girişi olması isteniyorsa SQL komutu seçeneği olarak aşağıdakilerden hangisi kullanılabilir?
  - a. NOT NULL
  - b. ALLOW NULLS
  - c. NCHAR
  - d. REFRESH
  - e. NEW DATABASE
  
- 6.** Bir işletme veritabanında müşterileri için; adı, soyadı ve maksimum 5000 olan müşteri sayısı için bir numara vb. bilgiler olacak şekilde bir tablo oluşturulmak istensin. Aşağıdakilerden hangi veri oluşturma SQL komutu bu talebi tam karşılar?
  - a. 

```
CREATE TABLE "Musteriler"(
    [MusteriID] nchar (10) NULL,
    [MusteriAdi] nchar (20) NULL,
    [MusteriSoyadi] nchar (20) NULL,
    CONSTRAINT [PK_Musteri] PRIMARY KEY
        ("MusteriID"))
```
  - b. 

```
SELECT MusteriID
FROM Musteriler
WHERE MusteriID==5000;
```
  - c. 

```
CREATE TABLE [Musteriler](
    [MusteriID] int NOT NULL,
    [MusteriAdi] nchar (20) NULL,
    [MusteriSoyadi] nchar (20) NULL)
```
  - d. 

```
CREATE TABLE [Musteriler](
    [MusteriID] nchar (5) NULL,
    [MusteriAdi] nchar (20) NULL,
    [MusteriSoyadi] nchar (20) NULL)
```
  - e. 

```
CREATE TABLE [Musteriler](
    [MusteriID] "int" IDENTITY (1, 1)
NOT NULL,
    [MusteriAdi] nvarchar (20) NULL,
    [MusteriSoyadi] nvarchar (20) NULL,
    CONSTRAINT [PK_Musteri] PRIMARY KEY
        ("MusteriID"))
```
  
- 7.** Bir işletme veritabanındaki tabloda çalışanların vatan-daşlık numarası, ad, soyadları ayrı ayrı tutulması planlanmaktadır. Herhangi bir bölümde aynı ad ve soyada sahip kişilerin çalışması istenmiyor. Bu isteği veritabanı üzerinde karşılayabilecek en uygun komut aşağıdakilerden hangisidir?
  - a. IDENTITY
  - b. PRIMARY KEY
  - c. UNIQUE
  - d. Attribute
  - e. CHECK

**8.** Aşağıdakilerden hangisi Yabancı Anahtar (Foreign Key) kullanım amacını en doğru tarif eder?

- a. Bir tablodan farklı bir tablodaki alana öznitelik için bağlantı tanımlamasını sağlar.
- b. Veritabanı yönetim sistemine dış bağlantı sırasında birincil olarak kullanılır.
- c. İlişkisel şemanın tablo olarak tanımlamasını sağlar
- d. SQL Server Management Studio'nun kullanımı için parola özelliği taşır.
- e. Tablodaki satırlar arası veri tekrarını önler.

**9.** E-ticaret hizmeti sunan bir işletme veritabanında müşteriler ile ilgili tabloda kişisel bilgiler içinde doğum tarihi bilgisi de tutulmaktadır. İşletmenin müşteri sayısı milyonun üzerinde olup işletmenin farklı yaş aralıklarındaki müşterilere ait bilgiler için farklı şubelerde gün içinde sürekli belli sorular ile veritabanında işlemler yapmaktadır. Bu veritabanında müşteriler tablosu üzerinde olan sorgularda hız artırımı için aşağıdakilerden hangisi kullanılabilir?

- a. CASCADE TABLE
- b. CREATE INDEX
- c. UNIQUE VELOCITY
- d. DROP TABLE
- e. CHECK

**10.** Aşağıdaki SQL komutlarından hangisi, veritabanı yönetim sistemlerinde raporlama amaçlı birden fazla tablo üzerinde sorgu oluşturma için kullanılabilir?

- a. FOREIGN KEY...REFERENCES
- b. CREATE DATABASE
- c. USE
- d. CREATE VIEW
- e. CREATE TABLE

## Kendimizi Sınayalım Yanıt Anahtarları

- |       |   |
|-------|---|
| 1. d  | Yanınız yanlış ise “Veri Tanımlama” konusunu yeniden gözden geçiriniz.  |
| 2. e  | Yanınız yanlış ise “Veri Tanımlama Dili” konusunu yeniden gözden geçiriniz.   |
| 3. b  | Yanınız yanlış ise “Veri Tanımlama Dili” konusunu yeniden gözden geçiriniz.   |
| 4. d  | Yanınız yanlış ise “Örnek Bir Veritabanı Yönetim Sistemi Kurulumu” konusunu yeniden gözden geçiriniz.                 |
| 5. a  | Yanınız yanlış ise “Veri Tanımlama Dili Tablo İşlemleri” konusunu yeniden gözden geçiriniz.                           |
| 6. e  | Yanınız yanlış ise “Veri Tanımlama Dili Tablo İşlemleri” konusunu yeniden gözden geçiriniz.                           |
| 7. c  | Yanınız yanlış ise “Tablolarda Anahtar ve Diğer Büyünlük Kısıtlarının Tanımlaması” konusunu yeniden gözden geçiriniz. |
| 8. a  | Yanınız yanlış ise “Veri Tanımlama Dili Tablo İşlemleri” konusunu yeniden gözden geçiriniz.                           |
| 9. b  | Yanınız yanlış ise “Veri Tanımlama Dili İndeks İşlemleri” konusunu yeniden gözden geçiriniz.                          |
| 10. d | Yanınız yanlış ise “Veri Tanımlama Dili Diğer İşlemler” konusunu yeniden gözden geçiriniz.                            |

## Sıra Sizde Yanıt Anahtarları

### Sıra Sizde 1

Görünüm, önemli bir güvenlik aracı olarak kullanılabilir. VTYŞdeki bazı tablolar kullanıcılar açılmadan sadece gerekli alanları erişime açılabilir. Bu şekilde, kullanıcılar ihtiyacı olan verileri alırken tablolardaki diğer veriler korunmuş olur. Diğer taraftan, görünüm ile karmaşık bazı sorgular bir defa oluşturularak farklı kullanıcılar tarafından tekrar kullanılabilir. Görünüm oluşturma sırasında alan isimleri yeniden isimlendirilebileceği için daha basit ve ezberlenebilir isimler atanabilir. Görünüm ile çok katmanlı sorgu yapısı oluşturmak da mümkündür. Görünüm içinde oluşturulacak bir veri daha sonra diğer sorgulara girdi olarak kullanılabilir.

### Sıra Sizde 2

```
CREATE TABLE [Calisanlar1] ([CalisanTCNo]“int” IDENTITY (1, 1) NOT NULL,
[CalisanAdi] nvarchar (40) NOT NULL,
[CalisanMaas] [money] NULL DEFAULT (0),
CONSTRAINT [PK_Calisan] PRIMARY KEY ([CalisanTCNo]),
CONSTRAINT “CK_Maas” CHECK (“CalisanMaas”
>=1300 AND “CalisanMaas”<=15000))
```

### Sıra Sizde 3

```
CREATE TABLE “Malzemeler” (
“MalzemeNo” “int” IDENTITY (1, 1) NOT NULL,
“MalzemeAdi” nvarchar (40) NULL,
CONSTRAINT “PK_Malzemeler” PRIMARY KEY (“MalzemeNo”)
)

CREATE TABLE “MalzemeSatis” (
“MalzemeNo” “int” NOT NULL,
“Adet” “int” NOT NULL,
“SatisTarih” “datetime” NULL ,
CONSTRAINT “PK_MalzemeSatis” PRIMARY KEY (“MalzemeNo”), 
CONSTRAINT “FK_Satislar” FOREIGN KEY (“MalzemeNo” )
REFERENCES “dbo”.“Malzemeler” (“MalzemeNo ”)
)
```

### Sıra Sizde 4

İndeks MS SQL Serverda iki şekilde de oluşturulabilir. Kümeleme indekslerde satırların fiziksel kayıt alanında sıralaması değişir. Kayıtlar, bir sözlüğün alfabetik harf sırasına göre dizilmesi gibi dizilip o sırada erişim hızlı olur. Normal kümelemediş indeks ise, kitapların son sayfasında olan indekse benzer. Veriyi tüm kitapta aramadan, arka sayfadaki kendisi harf sırasında dizilmiş indekse bakarak aradığınızın hangi sayfada olduğunu bulabilirsiniz. Kümeleme indeks diğerine göre daha hızlıdır. Normal bir tablo sadece bir adet kümelemediş indekse sahip olabilir. Kümelemediş indeks sayısı fazla olabilmekle beraber, her indeks için ayrı hafıza alanı gerekiyorunda hafıza ihtiyacı bir dezavantajdır.

## Yararlanılan ve Başvurulabilcek Kaynaklar

- Özseven, T. (2014). **Veri Tabanı Yönetim Sistemleri 2**, Trabzon, Türkiye.
- Ramakrishnan R., Gehrke J., (2003). **Database Management Systems**, Third Edition, McGraw-Hill.
- 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.
- Yarımagań, Ü (2010). **Veri Tabanı Yönetim Sistemleri**, Akademî Yayıncılık.
- Silberschatz, A., Korth, H. F., Sudarshan, S., (2006). **Database System Concepts**, McGraw-Hill.
- Transact-SQL Reference: <https://msdn.microsoft.com/en-us/library/bb510741.aspx>, son erişim tarihi: 12.11.2015