

 **eKampus**
anadolum
e K a m p ü s
ve
anadolu mobil
dilediğin yerden,
dilediğin zaman,
öğrenme fırsatı!



(ekampus.anadolu.edu.tr)



(mobil.anadolu.edu.tr)

ekampus.anadolu.edu.tr



Açıköğretim Destek Sistemi
Açıköğretim Sistemi ile ilgili

merak ettiğiniz her şey AOS Destek Sisteminde...

- Kolay Soru Sorma ve Soru-Yanıt Takibi
- Sıkça Sorulan Sorular ve Yanıtları
- Canlı Destek (Hafta İçi Her Gün)
- Telefonla Destek

aosdestek.anadolu.edu.tr

AOS DESTEK Sistemi İletişim ve Çözüm Masası

0850 200 46 10

www.anadolu.edu.tr

T.C. ANADOLU ÜNİVERSİTESİ YAYINI NO: 3422
AÇIKÖĞRETİM FAKÜLTESİ YAYINI NO: 2272

VERİTABANI SİSTEMLERİ

Yazarlar

Dr.Öğr.Üyesi Şenay LEZKİ (Ünite 1)

Dr.Öğr.Üyesi Nihat ADAR (Ünite 2, 3, 8)

Öğr.Gör. Özlem ERORTA (Ünite 4)

Doç.Dr. Ahmet YAZICI (Ünite 5)

Doç.Dr. Şerafettin ALPAY (Ünite 6)

Dr.Öğr.Üyesi Sinan AYDIN (Ünite 6)

Doç.Dr. Kemal ÖZKAN (Ünite 7)

Editörler

Prof.Dr. Harun SÖNMEZ

Doç.Dr. Sinan AYDIN

Bu kitabın basım, yayım ve satış hakları Anadolu Üniversitesi'ne aittir.
“Uzaktan Öğretim” teknigine uygun olarak hazırlanan bu kitabı bütün hakları saklıdır.
İlgili kuruluştan izin alınmadan kitabı tümü ya da bölümleri mekanik, elektronik, fotokopi, manyetik kayıt
veya başka sekillerde çoğaltılamaz, basılamaz ve dağıtılamaz.

Copyright © 2016 by Anadolu University
All rights reserved

No part of this book may be reproduced or stored in a retrieval system, or transmitted
in any form or by any means mechanical, electronic, photocopy, magnetic tape or otherwise, without
permission in writing from the University.

Öğretim Tasarımcıları

*Prof.Dr. Tevfik Volkan Yüzer
Dr.Öğr.Üyesi Halil Cem Sayın*

Grafik Tasarım Yönetmenleri

*Prof. Tevfik Fikret Uçar
Doç.Dr. Nilgün Salur
Öğr.Gör. Cemalettin Yıldız*

Dil ve Yazım Danışmanı

Öğr.Gör. Hülya Düzenli

Ölçme Değerlendirme Sorumlusu

Emrah Emre Özkeskin

Kapak Düzeni

Prof.Dr. Halit Turgay Ünalan

Grafikerler

*Burcu Güler
Kenan Çetinkaya*

Dizgi ve Yayıma Hazırlama

Kitap Hazırlama Grubu

Veritabanı Sistemleri

E-ISBN

978-975-06-3244-0

Bu kitabın tüm hakları Anadolu Üniversitesi'ne aittir.

ESKİŞEHİR, Ocak 2019

3108-0-0-2202-V01

İçindekiler

Önsöz	ix
-------------	----

Veritabanı Sistemlerinin Temelleri	2
GİRİŞ	3
Veritabanının Tarihçesi	3
VERİTABANINA İLİŞKİN TEMEL KAVRAMLAR	5
GELENEKSEL DOSYA SİSTEMLERİ VE VERİTABANI YÖNETİM SİSTEMLERİ	6
VERİTABANI YÖNETİM SİSTEMLERİNİN AVANTAJLARI	8
Gereksiz Veri Tekrarı ve Veri Tutarlılığının Önlenmesi	8
Veri Bütünlüğünün Sağlanması	9
Veri Paylaşımının Sağlanması	9
Kullanımda Üst Düzey Uzmanlık Gerektirmemesi	9
Verilerin Gizliliğinin ve Güvenliğinin Sağlanması	10
Standart Yapı ve Kuralların Uygulanabilir Olması	11
VERİTABANI KULLANICILARI	11
Veritabanı Sorumluları	11
Veritabanı Yöneticisi	12
Veritabanı Tasarımcısı	12
Son Kullanıcılar	13
Sistem Analistleri ve Uygulama Programcılar	13
VERİTABANI YÖNETİM SİSTEMLERİNİN MİMARİSİ	14
Veri Modelleri	14
Veri Modellerinin Sınıflandırılması	14
Şemalar, Örnekler ve Veritabanının Durumu	15
Üç Şema Mimarisi	15
Veri Bağımsızlığı	16
Veritabanı Yönetim Sistemlerinde Kullanılan Diller	17
Veritabanı Yönetim Sistemlerinin Bileşen Modülleri	18
VERİTABANI TÜRLERİ	20
Hiyerarşik Veritabanı	20
Ağ Veritabanı	21
İlişkisel Veritabanı	21
Nesneye Yönelik Veritabanı	21
VERİTABANI YÖNETİM SİSTEMİ YAZILIMLARI	22
Özet	23
Kendimizi Sinayalım	25
Kendimizi Sinayalım Yanıt Anahtarları	26
Sıra Sizde Yanıt Anahtarları	26
Yararlanılan ve Başvurulabilecek Kaynaklar	27

Veritabanı Tasarımı	28
----------------------------------	-----------

GİRİŞ	29
VERİTABANI TASARIM AŞAMALARI	29
Kavramsal Veri Modeli	29

1. ÜNİTE

2. ÜNİTE

Kavramsal Veri Modelinin Özellikleri	30
Mantıksal Veri Modeli	30
Mantıksal Veri Modelinin Özellikleri	30
Fiziksel Veri Modeli	30
Fiziksnel Veri Modelinin Özellikleri	31
VARLIK İLİŞKİ MODELLEME	31
Varlıklar (Entity)	31
Güçlü ve Zayıf Varlık Kümeleri	32
Öznitelikler (Attributes)	32
Zorunlu ve Seçimli Öznitelikler	33
Basit ve Birleşik Öznitelikler	33
Türetilen Öznitelikler	33
Tek ve Çok Değerli Öznitelikler	33
Anahtar Öznitelik	34
Etki Alanı	34
İlişki	34
İlişkisel Varlıklar	35
İlişki Derecesi	36
İlişki Türleri	37
Varlık İlişki Diyagramlarında Kullanılan Şekiller	38
VARLIK İLİŞKİ DİYAGRAMININ TABLOLARA DÖNÜŞTÜRÜLMESİ	39
Bire Bir İlişkilerin Tablolara Dönüşürülmesi	39
Bire Çok veya Çoka Bir İlişkilerin Tablolara Dönüşürülmesi	40
Çoka Çok İlişkilerin Tablolara Dönüşürülmesi	41
Çok Değerli Özniteliklerin Tablolara Dönüşürülmesi	42
VERİTABANI YAŞAM DÖNGÜSÜ	43
Veritabanı Yaşam Döngüsü Fazları	44
Gereksinim Analizi	44
Veritabanı Tasarımı	44
Uygulama ve Yükleme	44
Operasyon	44
Bakım	44
Özet	45
Kendimizi Sınayalım	46
Kendimizi Sınayalım Yanıt Anahtarı	47
Sıra Sizde Yanıt Anahtarı	47
Yararlanılan ve Başvurulabilecek Kaynaklar	47
3. ÜNİTE İlişkisel Veritabanı Modeli	48
GİRİŞ	49
İLİŞKİSEL CEBİR	50
Birleşim (\cup)	51
Kesişim (\cap)	52
Fark (-)	52
Yansıtma (π)	52
Seçim (σ)	52
Kartezyen Çarpım (x)	52

Şartlı Bitişme (\bowtie_0)	53
Doğal Bitişme (\bowtie)	53
Bölme (\div)	53
Yeniden Adlandırma (ρ)	54
İLİŞKİSEL VERİTABANI NESNELERİ VE KAVRAMLAR	54
Tablolar, İlişkisel Örnek	54
Kayıtlar, Satırlar ve Tuplelar	55
Alanlar, Sütunlar, Öz nitelikler	56
Veri Tipleri	56
Basit Veri Tipleri	56
Karmaşık Veri Tipleri	57
Özelleştirilmiş Veri Tipleri	58
Null Değerinin Kullanımı	58
Kısıtlar (Constraint)	59
NOT NULL Kısıtı	59
DEFAULT Kısıtı	59
Anahtar Kısıtı	59
Birincil Anahtar (Primary Key)	60
Benzersiz Anahtar (Unique Key)	60
Yabancı Anahtar (Foreign Key)	60
CHECK Kısıtı	61
Görünümler (Views)	61
İndeksler (Indexes)	62
Saklı Yordamlar (Stored Procedures)	63
İLİŞKİSEL VERİTABANI ÖRNEĞİ	63
Northwind Veritabanı	63
Özet	67
Kendimizi Sinayalım	68
Kendimizi Sinayalım Yanıt Anahtarı	70
Sıra Sizde Yanıt Anahtarı	70
Yararlanılan ve Başvurulabilecek Kaynaklar	70

Masaüstü Veritabanı Sistemleri**72****4. ÜNİTE**

GİRİŞ	73
MASAÜSTÜ VERİTABANLARI	74
MS Access Veritabanı Yazılımı	74
MS Access'te Masaüstü Veritabanı Oluşturma	76
TABLOLAR, FORMULAR, RAPORLAR	77
Tablolar	78
Tablo İlişkileri	79
Anahtarlar	80
Bilgi Tutarlılığı	81
Formlar	81
Boş Form Oluşturma	82
Bölünmüş Form Oluşturma	82
Tablo ya da Sorgudan Form Oluşturma	83
Birden Çok Kayıt Görüntüleyen Form Oluşturma	83

Alt Form İçeren Form Oluşturma	83
Gezinti Formu Oluşturma	84
Raporlar	84
GRAFİK ARAYÜZ İLE VERİ SORGULAMA	85
Özet	89
Kendimizi Sınayalım	90
Kendimizi Sınayalım Yanıt Anahtarı	91
Sıra Sizde Yanıt Anahtarı	91
Yararlanılan ve Başvurulabilecek Kaynaklar	91
Yararlanılan ve Başvurulabilecek İnternet Adresleri	91

5. ÜNİTE**Veri Tanımlama 92**

GİRİŞ	93
VERİ TANIMLAMA DİLİ	94
Şema Oluşturma	94
Tablo ve Kısıtların Oluşturulması	95
İndeks Oluşturma	95
Görünüm Oluşturma	95
ÖRNEK BİR VERİTABANI YÖNETİM SİSTEMİ KURULUMU	95
Kurulumu Hazırlık	96
MS SQL Server 2014 Express Kurulumu	96
SQL Server Management Studio Kurulumu	99
Veritabanı İşlemleri	100
MS SQL Komutları ile Veritabanı Oluşturma	100
MS SQL Komutları ile Veritabanı Silme	101
Ms Sql Server 2014 Yardımcısı ile Örnek Veritabanını Oluşturma	101
VERİ TANIMLAMA DİLİ TABLO İŞLEMLERİ	102
Tablolarda Öz nitelik Veri Tipleri ve SQL Kısıt Tanımlaması	102
Tablolarda Anahtar ve Diğer Bütünlük Kısıtlarının Tanımlaması	104
Mevcut Tablolarda Değişiklik Yapılması	106
VERİ TANIMLAMA DİLİ İNDEKS İŞLEMLERİ	107
VERİ TANIMLAMA DİLİ DİĞER İŞLEMLER	108
Görünüm Oluşturma ve Silme İşlemleri	109
Saklı Yordam Oluşturma ve Silme İşlemleri	110
Özet	113
Kendimizi Sınayalım	114
Kendimizi Sınayalım Yanıt Anahtarı	116
Sıra Sizde Yanıt Anahtarı	116
Yararlanılan ve Başvurulabilcek Kaynaklar	117

6. ÜNİTE**Veri Sorulama 118**

GİRİŞ	119
SEÇME (SELECT) KOMUTU	119
Benzersiz Değerlerin Elde Edilmesi (DISTINCT)	121
Sorgu Sonuçlarının Sıralanması (ORDER BY)	122
Seçimlerin Sınırlanması (WHERE)	122
BİRDEN FAZLA TABLODAN VERİ SEÇİMİ	124
JOIN Komutu ile Tabloların Bağlanması	124

Cök Sayıda Tablonun Bağlanması	127
Veri Kümelerinin Birleştirilmesi	128
GRUPLAMA VE ÖZETLEME SORGULARI	129
Özetleme İşlevleri	132
Özetlenen Değerlerin Sınırlanması	134
SQL İŞLEVLERİ	134
Mantıksal İşlevler	135
Sayısal İşlevler	136
Metin İşlevleri	137
Tarih İşlevleri	138
ALT SORGULAR	139
Alt Sorguların Tablo Olarak Kullanılması	140
Alt Sorguların Kısıtlayıcı Olarak Kullanılması	141
Alt Soru Üst Sorguların İlişkilendirilmesi	142
Özet	144
Kendimizi Sınayalım	145
Kendimizi Sınayalım Yanıt Anahtarı	146
Sıra Sizde Yanıt Anahtarı	146
Yararlanılan ve Başvurulabilecek Kaynaklar	147
Veri İşleme	148
GİRİŞ	149
SQL SERVER MANAGEMENT STUDIO KULLANIMI	151
TABLOYA KAYIT EKLEME	154
INSERT INTO Komutu	155
Tabloya Satır Ekleme İşlemlerinde Karşılaşılan Hatalar	158
Veri Girişinde Yaygın Kullanılan İşlevler	159
NULL Değerler	160
TABLODAN KAYIT SİLME	163
Silme Komutu	164
Birden Fazla Tablo ile Eylem Sorguları	165
TABLO VERİLERİNİ DÜZENLEME	167
TRANSACTION	169
ROLLBACK ve COMMIT Kavramı	170
Özet	173
Kendimizi Sınayalım	174
Kendimizi Sınayalım Yanıt Anahtarı	175
Sıra Sizde Yanıt Anahtarı	175
Yararlanılan ve Başvurulabilecek Kaynaklar	175
Normalleştirme	176
GİRİŞ	177
FONKSİYONEL BAĞIMLILIK	177
Kısmi Bağımlılık (Partial Dependence)	179
Geçişli Bağımlılık (Transitive Dependence)	179
Tam Fonksiyonel Bağımlılık (Full Functional Dependence)	179
Çok Değerli Bağımlılık (Multiple Valued Dependence)	179

7. ÜNİTE**8. ÜNİTE**

Döngüsel Bağımlılık (Cyclic Dependency)	180
AYKIRILIKLAR	180
Ekleme Aykırılığı	180
Silme Aykırılığı	181
Güncelleme Aykırılığı	182
NORMALLEŞTİRME	182
Normalleştirme Amaçları	182
Normalleştirme Aşamaları	182
NORMAL FORMLAR	184
Birinci Normal Form (1NF)	185
İkinci Normal Form (2NF)	186
Üçüncü Normal Form (3NF)	188
3NF'de Çok Çok İlişki Durumu	188
3NF'de Geçişli Bağımlılık Durumu	188
3NF'de Hesaplanmış Değerleri Saklayan Alanların Durumu	189
BCNF, 4NF ve 5NF	190
Özet	191
Kendimizi Sınayalım	192
Kendimizi Sınayalım Yanıt Anahtarı	193
Sıra Sizde Yanıt Anahtarı	194
Yararlanılan ve Başvurulabilecek Kaynaklar	195

Önsöz

Sevgili öğrenciler,

Bu kitapta veritabanı sistemlerine ilişkin konular açıklamalı örneklerle ele alınmıştır. Konuların anlatımında uygulamaya ağırlık verilmiştir. Her ünitenin başında yer alan **Amaçlar** ve **Anahtar Kavramlar**, öğrencilerin ilgili üniteden ele edecekleri kazanımları göstermektedir. Öğrenciler, ünitelere çalışıktan sonra bu kazanımları elde edip etmediklerini ilgili **Amaçlar** ve **Anahtar Kavramlar** yardımıyla kolaylıkla sorgulayabilecektir. Her ünite içinde yer alan **Sıra Sizde** soruları, üniteyi okuyan öğrencilerin çalışmaları konuyu kavrayıp kavrayamadığını anlamak için verilmiştir. Ünite sonunda yer alan **Kendimizi Sınayalım** kısmında öğrenciler, ilgili ünitede öğrendiklerinin küçük bir sınamasını yapabileceklerdir.

Öğrencilerin bu kitaba çalışırken veritabanı sistemleriyle ilgili çeşitli komutları ezberleyerek değil problemleri nasıl çözeceğine yönelik olarak konuları iyice kavrayıp, anlamaya yönelik çalışmalar yapması gelecekte karşılaşacakları problemlerin çözümünde çok daha etkili olacaktır.

Bu kitabın meydana gelmesi için, kitap ekibinin oluşturulması ve bu ekibin çalışmalarının yürütülmesinde her türlü imkanı sağlayan Anadolu Üniversitesi Rektörlüğüne, Açıköğretim Fakültesi Dekanlığına ve Kitap Tasarım Genel Koordinatörlüğü nezdinde kitabın hazırlanması için emeği geçen çalışanlara editör ve yazarlar olarak teşekkür ederiz.

Editörler

Prof.Dr. Harun SÖNMEZ

Doç.Dr. Sinan AYDIN

1

Amaçlarımız

Bu üniteyi tamamladıktan sonra;

- 🕒 Veritabanına ilişkin temel kavramları tanımlayabilecek,
 - 🕒 Geleneksel dosya sistemleri ile veritabanı yönetim sistemlerini karşılaştırabilecek,
 - 🕒 Veritabanı kullanıcılarını sınıflandırabilecek,
 - 🕒 Veritabanı yönetim sistemlerinin mimarisini açıklayabilecek,
 - 🕒 Veritabanı türlerini ve yaygın olarak kullanılan veritabanı yönetim sistemi yazılımlarını sıralayabilecek
- bilgi ve becerilere sahip olacaksınız.

Anahtar Kavramlar

- Veri
- Veritabanı
- Veritabanı Yönetim Sistemi
- Veri Modeli
- Veritabanı Şeması
- Veri Tanımlama Dili
- Veri İşleme Dili
- Sorgulama Dili

İçindekiler

Veritabanı Sistemleri

Veritabanı Sistemlerinin Temelleri

- 
- GİRİŞ
 - VERİTABANINA İLİŞKİN TEMEL KAVRAMLAR
 - GELENEKSEL DOSYA SİSTEMLERİ VE VERİTABANI YÖNETİM SİSTEMLERİ
 - VERİTABANI YÖNETİM SİSTEMLERİNİN AVANTAJLARI
 - VERİTABANI KULLANICILARI
 - VERİTABANI YÖNETİM SİSTEMLERİNİN MİMARİSİ
 - VERİTABANI TÜRLERİ
 - VERİTABANI YÖNETİM SİSTEMİ YAZILIMLARI

Veritabanı Sistemlerinin Temelleri

GİRİŞ

Günümüzde küçük ya da büyük tüm işletmeler faaliyetlerini yerine getirebilmek, rakipleriyle yarışabilmek ve varlıklarını başarılı bir biçimde sürdürabilmek için bilgiye ihtiyaç duyarlar. Bu nedenle bilgi, günümüzde üretim faktörleri içinde yer almaktır ve sermaye, ham madde, enerji ve iş gücü olarak sıralanan geleneksel üretim faktörleri kadar önem taşımaktadır. Bununla birlikte bilginin istenilen amaca hizmet edebilmesi için zamanında elde edilebilmesi ve amaca uygun olması gereklidir. İşletmelerin faaliyetlerini düzenli ve etkin bir biçimde yerine getirebilmelerinde hayatı bir öneme sahip olan bilginin zamanında elde edilebilmesi ve ilgili kişi ve birimlere ihtiyaç duyulduğu anda iletilebilmesi ise bilgi sistemleri aracılıyla gerçekleştirilir. Bilgi sistemleri; ilgili verilerin toplanması, bu veriler üzerinde birtakım işlemler uygulanarak verilerin yararlı bilgilere dönüştürülmesi ve ilgili kişi ve birimlere iletilmesini, belirli bir düzen içinde gerçekleştiren sistemlerdir. İşletmelerde verilerin toplanması ve işlenmesi için kullanılan araçlar, teknikler, yaklaşımlar geçmişle günümüz arasında farklılık gösterse de bilgi sistemlerinin taşıdığı önem aynıdır. Teknoloji ve bilgisayar sektöründeki gelişmelere paralel olarak günümüzde bilgi sistemleri, bilgisayar donanımı, yazılım, veri kaynakları, ağ teknolojileri ve insan bileşenlerinden oluşmaktadır. Bu bileşenlerden veri kaynakları, işletmede ihtiyaç duyulan bilgiyi üretmek amacıyla kullanılacak verinin saklandığı (depolandığı) ve ihtiyaç duyulduğu anda erişilebileceği bileşeni ifade etmektedir. Bu kitapta ele alınan veritabanı sistemleri de kısaca veri kaynaklarının yönetiminden sorumlu sistemlerdir. Veritabanı sistemlerine ilişkin ayrıntılı bilgilerin yer aldığı izleyen ünitelerden önce bu ilk üniteye veritabanı sistemlerine ilişkin temel kavramlar ve genel bilgiler ele alınmıştır.

Veritabanının Tarihçesi

Bir konuya ilgili verilere hızlı ve kolay biçimde erişebilme isteği, verilerin düzenli biçimde saklanması ihtiyacını doğurmış, dolayısıyla veritabanı olsusunun bu adla olmasa bile geçmişten günümüze kadar bir biçimde hayatımızda yer almasına neden olmuştur. Geçmişle günümüz arasındaki fark kullanılan araç, gereç, teknik ve yaklaşımlarda ortaya çıkmaktadır. Geçmişte veriler fiziksel olarak, kağıt, defter, dosya kullanımı ile saklanmaktaydı. Bu anlamda yalnızca ev telefonlarının olduğu geçmişte, iletişim kurmak istediğimiz kişilerin ad, soyad ve telefon numaralarının yer aldığı ve klasik defter biçiminde düzenlediğimiz telefon rehberleri örnek olarak verilebilir. Benzer biçimde devlet kurumları, hastaneler ve ticari kuruluşlar da ihtiyaç duydukları verileri kağıt dosyalar biçiminde ve belirli bir düzende arşivlemekte ve gereklili olduğunda bu arşivlerde arama yaparak istenen

veriye erişmekteydi. Bu tür sistemlerde, arşivde istenen bilgiye erişmek için belirli kurallar ve düzenlemeler uygulanmaktadır. Kütüphanelerde herhangi bir kitaba erişmek için kullanılan kitap künhe bilgisi bu tür düzenlemelere örnek olarak verilebilir. Bu sistemlerin bazı temel prensipleri günümüzde hâlâ kullanılmaktadır. Düzenlenmesi ve saklanması gereken veri miktarının giderek artması ve bu verilerin fiziksel olarak uzun süre saklanabilmesi, korunabilmesi ve istendiğinde hızlıca erişilebilmesi kolay bir iş değildi. Söz konusu bu zorlukların üstesinden gelebilme istek ve ihtiyacı bilgisayar teknolojilerindeki gelişmelerin temel nedenlerinden biri olmuştur. Bilgisayar teknolojisinde bugün gelinen noktada çok büyük miktarda ve karmaşık yapıda olan verilerin saklanması ve yönetilmesi oldukça kolaylaşmıştır. Bu nedenle günümüzde veritabanı denilince akıllara bilgisayar ortamında oluşturulan veritabanları gelmektedir.

Bilgisayar kullanımının özel işletmeler için uygun maliyetli bir seçenek olmaya başladığı dönem olan 1960'larda veritabanlarının bilgisayarlar yardımıyla oluşturulması dönemin de başlamıştır. İzleyen kesimde 1960'lı yıllarda itibaren veritabanı uygulamalarının tarihsel gelişimine ilişkin kısa bilgilere yer verilmiştir.

1960'lar: 1960'lı yılların başında Charles Bachman tarafından IDS (Integrated Data Store-Bütünleştirilmiş Veri Depolama) adıyla ilk genel amaçlı veritabanı yönetim sistemi geliştirilmiştir. 1960'ların sonunda ise IBM tarafından IMS (Information Management System-Bilgi Yönetim Sistemi) adıyla ilk ticari VTYS (Veritabanı Yönetim Sistemi) geliştirilmiş ve bu yapı hiyerarşik veri modeline temel teşkil etmiştir (Vural ve Sağiroğlu, 2010, s.72).

1967'de CODASYL (COnference on Data SYstem Languages) grubu kullanıcı ve bilgisayar üreticileri için COBOL dili standartlarını belirlemiştir ve bu standartlar ANSI (American National Standards Institute-Amerikan Ulusal Standartlar Enstitüsü) tarafından kabul edilmiştir. Bu standartlaştırma başarısından sonra CODASYL grubu veritabanı standartlarını oluşturmak için DBTG'yi (DataBase Task Group-Veritabanı Görev Grubu) kurarak grubu veritabanı standartlarını oluşturmak üzere görevlendirmiştir. DBTG yaptığı çalışmalar sonucunda ağ veri modelleri için dil özelliklerini, veritabanı şemalarını, veri tanımlama ve veri işleme dili gibi önemli veritabanı bileşenlerini tanımlamıştır.

1969'da Edgar F. Codd tarafından geliştirilen ilişkisel veritabanı modeli, insanların veritabanlarılarındaki düşüncelerini değiştirecek nitelikte bir gelişme olarak ortaya çıkmıştır. Modelde, veritabanının şeması veya mantıksal organizasyonu, fiziksel bilgi deposundan bağımsız hâle getirilmiş ve bu da veritabanları için standart bir prensip hâline gelmiştir.

1970'ler: 1974 ve 1977 yılları arasında iki önemli ilişkisel veritabanı sistemi prototipi oluşturulmuştur. Bunlardan biri UBC'de geliştirilen Ingres, diğer ise IBM San Jose'de oluşturulan System R'dir. Ingres, QUEL diye bilinen sorgulama dili olarak kullanılmış ve Ingres Corp., MS SQL Server, Sybase, Wang's PACE, ve Britton-Lee gibi sistemlerin oluşumuna öncülük etmiştir. Diğer yandan System R, SEQUEL sorgulama dilini kullanmış ve SQL/DS, DB2, Allbase, Oracle, ve Non-Stop SQL'in gelişimine katkıda bulunmuştur. Ayrıca, İlişkisel Veritabanı Yönetim Sistemi (RDBMS) bu dönemde bilinen bir terim hâline gelmiştir.

Varlık-İlişki (Entity-Relationship) diye adlandırılan yeni bir veritabanı modeli, Peter Chen tarafından 1976 yılında önerilmiştir. Bu model, tasarımcıların mantıksal tablo yapısı yerine, veri uygulamasına odaklanmalarını mümkün kılmıştır.

1980'ler: Yapılandırılmış Sorgu Dili (SQL) standart sorgu dili hâline gelmiştir. Buna paralel olarak ilişkisel veritabanı sistemleri, bilgisayar satışlarındaki hızlı yükselişin veritabanı piyasasını canlandırmasıyla ticari bir başarıya dönüşmüştür. Bu da ağ ve hiyerarşik veritabanı modellerinin popüleritesinde büyük bir düşüse yol açmıştır. Yine bu yıllarda DB2, IBM'in popüler veritabanı ürünü olmuştur. IBM PC'nin tanıtımı da birçok yeni veritabanı şirketinin kuruluşuna ve PARADOX, RBASE 5000, RIM, Dbase III ve IV, OS/2 Database Manager ve Watcom SQL gibi ürünlerin gelişimine yol açmıştır.

1990'lar: 1990'ların başında veritabanı endüstrisinde yaşanan kriz nedeniyle bu sektörde varlığını sürdürmek isteyen şirketler krizi fırsat bilerek karmaşık veritabanı ürünlerini yüksek fiyatlarla satabilmişlerdir. Bu sıralarda, Oracle Developer, PowerBuilder, VB gibi uygulama geliştirme araçları piyasaya sunulmuştur. Ayrıca Access programı da 1990'lı yılların başında piyasaya sürülmüştür. Yine bu dönemde Nesne Veritabanı Yönetim Sistemleri (ODBMS) için prototipler de geliştirilmiştir. 1990'ların ortalarında internetin kullanılma-ya başlanması veritabanı endüstrisinde hızlı bir büyümeye yol açmıştır.

1990'ların sonlarında online işletmelere artan yatırımlar nedeniyle Front Page, Active Server Pages, Java Servelets, Dream Weaver, ColdFusion, Enterprise Java Beans, ve Oracle Developer 2000 gibi internete bağlı veritabanı bağlayıcılarına olan talep miktarı belirgin olarak artmış, yine bu dönemlerde MySQL, Apache, vb. gibi açık kaynak kodlu veritabanı yönetim sistemi yazılımları kullanıma sunulmuştur.

2000'ler: Internet endüstrisi 2000'lerin başında bir düşüş yaşamamasına rağmen veritabanı uygulamaları büyümeye devam etmiştir. Web tabanlı işletmelerin artması ve kullanımlarının yaygınlaşması ile veritabanı yönetim sistemlerinin önemi giderek artmaya devam etmekte ve yeni uygulamaların geliştirilmesine neden olmaktadır. Günümüzde, veritabanı yönetim sistemi yazılımlarında lider kabul edilen üç işletme Microsoft, IBM ve Oracle olarak sıralanmaktadır.

VERİTABANINA İLİŞKİN TEMEL KAVRAMLAR

Günümüzde sıkılıkla karşılaşılan bir kavram hâline gelen **veritabanı** çok genel anlamda, bir kurumun ihtiyaç duyduğu ve kullandığı veriler bütününe ifade eder. Bu noktada öncelikle *veri* kavramı ve veri kavramı ile karıştırılan *bilgi* kavramı üzerinde durmak faydalı olacaktır.

Veri ham gözlemler, işlenmemiş gerçekler ya da izlenimlerdir. Bu gözlemler, gerçekler ya da izlenimler harf, rakam ya da çeşitli simbol ve işaretler yardımıyla temsil edilir. Bir-birleriyle ilişkilendirilip yorumlanmadıkları sürece tek başlarına bir anlam ifade etmezler ve bu hâllerde karar verme konusunda da karar vericilere bir katkı sağlayamazlar.

Bilgi ise yalnız tanımıyla verinin işlenmiş ve karar verme sürecine destek olacak duruma dönüştürülmüş biçimidir. Sözkonusu işleme ve dönüştürme süreci; veri üzerinde kaydetme, sınıflama, sıralama, hesaplama, özetleme, çoğaltma, analiz ve raporlama işlemlerinin uygulanması ile gerçekleştirilir. Bu işlemler sonucunda veri anlam kazanarak bilgiye dönüşmüş olur.

Veritabanı, (database) herhangi bir konuda birbirile ilişkili olan ve amaca uygun olarak düzenlenmiş, mantıksal ve fizikselleşmiş olarak tanımlanmış veriler bütündür. Bununla birlikte her düzenli veri topluluğunu veritabanı olarak tanımlamak da doğru değildir. Bu nedenle veritabanının özelliklerinin sıralanması faydalı olacaktır.

- Veritabanı herhangi bir kurumda birden fazla uygulamada ortak olarak kullanılabilen verilerden oluşur.
- Veritabanında sürekli niteliği olan veriler bulunur. Buna göre, girdi ya da çıktı ve-risi olan ya da kurum için sürekli bir anlam ifade etmeyen geçici veriler veritaba-nında yer almaz.
- Veritabanı, ortak kullanılan verilerin tekrarlanmasına izin vermeden çok amaçlı kullanılmasına olanak verir.
- Veritabanında saklanan veriler durağan nitelikte değişmez veriler değildir. Ekleme, silme ya da güncelleme işlemleri ile veritabanındaki veriler değiştirilebilir (Yarı-mağan, 2000, s.1).

Veritabanı Yönetim Sistemi (VTYS-Database Management System); veritabanı tanımlamak, veritabanı oluşturmak, veritabanında işlem yapmak, veritabanının farklı kullanıcı yetkilerini belirlemek, veritabanının bakımını ve yedeklemesini yapmak için geliştirilmiş programlar bütündür.

Veritabanı Tanımlamak: Veritabanında yer alacak verinin adı, tipi, uzunluğu gibi veri yapıları ve özelliklerinin belirlenmesidir.

Veritabanı Oluşturmak: Veritabanını yaratma ve veriyi depolama sürecidir. Verilerin depolanacağı fiziksel bellek alanının belirlenmesi ve verilerin bu alana aktarılmasını içerir.

Veritabanında İşlem Yapmak: Veritabanını izleme, veriler üzerinde sorgulama yapma, veritabanında gerekli olan değişiklikleri yaparak veritabanını güncelleme ve verilerden rapor elde etme işlevlerini içerir.

Bu özelliklerin dışında veritabanı, veriyi koruma, veriler arasında ilişki kurma, farklı kullanıcılarla farklı yetki sınırları içinde veriye erişim imkânı sunma işlevlerini de yerine getirir.

Veritabanı ve veritabani yönetim sisteminin birlikte oluşturduğu bütün ise *veritabanı sistemi* olarak ifade edilir.

Geçmişten beri veriye erişim amacıyla farklı yaklaşımalar kullanılmıştır. Bu yaklaşımlardan ikisi; *sıralı erişim* ve *doğrudan erişim* biçimindedir.

Sıralı erişimde, istenilen veriye ulaşılınca kadar ilgili dosyadaki tüm verilerin sırayla okunması gereklidir. Geçmişte kullanılan müzik kasetleri bu tür erişim kullanımına örnektir. Bu kasetlerde şarkı olarak dinlediğimiz müzik verileri sırayla çalınmaktadır. Kasette yer alan 6 numaralı şarkıyı dinleyebilmek için ya ilk 5 şarkıyı dinlemek ya da kasetçaların ileri düğmesine basarak 5 şarkılık verinin fiziksel olarak ileri sarılması gereklidir. Her iki durumda da 6. şarkiya gelebilmek için ilk 5 şarkının fiziksel olarak taranması işlemi yapılmıştır. Bu şekilde, dosyada yer alan tüm verilerin okunması zorunluluğu ve istenilen bilgiye anında ulaşlamaması sıralı erişimin dezavantajı olarak ortaya çıkmaktadır. Bir diğer örnek olarak öğrenci bilgilerinin yer aldığı bir dosyada, dosyanın en sonunda yer alan bir öğrenci bilgisine erişmek için dosyanın tamamının okunması gerekecektir. Buna rağmen sıralı erişimin gerekli olduğu bazı durumlar da bulunmaktadır.

Doğrudan erişimde, sıralı erişimin aksine istenilen veriye ulaşabilmek için o veriye kadar olan diğer tüm verilerin okunması gerekmemektedir. Bu erişim biçiminde adından da anlaşılacağı gibi istenilen veriye doğrudan erişim mümkündür. Bu erişim biçiminde, verilerin yer aldığı fiziksel adresler birer indeks numarası ile tanımlanıp bu indeks numaraları da ayrı bir dosya olarak saklanır. Herhangi bir veriye erişilmek istendiğinde, önce indeks dosyasından verinin yer aldığı adresi gösteren indeks numarası bulunur, daha sonra bu numaraya karşılık gelen fiziksel adrese doğrudan erişim sağlanır. Günümüzde kullanılan müzik CD'lerindeki erişim türü bu erişim türüne örnektir.

SIRA SİZDE

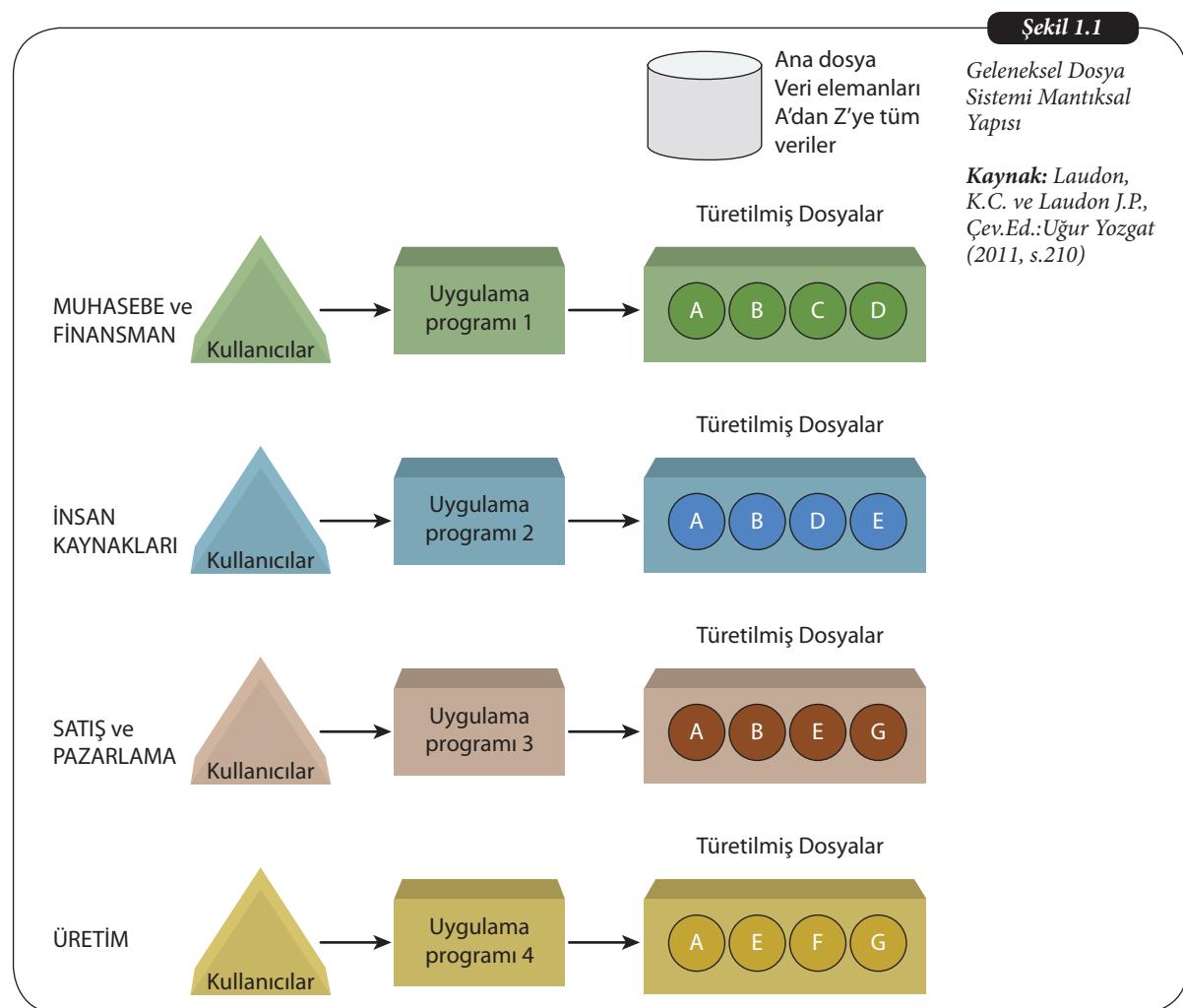


1

Veritabanı kullanımı işletmeler açısından neden önemlidir?

GELENEKSEL DOSYA SİSTEMLERİ VE VERİTABANI YÖNETİM SİSTEMLERİ

Verilerin belirli bir düzen içinde saklanması ve ihtiyaç duyulduğunda erişilebilmesi amacıyla bilgisayar kullanımında geçmişen günümüze farklı yaklaşımalar benimsenmiştir. Veritabanı yaklaşımından önce kullanılan ve *geleneksel dosya sistemi* olarak ifade edilen yaklaşımda veriler bilgisayarda ayrı ayrı dosyalar biçiminde saklanmaktadır. Birbiriyle ilişkili olan veriler bir dosyada, başka bir açıdan birbiriyle ilişkili veriler de başka bir dosyada yer almaktaydı. Bununla birlikte iki farklı dosya içinde aynı verinin yer olması da söz konusu idi.



Şekil 1.1'den de izlenebileceği gibi, geleneksel dosya sisteminde ilgili kuruluşun her alt sistemi kendi faaliyetlerini gerçekleştirmek için o alt sisteme özgü uygulamalar ve veri dosyalarına ihtiyaç duyar. Geleneksel dosya sisteminde, tüm verilerin saklandığı bir ana dosyanın yanı sıra her alt sisteme özgü ve ana dosyanın altkümesi olarak düşünülebilecek birçok alt dosya türetilir. Bu durum da verinin gereksiz yere tekrarına ve dolayısıyla bellek kapasitesinin boş yere kullanılmasına yol açar. Şekil 1.1 incelendiğinde A verisinin, işletmenin dört farklı alt sistemi olarak düşünülen tüm bölgelerinde ihtiyaç duyulan bir veri olduğu ve dolayısıyla bu bölgelerde türetilmiş tüm dosyalarda yer aldığı görülmektedir. Benzer biçimde B verisi üç farklı bölümün dosyalarında yer işgal etmektedir. F verisinin ise yalnızca üretim bölümünün ihtiyaç duyduğu bir veri olduğu ve diğer bölgelerin veri dosyalarında yer almadiği farkedilecektir. Bununla birlikte F verisi dışındaki diğer tüm verilerin tekrarlandığı görülmektedir. Aynı verinin birçok dosya içinde tekrarlanması veri yönetimiyle ilgili pek çok zorluğu da beraberinde getirecektir.

Zaman içinde artan veri miktarı, bu verinin depolanması için gerekli olan kapasite, veriye erişim ve işleme hızında yaşanan sıkıntılar geleneksel dosya sisteminin temel sınırlılıkları olarak ortaya çıkmıştır. Dolayısıyla bu sıkıntılar modern veritabanı sistemlerinin geliştirilmesini de gerekli hâle getirmiştir.

Bilgisayar tabanlı bilgi sistemleri kapsamında kullanılan ve farklı uygulamalar tarafından paylaşılan ortak verilerin düzenlenmesi, saklanması ve kullanılması amacıyla yönelik iki farklı yaklaşım vardır:

- Geleneksel dosya sistemi
- Veritabanı yönetim sistemi

Geleneksel dosya sistemlerinin oluşturduğu olumsuzluklar ve buna karşın veritabanı yönetim sistemlerinin sağladığı yararlar Tablo 1.1'de verilen biçimde özetlenebilir.

Tablo 1.1
Geleneksel Dosya
Sistemleri ile Veritabanı
Yönetim Sistemlerinin
Karşılaştırılması

Geleneksel dosya sistemlerinin sakıncaları	Veritabanı yönetim sistemlerinin üstünlükleri
<ul style="list-style-type: none"> • Veri tekrarı ve veri tutarsızlığına yol açar. • Veri paylaşımına olanak vermez. • Uygulamalarda ihtiyaç duyulan değişikliklerin gerçekleştirilebilmesi için uzmanlık bilgisi gerektirir. • İstenilen veriye ulaşmada güçlükler bulunur. • Verilerin güvenliği ve gizliliği konusunda sorun yaşanır. • Veriler ve uygulamalarla ilgili belirli bir standart yoktur. • Verileri yedekleme ve kurtarma konusunda güçlükler yaşanır. 	<ul style="list-style-type: none"> • Veri tekrarı ve veri tutarsızlığını öner. • Veri paylaşımına olanak verir. • Uzmanlık bilgisine ihtiyaç duyulmayacak derecede kullanım kolaylığı sağlar. • İhtiyaç duyulan veriye, tanımlanmış kullanıcı yetkileri kapsamında kolaylıkla erişilmesini sağlar. • Veri güvenliği ve gizliliğini güçlü bir biçimde yerine getirir. • Veriler ve uygulamalarla ilgili standart yapı ve kuralların olması kullanım kolaylığı sağlar. • Verileri yedekleme ve kurtarma konusunda kolaylık sağlayan programlar barındırır.

VTYS'ler, sağladıkları avantajlara rağmen maliyet açısından geleneksel dosya sistemlerine göre dezavantajlıdır. VTYS için gerekli olan donanım, yazılım ve veritabanı eğitimi için başlangıç yatırımlarının yüksek olması, veri güvenliğini sağlamak, aynı anda gerçekleştirilen işlemlerin kontrolü, veri kurtarma ve bütünlük fonksiyonları için gerekli olan sabit maliyetlerin yüksek olması VTYS'nin dezavantajlı yönlerini oluşturmaktadır.

Bu nedenle değişmesi hiç beklenmeyen basit ve iyi tanımlanmış veritabanı uygulamaları söz konusu olduğunda geleneksel dosya sistemlerini kullanmak daha avantajlı olacaktır.

VERİTABANI YÖNETİM SİSTEMLERİNİN AVANTAJLARI

Tablo 1.1'de verilen özelliklere ilişkin ayrıntılı açıklamalar, veritabanı yönetim sistemlerinin üstünlüklerine vurgu yapacak biçimde izleyen kesimde verilmiştir.

Gereksiz Veri Tekrarı ve Veri Tutarsızlığının Önlenmesi

Önceki kesimde de belirtildiği gibi geleneksel dosya sistemlerinde her alt sistem ya da uygulama için gerekli olan veriler ayrı dosyalar biçiminde düzenlenmektedir. Böylece verilerden bir kısmı bilgisayarda aynı anda birden fazla dosya içinde yer alabilmekte dolayısıyla verinin büyük ölçüde tekrar edilmesi sonucuya karşılaşmaktadır. Bu durum aynı zamanda veri derleme işleminin de tekrarlı olmasına ve veri derleme için yapılan harcamaların artmasına neden olabilmektedir. Diğer yandan veri tekrarı, verilerin depolanması için kullanılan bellek kapasitelerinin dolayısıyla donanım harcamalarının artmasına da yol açar. Veri tekrarı aynı zamanda tekrarlanan verilerin farklı dosyalardaki değerlerinin de farklı olabilmesine ve dolayısıyla veri tutarsızlığı ile uygulamalarda sorunlar yaşanmasına neden olabilir. Örneğin, bir işletmede yer alan ürünlerle ilişkin *stok numarası* bazı dosyalarda *stok kodu* olarak nitelendirilmiş olabilir. Aynı işletmenin stoklar, satışlar ve giyim mağazalarına ilişkin veri dosyalarında aynı ürüne ilişkin ölçü birimi olarak bir dosyada XL kodunun bir başka dosyada 3 rakamı ile belirtilmesi tutarsızlığa bir başka örnektir.

Veritabanı yönetim sistemlerinde ise ilgili kurumda kullanılan bilgi sistemi bir bütin olarak düşünülmektedir. Buna göre farklı alt sistemlerin ihtiyaç duyacağı tüm veriler tek bir merkezde toplanarak ihtiyaç duyan her birimin istediği veriye bu merkezden ulaşması sağlanır. Böylece birden çok uygulamada kullanılan ortak verilerin tekrar edilmesinin önüne geçilir. Veritabanı yönetim sistemi kullanımıyla; veriler arası ilişkileri kurmak için T.C.Kimlik numarası, personel sicil numarası ve stok kodu gibi ayırtedici nitelikte olan veri değerlerinin zorunlu tekrarlanması dışında, veri tekrarı önlenir.

Veri Bütünlüğünün Sağlanması

Veri bütünlüğü, veritabanında yer alan bir verinin farklı uygulamalarda kullanımı söz konusu olduğunda veri üzerinde yapılacak bir değişimin verinin kullanıldığı diğer uygulamalara da yansıtılması anlamına gelir. Bu, özellikle verinin silinmesi durumunda önem kazanmaktadır. Buna göre bir veri silindiğinde verinin ilişkili olduğu tüm uygulamalar dan o verinin silinmesi gereklidir. Geleneksel dosya sistemlerinde veri bütünlüğünün sağlanması güç iken veritabanı yönetim sistemlerinde, sisteme girilen kısıtlamalar ve kurallar yardımıyla veri bütünlüğü rahatlıkla sağlanabilmektedir.

Veri Paylaşımının Sağlanması

Geleneksel dosya sistemlerinde aynı veriye birden fazla kullanıcının aynı anda erişmesine olanak yoktur. Bu sistemlerde veriye erişim sıralı erişim kurallarına göre gerçekleştirilir. Buna göre veriye erişme isteğini ilk bildiren ilk sırada veriye erişir, diğer kullanıcının veriye erişebilmesi için ilk kullanıcının işinin bitmesi gereklidir. Veritabanı yönetim sistemlerinde ise sunucu/istemci mimarisi kullanılarak veriler tek bir merkezden (sunucu bilgiyasar), ağ üzerinde yer alan ve erişim yetkisi olan tüm kullanıcılarla (istemci bilgisayarlar) aynı anda veriye erişme olanağı sunar. Bu durum verinin aynı anda birçok kullanıcı tarafından paylaşılması anlamına gelir. Verilerin ortak paylaşımı veritabanı yönetim sistemi yaklaşımının temel amaçlarından biridir.

Bununla birlikte uygulamaların veritabanında eş zamanlı işlem yapmasının doğruluğu sakıncaların da veritabanı tasarlarken göz önüne alınması gereklidir. Veritabanı yönetim sistemi yaklaşımında birtakım kısıtlama ve kurallarla söz konusu olumsuzluklar için önlem alınabilir.

Bununla ilgili olarak bankacılık sektöründen bir örnek verebiliriz. Bir bankanın veritabanı tanımları arasında “bir banka hesabından, hesaptaki mevcut miktarдан daha fazla para çekilemez, hesap bakiyesi negatif olamaz” biçiminde ifade edilebilecek bir kısıtlamanın bulunduğu varsayılmıştır. Bu bankadaki bir hesapta 200.000 TL bulunduğu ve bu hesaptan para çekmeye yetkili iki kişinin olduğunu düşünelim. Parayı çekmeye yetkili kişilerin her ikisi de aynı anda farklı şubelerden 100.000 ve 150.000 TL çekmeye çalışın. Her iki işlem için hesaptaki mevcut miktarın hesapta 200.000 TL bulunduğu ve bu miktarın çekilmesi istenen miktarдан büyük olduğu görülmüştür. Eğer hiçbir önlem alınmadan her iki işlem de sürdürülürse hesap miktarı 250.000 TL azalarak hesap bakiyesi -50.000 TL ye ulaşır. Bu olumsuz durum veritabanı üzerinde tanımlanan bir kısıtlamaya önlenir. Örneğin de görüldüğü gibi veritabanı yönetim sistemi, verilerin aynı anda paylaşımına izin verirken, ortak paylaşımından doğabilecek olumsuz durumları da önleyebilecektir.

Kullanımda Üst Düzey Uzmanlık Gerektirmemesi

Günümüzde kurumlar, bilgi sistemlerinden yoğun bir biçimde yararlanmaktadır ve bu bilgi sistemleri kapsamında birbirile ilişkili çok çeşitli ve büyük miktarda veriler üzerinde her türlü güncelleme, sorgulama, raporlama vb. işlemler yapmak istemektedir. Geleneksel dosya sistemlerinde veriler dosyalar biçiminde saklanırken kurumun ihtiyaç duyduğu

uygulamaları gerçekleştirmek üzere de ayrıca geliştirilen uygulama programlarından yararlanılmaktadır. Verilerin karmaşıklığı, veriler arasında çeşitli düzeylerde birçok ilişki-lendirmenin yapılması gerekliliği ve uygulamaların çeşitliliği nedeniyle, kullanılan dosya yapıları, dosyalar arası ilişkileri kurmak için kullanılan teknikler ve dosyalara erişim için kullanılan yaklaşımlar oldukça karmaşık bir yapı sergilemektedir. Bu nedenle sözü edilen uygulama programlarının geliştirilmesi için bilişim teknolojisinde uzman kişilere ihtiyaç duyulmaktadır. Çeşitli verilere ulaşma ihtiyaçında olan çok sayıda kullanıcı kesiminin bilgi sistemi ile etkileşimi de yalnızca sözkonusu bu uygulama programları ile sağlanabile-mektedir. Bu durum sistemin kullanımını öncede sınırlandırır. Çünkü uygulama programları yalnızca programın geliştirilmesi sırasında tanımlanmış olan genel ihtiyaçla-rı karşılayabilir. Özellikle yönetici konumundaki kullanıcılar önceden belirlenmemiş çok seyrek ya da yalnız bir kez oluşan bilgi sistemi ihtiyaçlarını bu biçimde tanımlanmış uygulama programlarıyla karşılamakta zorlanırlar. Rutin olmayan bu tür ihtiyaçları karşılaya-cak programlar yine yalnızca bilişim teknolojisinde uzman olan kişilerce, uzun bir zaman içinde hazırlanabilir ve kullanıcıların bu kadar uzun süre beklemesi mümkün değildir. Bunun bir sonucu olarak da bu tür ihtiyaçlar zamanında karşılanamaz. Sonuç olarak bilgi sisteminin kullanımı sınırlı kalır. Bu noktada VTSY'lerin bu olumsuzlukları gideren iki yönü bulunmaktadır. Bunlardan birincisi, VTSY'lerin kullanıcıları verilerin saklanması için bilgisayar belleklerinde oluşturulan karmaşık fiziksel yapılarla ilgilenmek zorunda bırakmayıp, yalnızca ilgilendikleri verileri içeren basit mantıksal yapıları görmesine olanak sağladasıdır. İkinci olarak da VTSY'lerin kullanıcılarına veritabanı uygulamalarını kolay-likla gerçekleştirebilmeleri için sorgu dili gibi kullanımı basit olanaklar sunmasıdır. Bunun sonucunda önceden tanımlanmamış veri ihtiyaçlarını karşılamak için bilişim uzmanına olan gereksinim ortadan kalkmış ve ilgili kullanıcılar, sorgu dillerini kullanmak yoluyla veri ihtiyaçlarını karşılayabilir duruma gelmiştir. Bu da kullanımın önceden belirlenmiş operasyonel uygulamalarla sınırlı kalmaması ve yaygınlaşması sonucunu getirmiştir.

Verilerin Gizliliğinin ve Güvenliğinin Sağlanması

Herhangi bir kurumda, kurum çalışanları tarafından ortak kullanılan verilerin depolandı-ğı yapının (dosya ya da veritabanı) ve bu yapı içinde yer alan verilerin gizliliği ve güvenliği çok önemli bir konudur. Veriler üzerinde okuma, yazma, ekleme, silme ve güncelleme gibi faaliyetler gerçekleştirilebilir. Ancak her veri üzerinde her kullanıcının bu faaliyetle-rin tamamını gerçekleştirebilmesi, istenen bir durum değildir. Örneğin bir okulda öğrenci-lere ilişkin verilerle ilgilenen farklı kullanıcı grupları bulunur. Öğrencilerin kendileri, ders aldıkları öğretim üyeleri, idari personel, test biçimindeki sınav kâğıtlarının bilgi işlem teknolojileriyle okunmasını sağlayan teknik personel, sınav tarihleri ve sınav salonlarını planlayan personel, öğrenci işleri daire başkanlığı vb. bu kullanıcılara örnek olarak ve-rilebilir. Bu kullanıcılardan bir kısmı veriler üzerinde yalnızca okuma işlemi gerçekleştirebilirken, bir kısmı hem okuma hem de yazma, diğer bir kısmı ise okuma, yazma ve güncelleme işlemlerini yapabilir. Örneğin öğrenciler sınavlardan aldıkları notları yalnızca görebilir (okuma faaliyeti) ama değiştiremezler (yazma faaliyeti). Öğretim üyesi öğrenci notlarını sisteme girebilir (yazma faaliyeti) ve onayladıktan sonra da görüntüleyebilir (okuma faaliyeti). Bununla birlikte onaylama işleminden sonra değişiklik yapamaz. Sis-teme yanlış girilen bir notun düzeltmesi ise bir dizi dilekçe ve alınan kararlar sonucu (öğrenci/öğretim üyesinin yazılı dilekçesi üzerine, yapılan incelemeler sonrasında alınan yönetim kurulu kararı ile) öğrenci işleri daire başkanlığındaki ilgili personel tarafından gerçekleştirilebilir. Öğrenci bilgi sistemi için örneklenen bu durum çeşitli kurumlardaki veriler için de örneklenebilir.

Verilere ve verilerin saklandığı yapılara erişim ve erişim sonrası çeşitli düzeylerde kulanma yetkisinin tanımlanması verilerin gizliliği ve güvenliğini sağlar. Bu anlamda geleneksel dosya sistemlerinde gizlilik ve güvenliğin sağlanması pek kolay değildir.

Veritabanı yönetim sistemlerinde, veritabanı tasarılanır ve oluşturulurken veritabanı kullanıcılarının kimler olacağı ve hangi yetkilere sahip olacağı da belirlenir. Kullanıcı grupları ve yetkileri ile ilgili tanımlar veritabanındaki diğer tanımlarla birlikte saklanır. Herhangi bir kullanıcı veritabanı üzerinde işlem yapmak istediginde, veritabanı öncelikle kullanıcının yapmak istediği işleme ilişkin yetkisinin olup olmadığını kontrol eder. Kullanıcı yetkisiz olduğu bir işlem yapmak istiyorsa veritabanı yönetim sistemi bu işlemin yapılmasına izin vermez. Veritabanı yönetim sistemlerinin bu özelliği verilerin yetkisiz kullanıcılar tarafından görülmemesini engelleyerek verinin gizliliğini, verinin yetkisiz kullanıcılar tarafından değiştirilmesini önleyerek de verinin güvenliğini sağlamış olur.

Standart Yapı ve Kuralların Uygulanabilir Olması

Geleneksel dosya sistemlerinde her alt sistem kendi faaliyetlerine özgü uygulamaları ve her uygulama da kendi dosya yapısını kullandığı için dosya yapılanmalarında belirli bir standart olmayıp farklılıklar bulunmaktadır.

VTSY'lerde ise merkezi bir kontrol sistemi bulunur. VTSY'lerin yapısal özellikleri ve veritabanı sorumlusunun varlığı veriler ve veritabanı üzerinde belirli standartların oluşturulması ve uygulanabilmesini olanaklı kılar. Bu standartlar verinin yapısı, gösterim biçimi, adlandırılması, belgelenmesi ile ilgili yapısal standartlar olabileceği gibi kurum içi, kurumlar arası, ulusal ya da uluslararası düzeyde belirlenmiş kurallar biçiminde de olabilir. Standartların varlığı, veritabanı yapısını anlama ve kullanma açılarından büyük kolaylık sağlamanın yanısıra farklı sistemler arasında veri alışverişi için de çok önemli bir ihtiyaçtır.

Veritabanı yönetim sistemlerinin geleneksel dosya sistemlerine göre üstünlükleri nelerdir?



SIRA SİZDE
2

VERİTABANI KULLANICILARI

Veritabanı ile herhangi bir şekilde etkileşimde olan kişi ya da kişiler veritabanı kullanıcıları olup aşağıdaki gibi sınıflandırılabilirler:

- Veritabanı Sorumluları
 - Veritabanı Yöneticisi
 - Veritabanı Tasarımcısı
- Son Kullanıcılar
 - Standart Kullanıcılar
 - Sıradan ya da Parametrik Kullanıcılar
 - Gelişmiş Kullanıcılar
 - Bağımsız Kullanıcılar
- Sistem Analistleri ve Uygulama Programcları

Veritabanı Sorumluları

Veritabanı sorumluları, veritabanının tasarılanması, oluşturulması ve veritabanının işletim faaliyetlerinden birinci derecede sorumlu olan ve veritabanı üzerinde en fazla yetkiye sahip olan kullanıcılardır. Veritabanı sorumluları, *veritabanı yöneticisi* ve *veritabanı tasarımcısı* olarak iki başlık altında incelenebilir. Her iki sorumluluğu aynı kişi ya da kişiler alabileceği gibi veritabanını kullanacak olan kurumun/veritabanının büyülüğu ve kullanıcı sayısı gibi faktörlere bağlı olarak veritabanı yöneticisi ile veritabanı tasarımcısının farklı kişiler olması da mümkündür.

Veritabanı Yöneticisi

Veritabanı yöneticisinin (database administrator) veritabanına erişim yetkilerini belirleme, veritabanı kullanımının düzenlenmesi ve izlenmesini sağlama, ihtiyaç duyulan yazılım ve donanım kaynaklarını edinme biçiminde sıralanan sorumlulukları vardır. Ayrıca güvenlik ihlalleri ve kötü sistem yanıt süresi gibi sorunların çözümünden de sorumludur. Büyük işletmelerde bu sorumluluklar için yardımcı personele de ihtiyaç duyulur.

Veritabanı Tasarımcısı

Veritabanı tasarımcısı (database designer) veritabanında saklanacak olan verilerin tanımlanmasından ve bu verilerin depolanması ve gösterilmesi için gerekli olan uygun yapıların seçilmesinden sorumludur. Bu görevler çoğunlukla verilerin veritabanına depolanmasından ve veritabanı uygulamalarından önce yerine getirilir. Veritabanı tasarımcısı, muhtemel veritabanı kullanıcılarının ihtiyaçlarını anlamak ve onların bu ihtiyaçlarını karşılayabilecek özellikle bir tasarıyı oluşturmak amacıyla öncelikle sözkonusu veritabanı kullanıcıları ile iletişimde geçmelidir. Pek çok uygulamada, veritabanı tasarımcısı veritabanı yöneticisinin yardımcı personeli olup veritabanı tasarıyı tamamlandıktan sonra bu personele başka sorumluluklar atanmaktadır. Veritabanı tasarımcısı tasarım boyunca potansiyel kullanıcı gruplarıyla karşılıklı etkileşim hâlinde olup bu grupların veriye erişimini ve veri üzerinde işlem yapabilmelerini olanaklı kılan kullanıcı görünümlerini geliştirirler. Her kullanıcı grubuna ilişkin görünüm, analiz edilerek diğer kullanıcı gruplarına ilişkin görünümlerle bütünleştirilmeleri sağlanır. Veritabanı tasarıminın tüm kullanıcı gruplarının ihtiyaçlarını destekleyecek kapasitede olması gereklidir.

Veritabanı yönetici ya da veritabanı tasarımcısı biçiminde ayrılmaksızın, veritabanı sorumlularının yerine getirdikleri temel görevler aşağıdaki gibi özetlenebilir:

- *Veritabanı tasarımini yapma:* Veritabanının farklı düzey şemalarının oluşturulması, veriler üzerinde yapılacak her türlü işlem için gereksinimlerin belirlenmesi, veritabanı içeriğinin oluşturulması.
- *Bütünlük kısıtlamalarını belirleyip tanımlama:* Veritabanında veri bütünlüğünün sağlanabilmesi, veri kaybının önüne geçilebilmesi, veri bütünlüğünü tehlkeye sokacak kullanıcı hatalarının önlenmesi amacıyla gerekli kurallar, ilişkiler ve kısıtlamaların belirlenmesi.
- *Veritabanı kullanım yetkilerini tanımlama:* Veritabanı kullanıcılarının ve kullanım yetkilerinin tanımlanması ile her kullanıcı grubunun hangi veriler üzerinde hangi işlemleri yapmaya yetkili olduğunu belirlenmesi. Bu tanımlamalar sonrasında ilgili kullanıcılarla yetkili oldukları erişimin sağlanması.
- *Veritabanı güvenliğini sağlama:* Kullanıcıdan ya da yazılım sorunlarından kaynaklanabilecek veri kaybının önlenmesi amacıyla veritabanının yedeklerinin alınması ve kurtarma işlemlerine ilişkin düzenlemelerin yapılması.
- *Veritabanının işletimini izleme ve sürekliliğini sağlama:* Veritabanın kullanıma sürekli açık olmasının sağlanması ve sistemde meydana gelebilecek herhangi bir sorunu hızlı biçimde giderebilecek tedbirlerin alınması.
- *Güncelleme ihtiyaçlarına cevap verebilme:* Kullanıcı ihtiyaçlarında ortaya çıkabilecek değişikliklerin izlenebilmesi ve bu değişikliklere paralel olarak veritabanı içeriği, şema tanımları, bütünlük kısıtlamaları, fiziksel yapı ile ilgili parametreler, kullanıcı tanımları ve kullanıcı yetkilerinde gerekli değişikliklerin oluşturulması ve tanımlanması.
- *Veritabanından beklenen performansı sağlama:* Veritabanı kullanıcılarının bekletterine cevap verecek bir yapının donanım ve yazılım ihtiyaçlarının sağlanması, veritabanındaki tanımlamaların, kısıtlamaların kullanıcı ihtiyaçlarına cevap verebilecek yeterlilikte olup olmadığıının izlenmesi, gerekiyorsa değişikliklerin yapılması.

Son Kullanıcılar

Son kullanıcılar (end users), yaptıkları işler gereği veritabanına sorgulama ya da güncelleme yapmak veya rapor türemek için erişen kullanıcılardır. Bu tür kullanıcılar veritabanı ile kullanıcının bağlantısını sağlayan ve uygulama programcıları tarafından geliştirilen yazılımları kullanırlar. Son kullanıcılar da kendi içinde grupperlendirilebilir:

Standart son kullanıcılar: Veritabanına nadiren erişim yapan fakat her seferinde farklı bilgi ihtiyacı olabilen kullanıcılardır. Bu tür kullanıcılar isteklerini belirtmek için gelişmiş veritabanı sorgu dili kullanırlar. Orta ya da üst düzey yöneticiler bu gruba örnek verilebilir.

Sıradan ya da parametrik son kullanıcılar: Son kullanıcıların önemli bir bölümünü bu tür kullanıcılar oluşturur. Bu kullanıcıların temel iş fonksiyonları, veritabanı üzerinde sürekli bir sorgulama ve güncelleme yapmalarını gerektirir. Standart (önceden belirlenmiş) sorgu ve güncelleme yaparlar. Bu gruba giren kullanıcılar çok çeşitli konum ve görevlerde olabilirler. Örneğin;

- Banka memurunun, mevduat hesaplarına yatırılan ya da çekilen miktarları veya hesap bakiyelerini kontrol etmesi
- Havaalanları ve oteller için rezervasyon yapan acentaların ya da araç kiralama şirketlerinin, gelen müşteri taleplerini cevaplayabilmek amacıyla sistemlerinde uygun yer ya da araç olup olmadığını sorgulamaları ve buna göre rezervasyon yapmaları sıradan son kullanıcılar ve veritabanını kullanma biçimlerine verilebilecek örnekler arasındadır.

Gelişmiş son kullanıcılar: VTYS'nin sağladığı özellikler yardımıyla ayrıntılı olarak belirledikleri karmaşık gereksinimlerini karşılamak amacıyla veritabanını kullanan gruptur. Mühendisler, bilim adamları, işletme analistleri vb. bu gruptaki kullanıcılarla örnek olarak verilebilir.

Bağımsız son kullanıcılar: Bu kullanıcılar menü kullanımını ya da araç çubukları gibi grafiksel öğeler yardımıyla kullanım kolaylığı sağlayan hazır paket programlarını kullanarak kişisel veritabanlarının sürekliliğini sağlar. Vergilere ilişkin olarak hazırlanan ve kişisel finansal verileri depolayan vergi paketi kullanıcıları bu tür kullanıcılarla verilebilecek bir örnektir.

Tipik bir VTYS, veritabanına erişim için farklı kullanıcı gruplarının ihtiyaç duyduğu birçok özelliği sağlar. Her kullanıcı grubu bu özelliklerden ihtiyaç duyduğu kadarıyla yararlanır. Örneğin, sıradan son kullanıcıların VTYS tarafından sağlanan özellikler hakkında bilgi sahibi olma ihtiyacı çok az olacaktır. Bu tür kullanıcıların, veritabanına erişimi sağlayan kullanıcı ara yüzlerini kullanmayı ve standart işlemlere ilişkin uygulamaları bilmeleri yeterli olacaktır. Standart son kullanıcılar da VTYS özellikleri içinden yaptıkları işin gereği olarak, yalnızca sürekli tekrarlayacakları özellikleri öğrenirler. Gelişmiş son kullanıcılar ise karmaşık gereksinimlerini karşılayabilmek için VTYS'nin pek çok özelliğini öğrenmeye çalışırlar. Bağımsız son kullanıcılar da veritabanı ihtiyaçlarını karşılamak üzere belirli bir yazılım paketini kullanma konusunda ustalarıslar.

Sistem Analistleri ve Uygulama Programcıları

Sistem analisti son kullanıcıların, özellikle de sıradan son kullanıcıların gereksinimleri ni belirleyen ve standart işlemler yoluyla bu gereksinimleri karşılayabilecek ayrıntıları belirleyen kişi ya da kişilerdir. Uygulama programcıları ise sistem analisti tarafından belirlenen ayrıntıları program hâline getiren ve daha sonra test eden, hataları ayıplayan, belgeleyen ve kaydedilmiş işlemler olarak sürekliliğini sağlayan kişilerdir. Yaygın olarak yazılım geliştiriciler ya da yazılım mühendisleri olarak da anılan analistler ve programcıların yukarıda sıralanan görevlerini yerine getirebilmeleri için VTYS'nin sağladığı tüm özellikleri bilmeleri gereklidir.

SIRA SIZDE



3

Veritabanı yöneticisi ile veritabanı tasarımcısı arasındaki fark nedir?

VERİTABANI YÖNETİM SİSTEMLERİNİN MİMARİSİ

VTYS'lerinin mimarisi geçmişten günümüze incelendiğinde, ilk veritabanı sistemlerinde, VTYS'nin tüm yazılım paketlerinin tek bir sisteme entegre edildiği, modern VTYS'lerinde ise istemci/sunucu mimarisi ile modüler bir yapılanmanın söz konusu olduğu görülmektedir. Buna bağlı olarak, büyük merkezi ana bilgisayarlar yerini, iletişim ağları aracılığıyla çeşitli sunucu bilgisayarlara bağlanan yüzlerce iş istasyonu ve kişisel bilgisayarlara bırakmıştır. Burada sözü edilen sunucu bilgisayarlar; web sunucular, veritabanı sunucuları, dosya sunucuları, uygulama sunucuları vb. olarak örneklenebilir.

Basit bir istemci/sunucu VTYS mimarisinde sistem fonksiyonel olarak iki module ayrılmaktadır. İstemci modülü (client modül), VTYS'nin herhangi bir kullanıcı iş istasyonunda ya da kişisel bilgisayar üzerinde çalışan parçasıdır. Tipik olarak istemci modülünde veritabanına erişmek için uygulama programları ve kullanıcı arayüzleri bulunur. Sunucu modülünde (server module) ise veri deposu ile veri deposuna erişimi ve sorgulamayı sağlayacak fonksiyonlar yer alır.

Veri Modelleri

Veritabanı yaklaşımının temel karakteristiklerinden biri, veritabanının bazı veri soyutlama düzeyleri sağlamasıdır. **Veri soyutlama** (data abstraction), verilerin düzenlenmesi ve depolanmasına ilişkin ayrıntıların gizlenmesi ve verinin daha iyi anlaşılmasını sağlamak için veriye ilişkin temel özelliklerin vurgulanması anlamına gelir. Veri modeli ise söz konusu bu soyutlamaları gerçekleştirebilmek için gerekli olan araçları sağlar.

Veri modeli (data model), bir veritabanının mantıksal yapısını tanımlamada kullanılacak kavramlar, işlemler ve kurallar bütünüdür. Veritabanının mantıksal yapısı; veri tipleri, veriler arasındaki ilişkiler, veri üzerinde uygulanacak kısıtlamalar vb. dir. Veri modellerinin çoğu verinin geri çağrılması ve veritabanı üzerinde güncelleme yapmaya yönelik işlemleri de içerir. Veri modelinin bu temel işlevlerine ek olarak veritabanının dinamik hareketlerini belirleyen kavramları içermesi de yaygınlaşmıştır. Bu özellik veritabanı tasarımcısına veritabanı üzerinde, geçerli kullanıcı tanımlı işlemleri yapmasına olanak sağlar.

Veri Modellerinin Sınıflandırılması

Yüksek düzeyli ya da kavramsal veri modelleri (conceptual data models), kullanıcıların *veri algılama biçiminde* ilişkili kavramları kapsar.

Düşük düzeyli ya da **fiziksel veri modelleri** (physical data models), *verinin bilgisayar ortamında nasıl depolanacağına* ilişkin ayrıntıları tanımlayan kavramları kapsar. Fiziksel veri modelleri kavramları genellikle son kullanıcılar için değil, bilgisayar uzmanları için geliştirilir.

Bu iki uç model arasında kalan model sınıfı ise *temsili modeller* ya da *uygulama veri modelleri* (implementation data model) olarak adlandırılır. Bu model hem son kullanıcılar tarafından kolaylıkla anlaşılabilen kavramları hem de verinin bilgisayarda depolanması yöntemlerine ilişkin kavramları içerir.

Kavramsal veri modelleri; varlıklar, öznitelikler, ilişkiler gibi kavramları kullanır. Bu kapsamında varlık-ilişki modelleme konusu 2.Ünite'de ele alınmıştır. Uygulama veri modelleri geleneksel ticari VTYS'lerinde en sık kullanılan modellerdir. Bu modeller geçmişte yaygın olarak kullanılan ağ ve hiyerarşik veri modeli ile günümüzde yaygın olarak kullanılan ilişkisel modelleri içerir. Uygulama veri modelleri, verileri kayıt yapılarıyla gösterdiklerinden bazen kayıt tabanlı veri modelleri olarak da adlandırılırlar.

Veritabanı yaklaşımı **veri soyutlama** özelliği ile farklı kullanıcıların verileri tercih ettikleri ayrıntı seviyesinde algılamalarına, gereksiz ayrıntıları görmemelerine olanak sağlar.

Temel **veri modeli** işlemleri; veritabanı üzerinde ekleme, silme, değiştirme, veriyi geri çağrıma gibi genel işlemleri içerir.

Fiziksel veri modelleri, kayıt biçimini, kayıt sırası ve erişim yolu bilgilerini göstermek suretiyle verilerin bilgisayarda dosya olarak nasıl saklandığını belirler.

Nesneye yönelik modeller kavramsal veri modeliyle yakın ilişkili olan yüksek düzeyli uygulama modellerinin yeni üyesi olarak düşünülebilirler. Nesneye yönelik veri modelleri için bir standart ODGM (Object Data Management Group-Nesneye Yönelik Veri Yönetim Grubu) olarak adlandırılır. Nesneye yönelik veri modelleri aynı zamanda (özellikle yazılım mühendisliği alanında) yüksek seviyeli kavramsal model olarak da kullanılmaktadır.

Şemalar, Örnekler ve Veritabanının Durumu

Veritabanının herhangi bir veri modeliyle tanımlanması *veritabanı şeması* (database schema) olarak adlandırılır. Sözkonusu bu şema veritabanının tasarlanması sürecinde oluşturulur ve sık sık değişiklik göstermez. Veri modellerinin çoğu şemaların bir diyagram olarak gösterilebilmesi amacıyla kullanılan belirli kuralları vardır. Şema görünümleri *şema diyagramı* (schema diagram) olarak adlandırılır. Şema diyagramları şemaların yalnızca bazı yönlerini gösterir. Diğer yönler diyagramdan belirlenemez. Örneğin, her bir veri kalemne ilişkin veri tipi, çeşitli dosyalar arasındaki ilişkiler ya da verilere ilişkin kısıtlar diyagramdan izlenemez. Veritabanındaki mevcut veriler oldukça sık değişimlidir. Örneğin, bir öğrenci veritabanında her yeni öğrenci kaydında veritabanı değişecektir. Bir veritabanının herhangi bir andaki durumuna *anlık görüntü* denir. Bu aynı zamanda veritabanının mevcut *örnek* (instance) kümesi olarak adlandırılır. Veritabanında her şema yapısı kendi oluşum kümesine sahiptir. Örneğin “*öğrenci*” veritabanındaki bireysel öğrenci kayıtları oluşumlara örnektir. Veritabanı durumu, belirli bir şema ile ilişkilendirilerek yapılandırılır. Her zaman yeni bir kayıt ekleme, silme ya da bir kayıttaki verinin değerini değiştirmek mümkündür (böylece veritabanının durumu bir durumdan diğerine değiştirilebilir).

VTYS şema yapılarını ve kısıtlarını (meta veri) veritabanı içinde tanımlar ve VTYS yazılımı ne zaman ihtiyaç duysa bu şemaya başvurur.

Üç Şema Mimarisi

Veritabanı yaklaşımının önemli karakteristikleri aşağıdaki biçimde sıralanabilir:

1. Veritabanı tanımlarını (şema) depolamak için kendi kendine tanım yapabilmeyi sağlayan bir katalog kullanır.
2. Veri ve program izolasyonunu sağlar (program – veri bağımsızlığı; program – işlem bağımsızlığı)
3. Çoklu kullanıcıyı destekler

Üç şema mimarisi, yukarıda sıralanan bu özelliklerin gerçekleştirilemesine ve anlaşılımasına yardımcı olacaktır.

Üç şema mimarisinin amacı kullanıcı uygulamalarını fiziksel veritabanından ayırt etmektedir. Bu mimaride şemalar izleyen üç düzeyde tanımlanır:

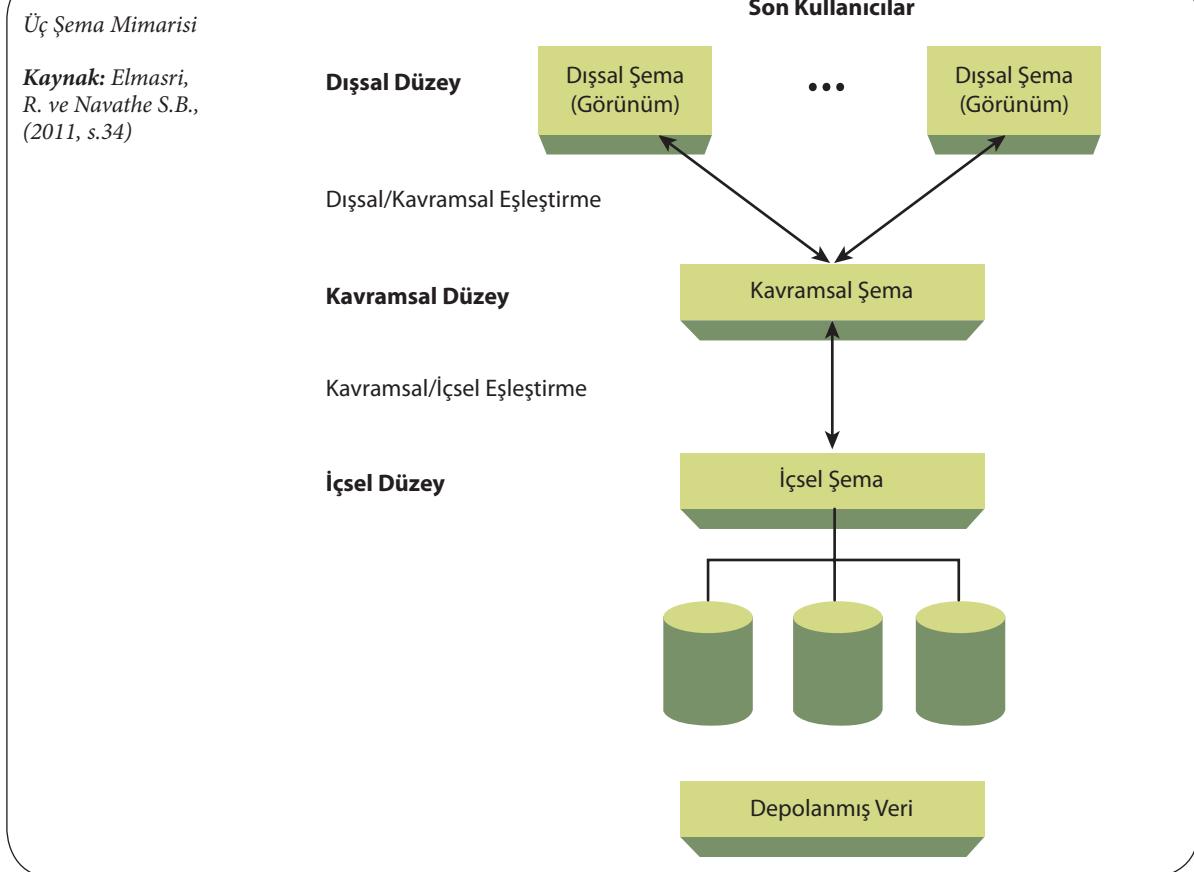
1. **İçsel (fiziksel) düzey** (internal level), veritabanının *fiziksel depolama yapısını* tanımlayan içsel şemayı içerir. İçsel şema, veriyi depolama ayrıntılarının tamamını ve veritabanına erişim yollarını tanımlayan fiziksel veri modelini kullanır.
2. **Kavramsal düzey** (conceptual level), kullanıcı topluluğu için tüm veritabanının yapısını tanımlayan kavramsal şemayı içerir. Kavramsal şema fiziksel depolama yapısının ayrıntılarını gizler ve veritabanında yer alan verilerin tipine, veriler arası ilişkilere, kullanıcı işlemlerine ve kısıtlara ilişkin tanımlara yoğunlaşır. Veritabanı sistemlerinde, uygulama veri modelleri kavramsal şemayı tanımlamak için kullanılır. Bu kavramsal şema, yüksek düzeyli kavramsal veri modelleri kullanılarak tasarlanır.
3. **Dışsal (görünüm) düzey** (external level), bir dizi dışsal şema ya da kullanıcı görünümü içerir. Her dışsal şema bir grup kullanıcının ilgilendiği bazı veritabanı bölümlerini tanımlar. Böylece veritabanının diğer kısmı bu kullanıcı grubundan gizlenir. Her dışsal şema, bir yüksek düzey veri modelinde tasarlanan dışsal şema tabanlı uygulama veri modeli kullanır.

Çoğu VTYS tam ve açık olarak bu üç düzeye ayrılmaz fakat bu üçlü şema mimarisini bir ölçüde destekler. Örneğin bazı eski VTYS'leri fiziksel düzey ayrıntılarını ayrı bir düzey olarak ele almayıp kavramsal şema içinde barındırmaktaydı.

Burada dikkat edilmesi gereken nokta, bu üç şemanın verilerin yalnızca tanımı olduğunu. Depolanmış gerçek veri yalnızca içsel (fiziksel) düzeydedir.

Şekil 1.2'den de izlenebileceği gibi üç şema mimarisine dayanan VTYS'lerinde her kullanıcı grubu kendi dışsal (görünüm) şemalarına başvurur. Bu nedenle VTYS, dış şemada yapılan belirli bir isteği kavramsal şema isteğine daha sonra da depolanmış veritabanı üzerinde işlem yapmak üzere içsel şema isteğine dönüştürmek zorundadır. Yapılan istek veritabanından bilgi çağrılmak ise veritabanından çıkarılan veri, kullanıcının dışsal görünümüyle eşleşecek biçimde dönüştürülür. Şema düzeyleri arasında bu şekilde gerçekleştirilen dönüşümler *eşleştirme* (mapping) olarak adlandırılır. Bu şekildeki eşleştirmeler zaman alıcı olabilir bu nedenle bazı VTYS'leri dışsal görünümleri desteklemezler. Buna rağmen bu sistemlerde bile kavramsal ve içsel düzeyler arasında belirli miktarda dönüşüm gerekli olacaktır.

Şekil 1.2



Veri Bağımsızlığı

Veri bağımsızlığı, herhangi bir düzeydeki şema değiştiğinde bir üst düzeydeki şemanın değişmeden kalmasını bununla birlikte iki düzey arasındaki eşleştirmelerin değişmesini sağlar. Bu eşleştirme sonucunda bir üst düzeydeki şemaya başvuran uygulama programları değişiklik ihtiyacı göstermez.

Şema düzeyleri arasındaki dönüştürme kapasitesi olarak tanımlanabilen veri bağımsızlığı (data independence) yine üç şema mimarisi ile açıklanabilir. Veri bağımsızlığı iki başlıktta ele alınır:

- Mantıksal veri bağımsızlığı** (logical data independence), kavramsal şemanın dışsal şemalarda ya da uygulama programlarında değişiklik yapılmaksızın değiştirilebilmesi anlamına gelir. Kavramsal şemada değişiklik, veritabanını genişletmek (bir kayıt tipi ya da veri kalemi eklemek) kısıtları değiştirmek ya da veritabanını eksiltmek (bir kayıt ya da veriyi silmek) biçiminde olabilir.

- 2. Fiziksel veri bağımsızlığı** (physical data independence), kavramsal şemada bir değişiklik yapılmaksızın içsel şemada değişiklik yapma kapasitesidir. Buna bağlı olarak içsel şemada yapılan değişiklik dışsal şemalarda da bir değişiklik yapılmasını gerektirmez. İçsel şemalarda değişiklik ihtiyacı fiziksel dosyalar üzerinde yeniden düzenleme ihtiyacı ile ortaya çıkabilir. Veritabanına ek erişim yapısının oluşturulması, veritabanının bilgisi çağrıma performansının geliştirilmesi ya da veritabanının güncellenmesi içsel şema değişikliklerine verilebilecek örneklerdir.

Fiziksel veri bağımsızlığı, fiziksel ayrıntıların olduğu çoğu veritabanı ve dosya ortamında mevcuttur. Bununla birlikte kavramsal veri bağımsızlığına erişmek daha zordur çünkü kavramsal veri bağımsızlığı uygulama programlarını etkilemeden yapısal değişiklikler ve kısıtlama değişikliklerine izin verir. Çok düzeyli VTSY'leri söz konusu olduğunda bu VTSY'lerin, farklı düzeyler arasındaki veri ve istek eşleştirmelerini haritalayan bir katalogu da (fihrist) barındırması gereklidir. VTSY bu amaçla ek bir yazılım kullanır.

Veritabanı Yönetim Sistemlerinde Kullanılan Diller

VTSY daha önce de belirtildiği gibi farklı özellikteki kullanıcılarla destek verir. Bu nedenle VTSY'lerin bu farklı kullanıcı gruplarından her birine yönelik uygun dil ve arayüz kullanmaları gereklidir.

Veritabanı tasarımları tamamlandıktan ve veritabanını uygulayacak bir VTSY seçildikten sonra ilk adım veritabanı için kavramsal ve fiziksel şemaların belirlenmesi ve bu iki düzey arasında eşleştirmelerin yapılmasıdır. Düzeyler arasında mutlak bir ayırımın yapılmadığı pek çok VTSY'de iki şemayı tanımlamak amacıyla **veri tanımlama dili** (data definition language) olarak adlandırılan bir dil, veritabanı yöneticisi ve veritabanı tasarımcısı tarafından kullanılır. İki şema bu şekilde tanımlandıktan sonra sözkonusu tanımlar, VTSY'de **veri tanımlama dili derleyicisi** (data definition language compiler) tarafından işlenerek VTSY katalogunda depolanacak uygun yapılar biçimine dönüştürülür.

VTSY veritabanı tanımlarının, VTSY tarafından derlenerek saklanması veritabanı yaklaşımının temel özelliklerinden biridir. Bu süreç sayesinde, veritabanı tanımlarının yetkili kişiler tarafından bir kez yapılması, tanımların kalıcılığının sağlanması ve kullanıcıların bu tanımları kullanmaları ve bu tanımlara uygun işlem yapmaları sağlanmış olur (Yarımağan, 2000, s.2).

Veritabanında veri tanımlarının yer aldığı yapı *veri sözlüğü* (data dictionary) olarak da adlandırılabilir.

Kavramsal ve fiziksel düzeyler arasında açık biçimde ayırımın olduğu VTSY'lerde veri tanımlama dili yalnızca kavramsal şemayı belirlemek için kullanılır. Böylesi VTSY'lerde fiziksel şemayı belirlemek içinse *depolama tanımlama dili* (storage definition language) adı verilen bir başka dilden yararlanılır. İki şema arasındaki eşleştirme ise veri tanımlama dili ya da depolama dilinden biri kullanılarak belirlenir.

Günümüzde kullanılan ilişkisel VTSY'lerin çoğunda depolama tanımlama dilinin görevini yerine getiren belirli bir dil yoktur. Bunun yerine fiziksel şema fonksiyonları, parametreler ve depolamayla ilgili ayrıntıların birleşimiyle belirlenir. Bu, veritabanı yönetici personeline indeksleme seçenekleri ve depolama için verinin eşleştirilmesini kontrol etme izni verir.

Üç şemali mimarinin tam anlamıyla kullanıldığı yapıarda *görünüm tanımlama dili* (view definition language) olarak adlandırılan üçüncü bir dile ihtiyaç duyulur. Görünüm tanımlama dili, kullanıcı görünümlerini belirlemek ve bunların kavramsal şemadaki eşleştirmelerini belirlemek için kullanılır. Fakat pek çok veritabanının *veri tanımlama dili* hem kavramsal hem de dışsal (görünüm) şemaların her ikisini de tanımlamak için kullanılır. Örneğin, ilişkisel VTSY'lerinde SQL (yapışsal sorgulama dili) görünüm tanımlama dili rolüyle tanımlanır ve önceden tanımlanmış bir sorgunun sonucu olarak kullanıcı ya da uygulama görünümlerini tanımlar.

Veritabanı tanımları **Veri Tanımlama Dili** kullanılarak oluşturulur. Daha sonra bu tanımlar **Veri Tanımlama Dili Derleyicisi** kullanılarak çözümlenir ve gerektiğinde kullanıcıya iletilmek üzere VTSY tarafından uygun yapılara dönüştürüllerak saklanır.

Veritabanı şemaları yukarıda sözü edilen biçimde tanımlanıp derlendiğinde ve ilgili veriler veritabanına kaydedildiğinde veritabanı, üzerinde veri düzenleme işlemi yapılabılır duruma gelir. Burada sözü edilen veri işleme; veritabanından veri çağrıma, veri ekleme, veri silme ve veri üzerinde değişiklikler yapma faaliyetlerini içerir. VTYS bu işleme faaliyetlerini gerçekleştirebilmek amacıyla *veri işleme dili* (data manipulation language) olarak adlandırılan bir dil kullanır. Veritabanından bilgi alma amacıyla sorgulama yapmak için kullanılan veritabanı dili ise *sorgulama dili* (query language) olarak adlandırılır.

Günümüzde kullanılan VTYS'lerinde yukarıda belirtilen dil çeşitleri genellikle birbirinden ayrı diller olarak düşünülmez aksine tüm dil çeşitlerinin görevini yerine getiren geniş kapsamlı birleştirilmiş bir dil kullanılır. Kapsamlı birleştirilmiş dile tipik örnek ilişkisel veritabanı dili SQL (*Structured Query Language-Yapısal Sorgulama Dili*)'dır.

Veritabanı Yönetim Sistemlerinin Bileşen Modülleri

Veritabanı ve VTYS kataloğu genellikle disk üzerinde depolanır. Diske erişim, öncelikle disk üzerinde okuma/yazma işlemlerini programlayan işletim sistemi tarafından kontrol edilir. Çoğu VTYS'nin, diskin okunup/yazılmasını programlamak için kendi *ara bellek yönetim modülü* (buffer management module) bulunur. Böylece diskin okuma/yazma görevini azaltan bu modül veritabanı performansını dikkate değer ölçüde arttırmır. VTYS'nin daha yüksek düzey modülü olan *depolanmış veri yöneticisi* (stored data manager) ise veritabanının ya da katalogun bir bölümü olsun ya da olmasın, diske depolanmış VTYS bilgilerine erişimi kontrol eder.

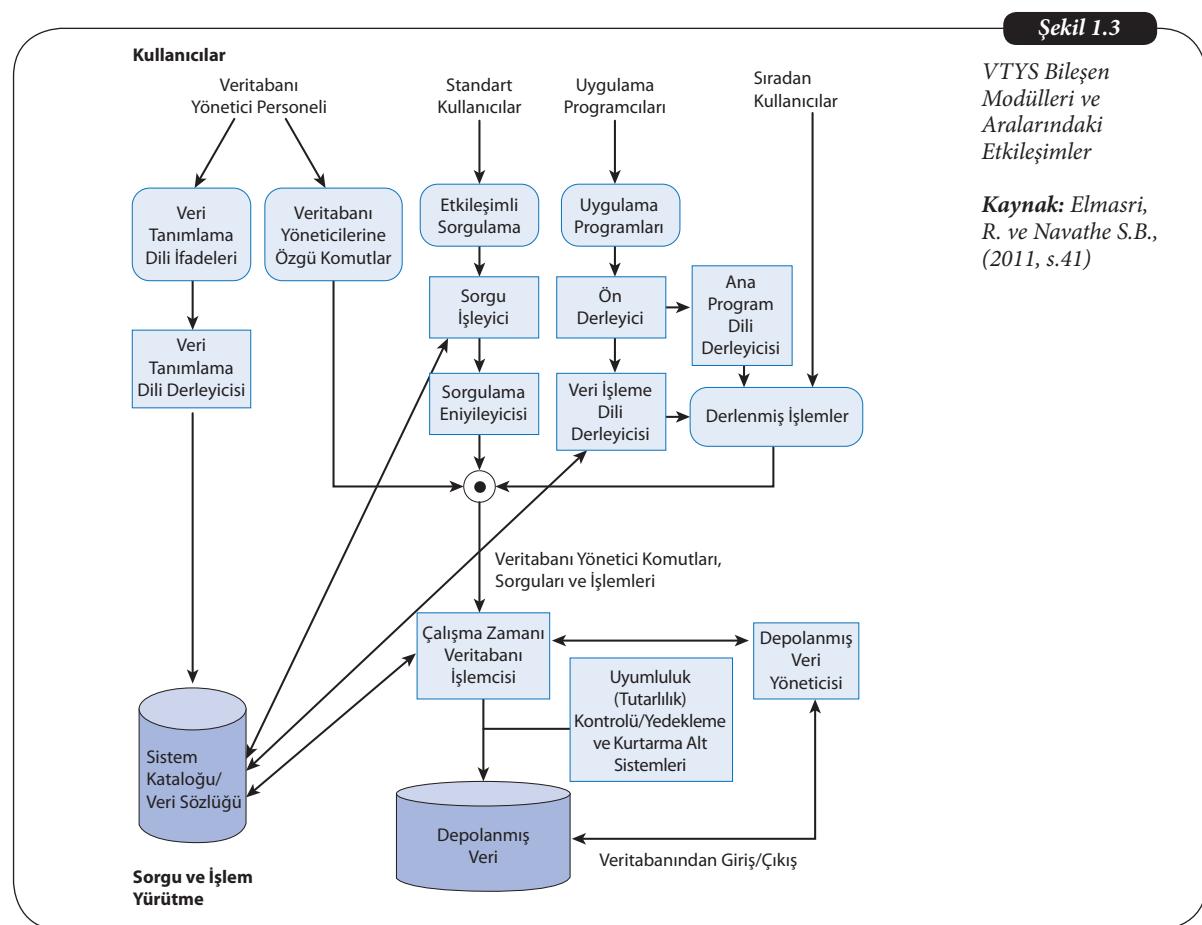
Şekil 1.3, tipik VTYS bileşenlerini basitçe açıklamaktadır. Şekil iki parçaya ayrılmış olup üst kısmı veritabanının çeşitli kullanıcılarını ve veritabanı ile etkileşimlerini göstermektedir. Alt kısmı ise, verilerin depolanması ve veri üzerinde işlem yapmaktan sorumlu iç bileşenlerini göstermektedir.

Öncelikle Şekil 1.3'ün üst kısmını inceleyelim. Bu kısım, veritabanı yönetici personelleri, sorguları formüle etmek için etkileşimli ara yüzlerle çalışan standart kullanıcılar, programlama dillerini kullanarak program oluşturan uygulama programcısıları ve önceden tanımlanmış işlemler için parametreler sağlayarak veri giriş işlerini yapan sıradan kullanıcılar için ara yüzleri göstermektedir. Veritabanı yönetici personelleri veri tanımlama dili ve veritabanı yönetici personeline özgü komutlar yardımıyla veritabanı tanımlarını ve veritabanı üzerinde gerekli değişiklikleri yaparlar.

Veri tanımlama dili derleyicisi, veri tanımlama dilinde belirlenmiş şema tanımlarını işler ve şemaların tanımlarını (meta-veri) VTYS kataloğına depolar. Katalog; dosyaların isimleri ve boyutları, veri kalemlerinin adları ve veri türleri, her dosyanın depolama ayrıntıları, şemalar arasındaki bilgi eşleştirme ve kısıtlamalar gibi bilgileri içerir. Ayrıca katalog, VTYS modüllerinin ihtiyaç duyduğu birçok bilgi türünü de depolar. Böylece daha sonra ihtiyaç duyulan bilgiler için kataloğa başvurulabilir.

Veritabanından nadiren bilgi ihtiyacı olan standart kullanıcılar, *etkileşimli sorgulama* olarak adlandırılan, bir tür ara yüz formu kullanarak veritabanı ile etkileşime geçerler.

Veritabanına iletilen sorguların, anlamsal ve sözdizimsel çözümlemesi, kullanıcı yetkisinin kontrolü ve sorguları işlemek için hangi algoritmanın kullanılması gerektiğini belirleme işi *sorgu işleyici* (query processor) adı verilen bileşen tarafından gerçekleştirilir. Sorgu işleyicisinin alt bileşenlerinden biri olan *sorgu eniyileyici* (query optimizer) ile işlemlerin yeniden düzenlenmesi ve yeniden sıralanması, fazlalıkların elenmesi ve yürütüm boyunca doğru algoritma ve indekslerin kullanılmasıyla sorgunun en iyi biçimde işletilmesi sağlanır. Sorgu eniyileyici, depolanmış veri hakkında istatistikî ve diğer fiziksel bilgiler için sistem kataloğına başvurarak sorgu için gerekli işlemleri uygulayan yürütülebilir kodları üretir. Söz konusu bileşen bu işlemleri yaparken ayrıca çalışma zamanı işlemcisile de etkileşim hâlindedir.



Uygulama programcıları, Java, C veya C++ gibi ana programlama dillerinde yazılan ve bir ön derleyiciye gönderilen programlar yazar. Ön derleyici ana programlama dilleri ile yazılan ve uygulama programından gelen veri işleme komutlarını çözümler. Programın geri kalan komutları da ana programlama dili derleyicisine gönderilir. Veri işleme komutları ve programın geriye kalan komutları; yürütüm kodları, çalışma zamanı veritabanı işlemci ile bağlantılı kılınarak kaydedilir. Kaydedilmiş işlemler, parametrelerle işlem yapan sıradan kullanıcılar tarafından tekrar tekrar yürütülür. Her yürütüm, ayrı bir işlem olarak düşünülür. Buna örnek olarak, hesap numarası ve para miktarı parametreleri kullanılarak bankadan para çekme işlemi verilebilir.

Şekil 1.3'ün alt kısmında, *çalışma zamanı veritabanı işlemci* (runtime database processor) veritabanı yöneticilerine özgü komutları, yürütülebilir sorgu planlarını ve parametreleri önceden belirlenmiş işlemleri yürütür. Sistem kataloğu ve veri sözlüğü ile birlikte çalışır ve onları güncelleyebilir. Ayrıca, disk ve ana bellek arasındaki düşük düzey girdi/çıktı (okuma/yazma) işlemlerini yerine getirmek amacıyla temel işletim sistemi hizmetlerini kullanan *depolanan veri yönetici* ile de etkileşim hâlindedir. Şekilde, *uyumluluk kontrolü* ile *yedekleme ve kurtarma alt sistemleri* tek modül olarak gösterilmiştir. Bunlar, veritabanı işlemleri için yönetimi için çalışma zamanı veritabanı işlemciyle bütünlüğe taraflarıdır.

Veri sözlüğü şemalar ve kısıtlar hakkında katalog bilgisini depolamanın yanı sıra tasarım kararları, kullanım standartları, uygulama programı açıklamaları ve kullanıcı bilgileri gibi diğer bilgileri de depolar. Böyle bir sistem *bilgi havuzu* olarak da adlandırılır. Bu bilgiye kullanıcılar veya veritabanı yönetici personeli tarafından gereksinim duyulduğunda doğrudan erişilebilir. Veri sözlüğü VTYS kataloğuna benzemekle birlikte daha geniş bilgi çeşitleri içerir ve esas olarak VTYS yazılımından ziyade kullanıcılar tarafından erişilir.

SIRA SİZDE



Veritabanlarında üç şema mimarisi hangi amaçla kullanılır?

VERİTABANI TÜRLERİ

Daha önce de söz edildiği üzere her veritabanı yönetim sistemi bir veri modeli kullanır. Veritabanında yer alacak veriler ve veriler arasında kurulacak ilişkiler mantıksal olarak ilgili veri modeline göre yapılandırılır ve veritabanları da buna göre sınıflandırılır.

Geçmişten günümüze kadar geliştirilmiş olan çok sayıda veri modeli, kullandıkları teknikler açısından dört temel başlıkta incelenir. Bu dört veri modelinden hangisini kullanmasına bağlı olarak veritabanları da aşağıda verilen dört başlık altında sınıflandırılabilir:

1. Hiyerarşik veritabanı (Hierarchical database)
2. Ağ veritabanı (Network database)
3. İlişkisel veritabanı (Relational database)
4. Nesneye yönelik veritabanı (Object oriented database)

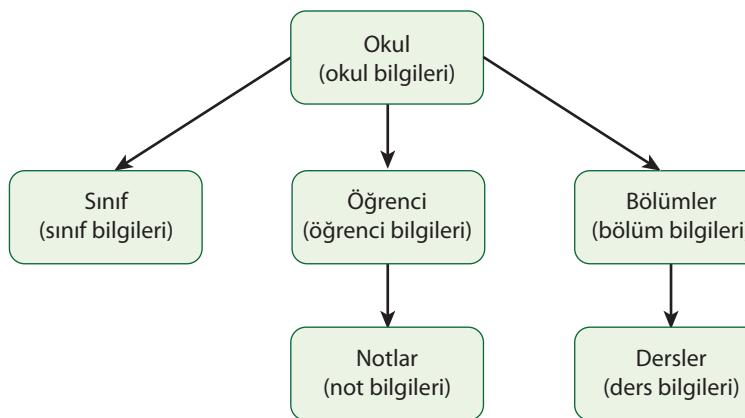
Hiyerarşik Veritabanı

Hiyerarşik veritabanı, en eski veri modeli olan hiyerarşik veri modelini temel alıp 1960 ve 1970'li yıllarda yaygın olarak kullanılmıştır. Bu tür veritabanlarında kullanılan veri modelinde kayıtlar, ilişkileri temsil eden ve ağaç yapısına benzeyen kök ve dallar biçiminde hiyerarşik bir yapıda oluşturulur. Bu yapı, başlangıç noktası ağacın kökü, bağlantılı kılınacak noktalar dallar ve ana dallara bağlı alt dallar olarak düşünülebilen bir yapı biçimindedir. Buna göre veriler arasındaki ilişkilerde hiyerarşinin üst bölümünde olan dallar alt bölümde birden çok dal ile bağlantılı olabilirken alt bölümünde olan dallar üstte kalan dallara yalnızca tek bir noktadan bağlantılı olabilirler. Bu nedenle bu yapıda ilişkiler ebeveyn-çocuk (bir ebeveynin birden fazla çocuğu olabilirken, bir çocuğun yalnızca bir ebeveyni olması) ilişkisine benzetilir. Bu model sunucu bilgisayarlarda çalışan yazılımlar tarafından kullanılmaktadır. Bu biçimde en çok kullanılan yazılım IBM firmasının geliştirdiği IMS (Information Management System) dir. Bu yapıda gereksiz veri tekrarı sözkonusu olup erişimde de sıkıntılardan yaşanabilecektir. Herhangi bir veriye erişilmek istendiğinde arama kök düğümden başlayarak aşağı doğru devam eder. Model yapısında herhangi bir dal silinirse bu dala bağlı tüm alt dallar ve dolayısıyla ilgili tüm veriler de silinmiş olur.

Şekil 1.4

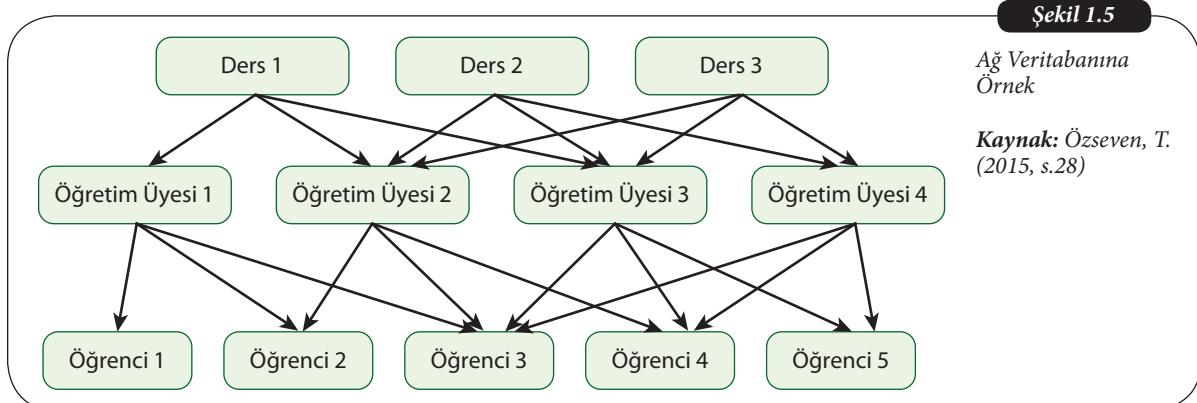
Hiyerarşik
Veritabanına Örnek

Kaynak: Özseven, T.
(2015, s.28)



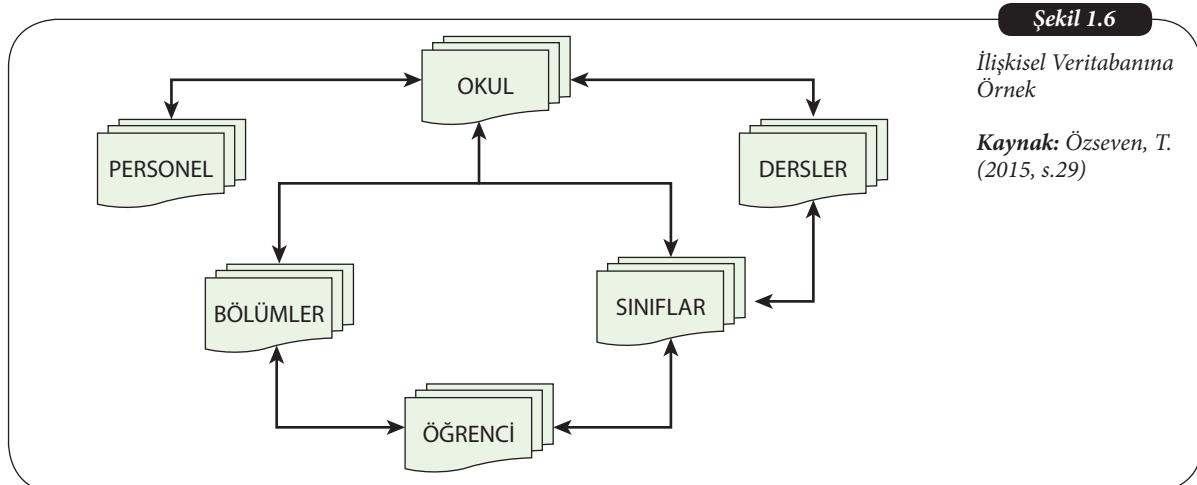
Ağ Veritabanı

Ağ veritabanı, 1970'li yıllar ile 1980'li yılların ilk yarısında kullanılan ve ağ veri modeli temel alan veritabanı türüdür. Hiyerarşik veri modelindeki ebeveyn-çocuk ilişkisinin yetersizliği ağ veri modeliyle giderilmeye çalışılmıştır. Buna göre her bağlantı noktası düğüm olarak ifade edilirse hiperarşik yapıdan farklı olarak ağ veri modelinde, her düğümün birden fazla ebeveyn ve birden fazla çocuk düğümü ile bağlantısı olabilir.



Ilişkisel Veritabanı

Bu tür veritabanı ilişkisel veri modelini temel almış ve ilk olarak 1970 yılında ortaya atılmıştır. 1970'li yılların sonunda kullanılmaya başlanmış ve 1985 yılından sonra kullanımı yaygınlaşmıştır. Bu yapıda ilk iki veri modelinden farklı olarak birden çok ilişki biçimini kullanılabılır. Günümüzde kullanılan veritabanı yönetim sistemlerinin hemen hemen hepsinde tercih edilen model ilişkisel veri modelidir. Bu nedenle modele ilişkin ayrıntılı bilgiler Ünite 3'te ele alınmıştır.



Nesneye Yönelik Veritabanı

Günümüzde kullanılan ve gelecekte de kullanılacak pek çok uygulamada yalnızca harf, rakam ya da çeşitli karakterler kullanılarak yapılandırılmış verileri değil aynı zamanda multimedya (çeşitli çizim, fotoğraf, görüntü, ses ya da video gibi nesneleri) de içeren veritabanı yönetim sistemlerine ihtiyaç duyulmaktadır. Verileri satırlar ve sütunlar biçiminde düzenlemek için tasarlanmış olan veritabanı yönetim sistemleri grafik unsurları ve multimedya unsurlarını kullanmaya pek uygun değildir. Bu nedenle bu eksikliği gidermek amacıyla nesneye yönelik veri modelleri geliştirilmiştir. Söz konusu bu modelleri kullanan veritabanları da nesneye yönelik veritabanı olarak adlandırılmaktadır.

Nesneye-yönelik veri tabanları 1990'lı yıllarda kullanılmaya başlanmıştır ve giderek popüler hâle gelmektedir. Bunun nedeni, çeşitli multimedya unsurlarını ya da çeşitli kaynaklardan parça parça alınan verileri benzer biçimde birleştiren web uygulamalarında kullanılan Java uygulamalarını yönetmek için kullanılabilir olmasıdır. Nesneye yönelik veritabanları ilişkisel veri modellerinden farklı olarak daha karmaşık veri türleri üzerinde işlem yapmasına rağmen, çok sayıda işlemi yürütme açısından ilişkisel veritabanından göreceli olarak daha yavaştır. Bu nedenle günümüzde hem ilişkisel hem de nesneye yönelik veri modellerini birlikte kullanan veritabanı yönetim sistemlerinin yaygınlaştığı görülmektedir.

VERİTABANI YÖNETİM SİSTEMİ YAZILIMLARI

Günümüzde yaygın olarak bilinen veritabanı yönetim sistemi yazılımları; MS SQL Server, Oracle, MySQL, Sybase, MS Access, PostgreSQL, IBM DB2, Informix, Advantage biçiminde sıralanabilir.

MS SQL Server: Microsoft firması tarafından geliştirilmiş ve ilişkisel veri modelini kullanan veritabanı yönetim sistemi yazılımıdır. Yalnızca Windows işletim sistemlerinde çalışır. Sunucu ya da istemci olarak kurulabilir.

Oracle: Oracle firması tarafından geliştirilmiş ilişkisel veritabanı yönetim sistemi yazılımıdır. Özellikle kurumsal amaçlı kullanılan bu yazılım büyük miktarda veriyi çok kullanıcılı ortamlarda saklama ve güvenli erişim sağlama açısından, yüksek ölçekli uygulamalar için tercih edilir. Birçok işletim sistemi üzerinde kullanılabilen bu yazılımın maliyeti de yüksektir.

MySQL: Açık kaynak kodlu bir veritabanı yönetim sistemi yazılımıdır. Unix, OS/2 ve Windows işletim sistemlerinde ücretsiz olarak kullanılabilmektedir. Bunun dışında ticari lisans seçeneği de mevcuttur. Özellikle web ortamında yaygın olarak kullanılan MySQL'in kullanımı kolay kullanıcı yetkilendirme sistemi güçlündür. Python'dan Java'ya kadar birçok programlama dili ile erişilebilmektedir.

Sybase: Avrupanın en büyük yazılım şirketi olan SAP'a (Systemanalyse und Programmierung-Sistem Analizi ve Program geliştirme) bağlı Sybase firması tarafından geliştirilen veritabanı yönetim sistemi yazılımıdır. Orta ve büyük ölçekli uygulamalar için tercih edilmektedir. Ülkemizde daha çok bankacılık sektörü ile kamusal alanlarda kullanılmaktadır. Veritabanı üzerindeki işlemler T-SQL (Transact-SQL) sorulama dili ile gerçekleştiriliyor.

PostgreSQL: PostgreSQL açık kaynak kodlu ücretsiz bir yazılımdır. PostgreSQL, yeni nesil veritabanı yönetim sistemi araştırma prototipi olan POSTGRES yazılımının geliştirilmiş hâlidir. POSTGRES'in zengin veri tiplerini ve güçlü veri modelini kullanırken aynı zamanda SQL'in geliştirilmiş alt kümesi olan PostQuel dilini kullanır. İyi performans ve güvenli ve geniş özelliklere sahip bir veritabanı yönetim sistemi yazılımıdır. Hemen hemen tüm UNIX ve UNIX türevi (Linux, FreeBSD gibi) işletim sistemlerinde çalışır. Bunun dışında Microsoft Windows NT tabanlı işletim sistemlerinde de çalışırılabilir.

MS Access: Microsoft firmasının Microsoft Office yazılım paketi içinde yer alan veritabanı yönetim sistemi yazılımıdır. Küçük ölçekli uygulamalarda masaüstü veritabanı sistemi olarak kullanılır. Yalnızca Windows işletim sisteminde çalışan ve işletim sisteminin sağladığı güvenlik seçeneklerini kullanan bir yapıya sahiptir.

DB2: IBM firması tarafından geliştirilen veritabanı yönetim sistemi yazılımıdır. Küçük ölçekli işletmeler açısından maliyetli bir sistem olan DB2 büyük ölçekli uygulamalarda tercih edilir. Windows, Linux ve Unix işletim sistemlerinde kullanılabilir.

Özet



Veritabanına ilişkin temel kavramları tanımlamak

Veri; ham gözlemler, işlenmemiş gerçekler ya da izlenimlerdir. Bu gözlemler, gerçekler ya da izlenimler harf, rakam ya da çeşitli sembol ve işaretler yardımıyla temsil edilir. **Veritabanı** (database) herhangi bir konuda, birbiriyile ilişkili olan ve amaca uygun olarak düzenlenmiş, mantıksal ve fiziksel olarak tanımlanmış veriler bütündür. **Veritabanı Yönetim Sistemi** (VTYS); veritabanı tanımlamak, veritabanı oluşturmak, veritabanında işlem yapmak, veritabanının farklı kullanıcı yetkilerini belirlemek, veritabanının bakımını ve yedeklemesini yapmak için geliştirilmiş programlar bütündür. Veritabanı ve veritabanı yönetim sisteminin birlikte oluşturduğu bütün ise **veritabanı sistemi** olarak ifade edilir.



Geleneksel dosya sistemleri ile veritabanı yönetim sistemlerini karşılaştırmak

Veritabanı yaklaşımından önce kullanılan ve geleneksel dosya sistemi olarak ifade edilen yaklaşımada veriler bilgisayarda ayrı ayrı dosyalar biçiminde saklanmaktadır. Birbiriyile ilişkili olan veriler bir dosyada, başka bir açıdan birbiriyile ilişkili veriler de başka bir dosyada yer almaktaydı. Bununla birlikte iki farklı dosya içinde aynı verinin yer olması da söz konusu idi. Geleneksel dosya sistemlerinin ortaya çıkardığı sakincalar veritabanı yönetim sistemlerinin geliştirilmesinde önemli bir etken olmuştur. Geleneksel dosya sistemlerinde büyük ölçüde veri tekrarı ve veri tutarsızlığı sözkonusu olurken veritabanı yönetim sistemleri veri tekrarını ve tutarsızlığını önlemektedir. Benzer biçimde veri paylaşımına olanak vermemesi, uygulamalarda ihtiyaç duyulan değişikliklerin gerçekleştirilebilmesi için uzmanlık bilgisi gerektirmesi, istenilen verİYE ulaşmada güçlükler yaşanması, verilerin güvenliği ve gizliliği konusunda sorunlar yaşanması, veriler ve uygulamalarla ilgili belirli bir standart uygulanamaması, verileri yedekleme ve kurtarma konusunda güçlükler yaşanması geleneksel dosya sistemlerinin diğer dezavantajları olarak sıralanabilir. Sıralanan bu dezavantajlar veritabanı yönetim sistemleri ile giderilebildiğinden, sözkonusu bu özellikler veritabanı yönetim sistemlerinin üstünlükleri olarak karşımıza çıkmaktadır.

VTYS'ler, sağladıkları avantajlara rağmen, maliyet açısından geleneksel dosya sistemlerine göre dezavantajlıdır. VTYS için gerekli olan donanım, yazılım

ve veritabanı eğitimi için başlangıç yatırımlarının yüksek olması, veri güvenliğini sağlamak, aynı anda gerçekleştirilen işlemlerin kontrolü, veri kurtarma ve bütünlük fonksiyonları için gerekli olan sabit maliyetlerin yüksek olması VTYS'nin dezavantajlı yönlerini oluşturmaktadır.

Bu nedenle, değişmesi hiç beklenmeyen basit, iyi tanımlanmış veritabanı uygulamaları sözkonusu olduğunda geleneksel dosya sistemlerini kullanmak daha avantajlı olacaktır.



Veritabanı kullanıcılarını sınıflandırmak

Veritabanı kullanıcıları veritabanı ile etkileşim biçimlerine göre sınıflandırılırlar.

Veritabanı yöneticisi; veritabanına erişim yetkilerini belirleme, veritabanı kullanımının düzenlenmesi ve izlenmesini sağlamak, ihtiyaç duyulan yazılım ve donanım kaynaklarını edinme biçiminde sıralanan sorumluluklara sahiptir. Ayrıca güvenlik ihlalleri ve kötü sistem yanıt süresi gibi sorunların çözümünden de sorumludur.

Veritabanı tasarımcısı; veritabanında saklanacak olan verilerin tanımlanmasından ve bu verilerin depolanması ve gösterilmesi için gerekli olan uygun yapıların seçilmesinden sorumludur.

Son kullanıcılar; yaptıkları işler gereği veritabanına sorgulama ya da güncelleme yapmak veya rapor üretmek için erişen kullanıcılardır. Bu tür kullanıcılar veritabanı ile kullanıcının bağlantısını sağlayan ve uygulama programcıları tarafından geliştirilen yazılımları kullanırlar.

Sistem analisti son kullanıcıların, özellikle de sıradan son kullanıcıların gereksinimlerini belirleyen ve standart işlemler yoluyla bu gereksinimleri karşılayabilecek ayrıntıları belirleyen kişi ya da kişilerdir.

Uygulama programcıları ise sistem analisti tarafından belirlenen ayrıntıları program hâline getiren ve daha sonra test eden, hataları ayıklayan, belgeleyen ve kaydedilmiş işlemler olarak sürekliliğini sağlayan kişilerdir.



Veritabanı yönetim sistemlerinin mimarisini açıklamak

Veri modeli, bir veritabanının mantıksal yapısını tanımlamada kullanılacak kavramlar, işlemler ve kurallar bütündür. Veritabanının mantıksal yapısı; veri tipleri, veriler arasındaki ilişkiler, veri üzerinde uygulanacak kısıtlamalar vb. dir. Veri modelleri **kavramsal veri modeli, fiziksel veri modeli ve uygulama veri mo-**

deli olarak sınıflandırılırlar. Veritabanının herhangi bir veri modeliyle tanımlanması ise *veritabanı şeması* olarak adlandırılır. Veritabanı yönetim sistemlerinde genel olarak üç şema mimarisi kullanılır. Üç şema mimarisinin amacı kullanıcı uygulamalarını fiziksel veritabanından ayırt etmektir. Bu mimaride şemalar izleyen üç düzeyde tanımlanır:

İçsel (fiziksel) düzey; veritabanının *fiziksel depolama yapısını* tanımlayan içsel şemayı içerir. İçsel şema, veriyi depolama ayrıntılarının tamamını ve veritabanına erişim yollarını tanımlayan fiziksel veri modelini kullanır.

Kavramsal düzey; kullanıcı topluluğu için tüm veritabanının yapısını tanımlayan kavramsal şemayı içerir. Kavramsal şema fiziksel depolama yapısının ayrıntılardan gizler ve veritabanında yer alan verilerin tipine, veriler arası ilişkilere, kullanıcı işlemlerine ve kısıtlara ilişkin tanımlara yoğunlaşır.

Dışsal (görünüm) düzey; bir dizi dışsal şema ya da kullanıcı görünümü içerir. Her dışsal şema bir grup kullanıcının ilgilendiği bazı veritabanı bölümlerini tanımlar. Böylece veritabanının diğer kısmı bu kullanıcı grubundan gizlenir.

Çoğu VTYS tam ve açık olarak bu üç düzeye ayrılmaz fakat bu üçlü şema mimarisini bir ölçüde destekler. Üç şema mimarisine dayanan VTYS'lerinde her kullanıcı grubu kendi dışsal (görünüm) şemalarına başvurur. Bu nedenle VTYS, dış şemada yapılan belirli bir isteği kavramsal şema isteğine daha sonra da depolanmış veritabanı üzerinde işlem yapmak üzere içsel şema isteğine dönüştürmek zorundadır. Yapılan istek veritabanından bilgi çağrımak ise veritabanından çıkarılan veri, kullanıcının dışsal görünümüyle eşleşecek biçimde dönüştürülür. Şema düzeyleri arasında bu şekilde gerçekleştirilen dönüşümler *eşleştirme* (mapping) olarak adlandırılır.

Şema düzeyleri arasındaki dönüştürme kapasitesi kısaca *veri bağımsızlığı* (mantıksal ve fiziksel) olarak ifade edilir.

VTYS'lerinde veritabanı tanımları *veri tanımlama dili* kullanılarak oluşturulur. Veritabanından veri çağrıma, veri ekleme, veri silme ve veri üzerinde değişiklikler yapma faaliyetlerini gerçekleştirebilmek amacıyla kullanılan dil *veri işleme dili* olarak adlandırılırken veritabanından bilgi alma amacıyla sorgulama yapmak için kullanılan veritabanı dili ise *sorgulama dili* olarak adlandırılır.

Günümüzde kullanılan VTYS'lerinde yukarıda belirtilen dil çeşitleri genellikle birbirinden ayrı diller olarak düşünülmektedir. Aksine tüm dil çeşitlerinin görevini yerine getiren geniş kapsamlı birleştirilmiş bir dil kullanılır. Kapsamlı birleştirilmiş dile tipik örnek ilişkisel veritabanı dili *SQL (Structured Query Language-Yapısal Sorgulama Dili)*dir.



Veritabanı türlerini ve yaygın olarak kullanılan veritabanı yönetim sistemi yazılımlarını sıralamak

Veritabanları, kullandıkları veri modeli temel alınarak aşağıda verilen biçimde sıralanabilir:

- Hiyerarşik veritabanı
- Ağ veritabanı
- İlişkisel veritabanı
- Nesneye yönelik veritabanı

Yaygın olarak kullanılan veritabanı yönetim sistemi yazılımları ise; MS SQL Server, Oracle, MySQL, Sybase, MS Access, PostgreSQL, IBM DB2, Informix, Advantage biçiminde sıralanabilir.

Kendimizi Sınayalım

1. Aşağıdakilerden hangisi herhangi bir konuda, birbiriyle ilişkili olan ve amaca uygun olarak düzenlenmiş, mantıksal ve fiziksel olarak tanımlanmış veriler bütününe ifade eder?

 - a. Bilgi
 - b. Bilgi deposu
 - c. Bellek
 - d. Veritabanı
 - e. Veritabanı yönetim sistemi
2. Aşağıdakilerden hangisi bilgisayar tabanlı bilgi sistemleri kapsamında kullanılan ve farklı uygulamalar tarafından paylaşılan ortak verilerin düzenlenmesi, saklanması ve kullanılması amacıyla yönelik yaklaşımlardan biridir?

 - a. Bilgi sistemleri
 - b. Erişim sistemleri
 - c. Uygulama programları
 - d. İşletim sistemleri
 - e. Geleneksel dosya sistemleri
3. Aşağıdakilerden hangisi veritabanında yer alan bir verinin farklı uygulamalarda kullanımı sözkonusu olduğunda veri üzerinde yapılacak bir değişimin verinin kullanıldığı diğer uygulamalara da yansıtılması anlamına gelir?

 - a. Veri paylaşımı
 - b. Veri bütünlüğü
 - c. Veri gizliliği
 - d. Veri güvenliği
 - e. Veride standart yapı
4. Aşağıdakilerden hangisi veritabanı yönetim sistemlerinin avantajlarından biri **değildir**?

 - a. Gereksiz veri tekrarı ve veri tutarsızlığının önlenmesi
 - b. Veri paylaşımının sağlanması
 - c. Bakım ve güncelleme gerektirmemesi
 - d. Kullanımda üst düzey uzmanlık gerektirmemesi
 - e. Standart yapı ve kuralların uygulanabilir olması
5. Aşağıdakilerden hangisi veritabanı kullanıcıları sınıflandırmasında son kullanıcılar sınıfı içinde yer alır?

 - a. Sistem analistleri
 - b. Uygulama programcılar
 - c. Parametrik kullanıcılar
 - d. Profesyonel kullanıcılar
 - e. Veritabanı tasarımcısı
6. Aşağıdakilerden hangisi, verilerin düzenlenmesi ve depolanmasına ilişkin ayrıntıların gizlenmesi ve verinin daha iyi anlaşılmasını sağlamak için veriye ilişkin temel özeliliklerin vurgulanmasını ifade eder?

 - a. Veri vurgulama
 - b. Veri soyutlama
 - c. Veri modelleme
 - d. Veri sınıflama
 - e. Veri eşleştirme
7. Aşağıdaki veri modellerinden hangisi veri modellerinin sınıflandırılmasında hem son kullanıcılar tarafından kolaylıkla anlaşılabilen kavramları hem de verinin bilgisayarda depolanması yöntemlerine ilişkin kavramları içerir?

 - a. Kavramsal veri modelleri
 - b. Yüksek düzeyli veri modelleri
 - c. Fiziksel veri modelleri
 - d. Düşük düzeyli veri modelleri
 - e. Uygulama veri modelleri
8. Aşağıdakilerden hangisi VTYS'lerinde kullanılan dillerden biri **değildir**?

 - a. Raporlama dili
 - b. Veri tanımlama dili
 - c. Veri işleme dili
 - d. Depolama tanımlama dili
 - e. Sorulama dili
9. Veritabanındaki kayıtların ilişkileri temsil eden kök ve dallar biçiminde düzenlenen yapılarla oluşturduğu veritabanı türü aşağıdakilerden hangisidir?

 - a. Ağ veritabanı
 - b. Hiyerarşik veritabanı
 - c. İlişkisel veritabanı
 - d. Nesneye yönelik veritabanı
 - e. Karma veritabanı
10. Aşağıdakilerden hangisi VTYS yazılımlarından biri **değildir**?

 - a. MS SQL Server
 - b. MS Access
 - c. Sybase
 - d. Unix
 - e. DB2

Kendimizi Sınayalım Yanıt Anahtarları

1. d Yanınız yanlış ise “Veritabanına İlişkin Temel Kavramlar” konusunu yeniden gözden geçiriniz.
2. e Yanınız yanlış ise “Geleneksel Dosya Sistemleri ve Veritabanı Yönetim Sistemleri” konusunu yeniden gözden geçiriniz.
3. b Yanınız yanlış ise “Veritabanı Yönetim Sistemlerinin Avantajları” konusunu yeniden gözden geçiriniz.
4. c Yanınız yanlış ise “Veritabanı Yönetim Sistemlerinin Avantajları” konusunu yeniden gözden geçiriniz.
5. c Yanınız yanlış ise “Veritabanı Kullanıcıları” konusunu yeniden gözden geçiriniz.
6. b Yanınız yanlış ise “Veritabanı Yönetim Sistemlerinin Mimarisi” konusunu yeniden gözden geçiriniz.
7. e Yanınız yanlış ise “Veri Modellerinin Sınıflandırılması” konusunu yeniden gözden geçiriniz.
8. a Yanınız yanlış ise “Veritabanı Yönetim Sistemlerinde Kullanılan Diller” konusunu yeniden gözden geçiriniz.
9. b Yanınız yanlış ise “Veritabanı Türleri” konusunu yeniden gözden geçiriniz.
10. d Yanınız yanlış ise “Veritabanı Yönetim Sistemi Yazılımları” konusunu yeniden gözden geçiriniz.

Sıra Sizde Yanıt Anahtarları

Sıra Sizde 1

Bilindiği gibi işletmeler tüm faaliyetlerini yerine getirebilmek, farklı düzeylerde kararlar alabilmek için bilgiye ihtiyaç duyarlar. Bilgi, ham gözlemler, işlenmemiş gerçekler ya da iznimler olarak tanımlanan **veriler** üzerinde kaydetme, sıralama, sınıflama, hesaplama, analiz vb. işlemlerin uygulanmasıyla elde edilir. İşletmelerde, üzerinde işlem uygulanarak verilecek kararlara destek sunacak biçimde dönüştürülecek verilerin belirli bir düzen içinde tutulması ve istenildiği anda erişilebilmesi önemlidir. Veritabanı herhangi bir konuda, birbirile ilişkili olan ve amaca uygun olarak düzenlenmiş, mantıksal ve fiziksel olarak tanımlanmış veriler bütündür. Veritabanı işletmenin farklı bölümleri tarafından ortak olarak kullanılan verilerin, gereksiz tekrardan arındırılmış biçimde saklanması ve gerektiğinde yalnızca yetkili kişiler tarafından erişimini olanaklı kılmalıdır. Bu nedenle veritabanları verilere güvenli biçimde ve hızlıca erişim sağlama açısından işletmelere büyük kolaylıklar sunmaktadır.

Sıra Sizde 2

Geleneksel dosya sistemlerinde veriler bilgisayarda ayrı ayrı dosyalar biçiminde saklanır ve birbirile ilişkili olan veriler ayrı ayrı dosyalar biçiminde düzenlenirdi. Bununla birlikte iki farklı dosya içinde aynı verinin yer olması da söz konusu idi. Zaman içinde artan veri miktarı, bu verinin depolanması için gerekli olan kapasite, veriye erişim ve işleme hızında yaşanan sıkıntılardan geleneksel dosya sisteminin temel sınırlılıkları olarak ortaya çıkmıştır. Bu sorunu gidermek amacıyla geliştirilen veritabanı yönetim sistemlerinin, aynı zamanda geleneksel dosya sistemlerinden ayrıldığı noktalar olan temel karakteristikleri ise aşağıdaki biçimde sıralanabilir:

- Veri tekrarı ve veri tutarsızlığını önler.
- Veri paylaşımına olanak verir.
- Uzmanlık bilgisine ihtiyaç duyulmayacak derecede kullanım kolaylığı sağlar.
- İhtiyaç duyulan veriye, tanımlanmış kullanıcı yetkileri kapsamında kolaylıkla erişilmesini sağlar.
- Veri güvenliği ve gizliliğini güçlü bir biçimde yerine getirir.
- Veriler ve uygulamalarla ilgili standart yapı ve kuralların olması kullanım kolaylığı sağlar.
- Verileri yedekleme ve kurtarma konusunda kolaylık sağlayan programlar barındırır.

Sıra Sizde 3

Veritabanı yöneticisi; veritabanına erişim yetkilerini belirler, veritabanı kullanımını düzenler ve izler. Bunun yanısıra ihtiyaç duyulan yazılım ve donanım kaynaklarını edinme, güvenlik ihlallerini ya da sistemin yanıt verme süresinde yaşanan sıkıntılardan çözümü de veritabanı yöneticisinin sorumlulukları arasındadır.

Veritabanı tasarımcısı; veritabanında saklanacak olan verilerin tanımlanmasından ve bu verilerin depolanması ve gösterilmesi için gerekli olan uygun yapıların seçilmesinden sorumludur. Veritabanı tasarımcısı tasarım boyunca, potansiyel kullanıcı gruplarıyla karşılıklı etkileşim hâlinde olup bu grupların veriye erişimini ve veri üzerinde işlem yapabilmelerini olanaklı kılan kullanıcı görünümelerini geliştirirler. Her kullanıcı grubuna ilişkin görünüm, analiz edilerek diğer kullanıcı gruplarına ilişkin görünümelerle bütünleştirilmeleri sağlanır. Veritabanı tasarımının tüm kullanıcı gruplarının ihtiyaçlarını destekleyecek kapasitede olması gereklidir.

Yukarıda verilen ayırmak keskin bir ayırmayıp her iki sorumluluğu aynı kişi ya da kişiler de alabilir. Veritabanı yöneticisi ile veritabanı tasarımcısının farklı kişiler olması veritabanını kullanacak olan kurumun/veritabanının büyülüklüğü ve kullanıcı sayısının faktörlere bağlıdır.

Yararlanılan ve Başvurulabilecek Kaynaklar

Sıra Sizde 4

Üç şema mimarisine göre veritabanlarında üç farklı düzeyde tanımlama yapılır. *İçsel şema*; veritabanının fiziksel depolama yapısını tanımlar. Bu tanımlar veriyi depolama ayrıntılarını ve veritabanına erişim yollarını içerir. *Kavramsal şema*; fiziksel depolama yapısının ayrıntılarını gizler ve veritabanında yer alan verilerin tipine, veriler arası ilişkilere, kullanıcı işlemlerine ve kısıtlara ilişkin tanımlara yoğunlaşır. *Dışsal şema*; kullanıcı görünümüyle ilgilenir. Her dışsal şema bir grup kullanıcının ilgilendiği bazı veritabanı bölümlerini tanımlar. Böylece veritabanının diğer kısmı bu kullanıcı grubundan gizlenir.

Üç şema mimarisinin amacı kullanıcı uygulamalarını fiziksel veritabanından ayırt etmektir. Üç şema mimarisine dayanan VTYŞ'lerinde her kullanıcı grubu yalnızca kendi dışsal (görünüm) şemalarına başvurur. Dış şemadan yapılan bu istek önce kavramsal şema isteğine daha sonra da depolanmış veritabanı üzerinde işlem yapmak üzere içsel şema isteğine dönüştürülür. Şema düzeyleri arasında bu şekilde gerçekleştirilen dönüşümler eşleştirme olarak adlandırılır. Şema düzeyleri arasındaki dönüştürme kapasiteleri mantıksal ve fiziksel veri bağımsızlığını sağlar. *Mantıksal veri bağımsızlığı*, kavramsal şemanın dışsal şemalarda ya da uygulama programlarında değişiklik yapılmaksızın değiştirilebilmesi anlamına gelir. *Fiziksel veri bağımsızlığı* ise kavramsal şemada bir değişiklik yapılmaksızın içsel şemada değişiklik yapma kapasitesidir. Buna bağlı olarak içsel şemada yapılan değişiklik dışsal şemalarda da bir değişiklik yapılmasını gerektirmez.

Sıra Sizde 5

Veritabanında yer alacak veriler ve veriler arasında kurulacak ilişkiler mantıksal olarak ilgili veri modeline göre yapılandırılır. Veritabanları da kullandıkları veri modeline göre sınıflandırılır.

Geçmişten günümüze kadar geliştirilmiş olan çok sayıda veri modeli, kullandıkları teknikler açısından dört temel başlıkta incelenir. Bu dört veri modelinden hangisini kullanğına bağlı olarak veritabanları da aşağıda verilen dört başlık altında sınıflandırılabilir:

- Hiyerarşik veritabanı
- Ağ veritabanı
- İlişkisel veritabanı
- Nesneye yönelik veritabanı

Aydın, S. (2013). *İşletme Bilgi Sistemleri*, (Editör: H. Durucasu), Eskişehir: Anadolu Üniversitesi Yayın No:2690, AÖF Yayın No:1656.

Barutçugil, İ. (2002). *Bilgi Yönetimi*, İstanbul: Kariyer Yayıncılık İletişim.

Burma, Z.A. (2009). *Veritabanı Yönetim Sistemleri ve SQL/PL-SQL/T-SQL*, Ankara: Seçkin Yayıncılık.

Elmasri, R. ve Navathe, S.B. (2011). *Fundamentals of Database Systems*, USA: Pearson Education, Inc.

Laudon, K.C. ve Laudon J.P. (2011). *Yönetim Bilişim Sistemleri Dijital İşletmeyi Yönetme*, Çeviri Editörü: U.Yozgat, Ankara: Nobel Akademik Yayıncılık.

O'brien, J.A., ve Marakas M. (2010). *Introduction to Information Systems in Business Management*, Fifteenth Edition, New York:McGraw-Hill Irwin.

Öğüt, A. (2003). *Bilgi Çağında Yönetim*, Ankara: Nobel Yayınevi Dağıtım.

Özseven, T. (2015). *Veritabanı Yönetim Sistemleri I*, Bursa: Ekin Basım Yayın Dağıtım.

Vural, Y. Ve Sağıroğlu, Ş. (2010). *Veritabanları Yönetim Sistemleri Güvenliği: Tehditler ve Korunma Yöntemleri*, Politeknik Dergisi, Cilt:13, Sayı:2 s.71-81.

Yarımagañ, Ü. (2000). *Veritabanı Sistemleri*, Ankara: Akademî Yayın Hizmetleri San.Tic. Ltd.Şti.

<http://quickbase.intuit.com/articles/timeline-of-database-history> (erişim tarihi: 26.11.2015)

<http://www.microsoft.com/tr-tr/server-cloud/products/sql-server-editions/overview.aspx> (erişim tarihi: 17.11.2015)

<http://www.mysql.com.tr/tr/KonuDetay.php?AKey=115&BKey=34&CKey=50> (erişim tarihi: 17.11.2015)

http://www.postgresql.org/message-id/attachment/17832/FAQ_turkish.html#1.1 (erişim tarihi: 17.11.2015)

Gündüz, D. <http://ab.org.tr/ab04/tammetin/178.pdf>

2

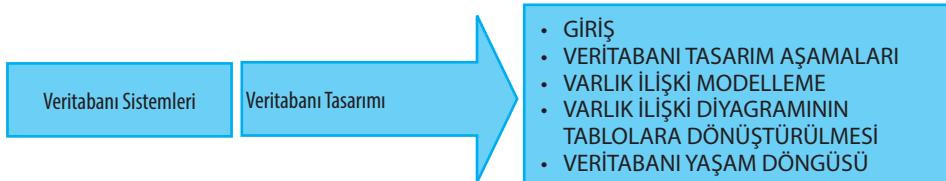
Amaçlarımız

- Bu üniteyi tamamladıktan sonra;
- 🕒 Veritabanı tasarım aşamalarını tanımlayabilecek,
 - 🕒 Varlık ilişki modelleme kavramlarını ve varlıklar arasındaki ilişkileri sıralayabilecek,
 - 🕒 Varlık ilişki modelini tablolara dönüştürebilecek,
 - 🕒 Veritabanı yaşam döngüsünün aşamalarını sıralayabilecek bilgi ve becerilere sahip olacaksınız.

Anahtar Kavramlar

- Veri Modelleri
- Varlık
- Öznitelik
- İlişki
- Kavramsal Tasarım
- Mantıksal Tasarım
- Fiziksel Tasarım
- Veritabanı Yaşam Döngüsü

İçindekiler



Veritabanı Tasarımı

GİRİŞ

İşletmelerde kullanılan bilgi sistemleri stratejik, organizasyonel gereksinimlerin karşılanması için veritabanı geliştirilmesine ihtiyaç duyar. Bu sayede müşteri desteği sağlanır, daha iyi üretim ve envanter takibi veya daha doğru satış planlaması yapılabilir. Veritabanı geliştirme süreci mevcut veri işleme adımlarının, genel işletme fonksyonlarının ve onların veri gereksinimlerinin analizi ile başlar. Böylece varsa mevcut veritabanındaki veriler ve işletmenin ihtiyaç duyduğu yeni veriler ortaya konulmuş olur. Veritabanında saklanacak veriler ve bunları kullanan geliştirilecek uygulamalar ve performans gereksinimlerinin belirlendiği bu aşama “gereksinim analizi” olarak adlandırılır. İzleyen aşamalarda ise kavramsal, mantıksal ve fiziksel veritabanı tasarımları gerçekleştirilir. Bu tasarım aşamaları izleyen kesimde verilmiştir.

Veritabanı tasarımını ifade etmek için üst seviye tasarım modelleri kullanılır. Bu modellerle veritabanı kullanıcılarının ihtiyaçlarının tasarımındaki karşılığı açık ve anlaşılır biçimde ortaya konulmuş olur. En yaygın kullanılan veritabanı tasarım modeli Varlık İlişki (ER-Entity Relationship model) modelidir. Günümüzde yaygın olarak kullanılmaya başlayan diğer bir modelde tümleşik modelleme dili şemasıdır (UML-Unified Modelling Language).

VERİTABANI TASARIM AŞAMALARI

Veritabanı tasarım aşamaları üç aşamadan oluşmaktadır. Bunlar kavramsal veri modelleme (conceptual data modeling), mantıksal veri modelleme (logical data modeling) ve fiziksel veri modellemedir (physical data modeling). Veritabanı tasarımını ifade eden modeller ile kullanıcılar ve geliştirme ekibi tasarıma son hâlini verir. İzleyen süreçlerde de veri tasarım modeli, geliştirme ekibi tarafından gerçekleştirir.

Kavramsal Veri Modeli

Kavramsal veri modeli (conceptual data model) veritabanında saklanacak verilerin, kısıtlarının ve ilişkilerinin bir gösterimidir. Kavramsal veri modelleme genellikle sistem analizi sırasında diğer gereksinim analizi ve yapılandırma adımları ile paralel olarak yapılır. Bu model ile kullanıcıların gereksinimleri ve bu gereksinimleri karşılaşacak tasarım tam olarak ifade edilmiş olur. Kavramsal veri modeli varlık ilişki diyagramının tasarılanması ya da tümleşik modelleme dili şeması ile oluşturulur. Kavramsal veri modeli sistem geliştirme süreci boyunca uygulanır.

Kavramsal Veri Modelinin Özellikleri

- İşletmenin genelini kapsayan iş kavramlarını kapsar.
- Stratejik veri projelerinde en sık kullanılan veri modelidir.
- Kullanıcı gereksinimlerini karşılamak için oluşturulur.
- Proje yöneticisi tarafından proje süreç izlemede kullanılır.
- Yüksek seviye veri yapılarını içerir.
- Yetkili ve yöneticilerin tüm seviyelerdeki mimari tanımların veri prensibini anlayabilecekleri teknik olmayan isimler kullanılır.
- Varlıklar, ilişkiler ve bunların özelliklerini içerir.

Mantıksal Veri Modeli

Mantıksal veri modeli (logical data model) Kavramsal modelin seçilen veritabanı yönetim sisteminin veritabanı şemasına dönüştürülmesidir. Bu kitapta veri tabanı yönetim sistemi olarak ilişkisel veri tabanı yönetim sistemi anlatılmaktadır. Bu nedenle mantıksal tasarımda da kavramsal modelin ilişkisel veritabanı şemalarına dönüştürülmesi açıklanmıştır. Mantıksal veri modeli ilişkisel veritabanı teknolojilerinden bağımsız, veri depolama, veri kısıtları ve ilişkiler tarafından tamamıyla bağlanmış veri modelidir. Bir mantıksal veri modeli insanları, yerleri, nesneleri, kuralları ve ilişkiler ile onlar arasındaki olayları standartlaştırır.

Mantıksal Veri Modelinin Özellikleri

- Tüm varlıkları ve aralarındaki ilişkileri içermektedir.
- Her varlığın tüm öznitelikleri belirlenmiştir.
- Her varlık için benzersiz anahtar belirtilir.
- Farklı varlıklar arasındaki ilişkiyi tanımlayan yabancı anahtarlar belirtir.
- Bu aşamada Normalleştirme uygulanır.
- Veri modeli kapsamına bağlı olmakla birlikte tipik olarak 50-1000 arasında varlık içerir.
- Veritabanı yönetim sisteminden, veri depolama konumlarından ya da teknolojilerinden bağımsız olarak oluşturulur.
- Varlıklar ve öznitelikler tanımlara sahip olur.
- Veri özniteliklerinin kısıtları ve veri türleri genellikle belirlenmiş olur.

Fiziksel Veri Modeli

Fiziksel veri modeli (physical data model) veri süreklilik teknolojisinin belirli bir sürümü ile bağlı, tamamıyla bağlanmış veri modelidir. Burada hedeflenen teknoloji ilişkisel bir veritabanı yönetim sistemi (ACCESS, MS SQL, ORACLE vb.) , XML dokümanı, elektronik tablo veya başka bir veri uygulama seçeneği olabilir. Fiziksel veri modeli veritabanının nasıl inşa edileceğini belirtir. Bir fiziksel veritabanı modeli, sütun adı, sütun veri türü, sütun kısıtlamaları, birincil anahtar, yabancı anahtar ve tablolar arasındaki ilişkiler de dâhil olmak üzere tüm tablo yapılarını göstermektedir. Bu aşamada veritabanının tipik yük altında istenilen performans ölçütlerini sağlama için tasarımında iyileştirmeler yapılır. Tablolarda indeksler oluşturma, tabloları birleştirme ve hatta önceki tasarım aşamalarından oldukça farklı yeni yapılar kullanmak söz konusu olabilir. Kavramsal ve mantıksal tasarımda değişikliğe gidilmesi durumunda önceden oluşturulmuş olan tasarım modelinin güncellenmesi gerekecektir.

Fiziksel Veri Modelinin Özellikleri

- Tüm tablo ve sütunlar belirtilir.
- Tablolar arasındaki ilişki yabancı anahtar ile sağlanır.
- Denormalization kullanıcı gereksinimlerine ve performans ölçütlerine dayalı olarak oluşabilir.
- Fiziksel veri modeli farklı ilişkisel veritabanı yönetim sistemleri için farklılık gösterebilir.
- Veritabanı yönetim sisteminin belirli bir sürümü, veri depolama konumları ya da teknolojileri için bağımlı olarak tasarlanmış ve geliştirilmiştir.
- Öznitelikler genellikle duyarlı ve uzunlukları atanmış veri türlerine sahip olacaktır.

VARLIK İLİŞKİ MODELLEME

Varlık ilişki modelleme bir işletme veya iş alanında kullanılan verilerin detaylı ve mantıksal gösterimidir. **Varlık ilişki modeli** gerçek hayatı varlıklar, aralarındaki ilişkiler ve varlıklar ile ilişkilerin özelliklerini içerir. Veri ilişki modelinin özellikle veritabanı tasarımındaki rolü önemlidir. Bu model ile kullanıcıların formal olmayan beklentileri veritabanı yönetim sisteminde gerçekleştirecek şekilde daha detaylı ve açık tanımlanmış olur. Varlık ilişki modeli grafiksel olarak **varlık ilişki diyagramı** ile ifade edilir.

İştekisel veritabanı modeli oluşturulabilmesi için öncelikle ilgili veriler çözümlenir, çözümlenen bu verilerin birbiri ile ilişkileri incelenir ve sonrasında varlık ilişki modelinden yararlanılıp varlık ilişki diyagramları oluşturulur. Bu varlık ilişki diyagramları veritabanında yer alacak olan tabloları, tabloların birbirlerine bağlantılarını ve tablolar ile bağlantılarının özelliklerini ortaya çıkarır ve ilişkisel veritabanı modelini oluşturur. Bu model ile ilk veritabanı tasarımları oluşturulmuş olur.

Varlık ilişki modeli (ER-Entity-relationship model):
Bir organizasyon veya işletmenin verilerinin mantıksal temsilidir.

Varlık ilişki diyagramı (ER-diagram): Varlık ilişki modelinin grafiksel gösterimidir.

Varlık ilişki modelmesinin amacı nedir açıklayınız?



Varlık ilişki modelinde veri yapısı grafiksel olarak gösterilirken üç temel öğe kullanılır. Bunlar;

- Varlıklar,
- Öznitelikler,
- İlişkilerdir.

Varlıklar (Entity)

Varlık işletmenin verisini tutmak istediği kullanıcı ortamındaki kişi, yer, nesne, olay veya kavramı temsil eder. Bu varlık çeşitleri için bazı örnekler şunlardır:

- | | | |
|--------|---|--------------------------------------|
| Kişi | : | Çalışan, öğrenci, hasta |
| Yer | : | Depo, ambar, şehir |
| Nesne | : | Makine, inşaat, araç |
| Olay | : | Satış, kayıt, sözleşme yenileme |
| Kavram | : | Muhasebe hesabı, okulda verilen ders |

Ortak özellikleri veya karakteristikleri içeren varlıkların oluşturduğu kümeye **varlık kümesi** denir. Varlık kümesindeki bir varlığa ise **varlık örneği** denir. Model içerisinde varlık kümesi dikdörtgen ile gösterilir ve içine varlığın adı yazılır. Mantıksal veritabanı tasarımda varlık kümesi tablo ile gösterilir. Şekil 2.1'de Çalışan, Araç ve Satış varlık kümelerinin grafiksel gösterimi verilmiştir.

Varlık kümesi(Entity type):
Ortak özellikleri paylaşan varlıkların koleksiyonudur.

Varlık örneği(Entity instance):
Varlık kümesinin özelliklerini taşıyan bir varlığın oluşudur.

Şekil 2.1

Çalışan, Araç ve Satış varlık kümelerinin gösterimi



Güçlü ve Zayıf Varlık Kümeleri

Bir organizasyonu tanımlayan temel varlık kümelerinin çoğunuğu güçlü varlık kümeleridir. Güçlü varlık kümesi diğer varlık kümelerinden bağımsız tanımlanabilen kümelerdir. Örneğin; çalışan, öğrenci, hasta güçlü varlık kümeleridir. Bu kümelerde her zaman varlıkların birbirinden benzersiz olarak ayıran bir öznitelik veya öznitelik birleşimi bulunur. Bunun tersine, bir varlık kümesi güçlü varlık kümesine bağlı olarak var olabiliyorsa zayıf varlık kümesi olarak adlandırılır. Şekil 2.2'deki Çalışan_Ayrıcalıkları zayıf varlık kümesine örnek verilebilir. Çalışan varlık kümesi olmadan Çalışan_Ayrıcalıkları kümesi var olamayacaktır. Zayıf varlık kümesini tanımlayan güçlü varlık kümesi tanımlayıcı sahip kümeye olarak da adlandırılır. Zayıf varlık kümesi çift çizgili dikdörtgen ile gösterilir.

Şekil 2.2

Güçlü ve zayıf varlık kümelerinin gösterimi; Çalışan, Çalışan_Ayrıcalıkları



Öznitelikler (Attributes)

Öznitelik(Attribute): Veri modelinde tanımlanan varlık veya ilişkilerin bir özellik ya da niteliğidir.

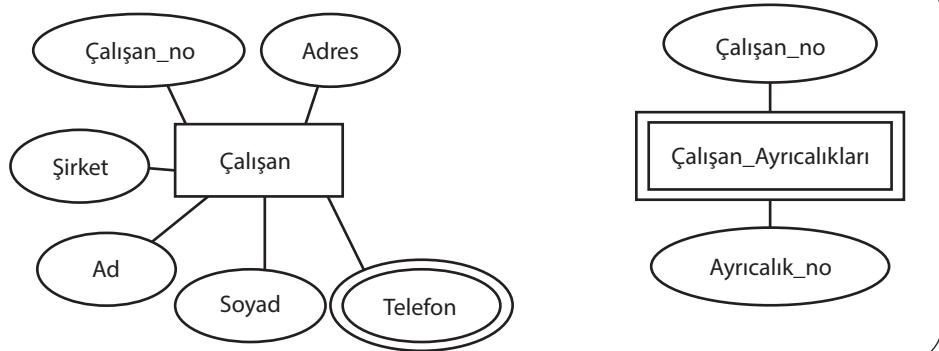
Her bir varlık kümelerinin özellikleri olur. Öznitelik, işletmede kullanılan varlık kümelerinin özellikleri veya karakteristikleridir (izleyen kesimde anlatılacak olan ilişkilerinde öznitelikleri olabilir). Modelde öznitelikler oval ile gösterilir ve içine adı yazılır. Öznitelik bağlı olduğu varlık kümese düz çizgi ile bağlanır. Şekil 2.3'te Çalışan ve Çalışan_Ayrıcalıkları varlık kümelerinin öznitelikleri gösterilmiştir. Çalışan: Çalışan_no, Şirket, Ad, Soyad, Telefon, Adres özniteliklerine, Çalışan_Ayrıcalıkları: Çalışan_no, Ayrıcalık_no özniteliklerine sahiptir. Aşağıda bazı varlık kümeleri ve bu kümelerin öznitelikleri örnek olarak verilmiştir.

Öğrenci : Öğrenci_no, Öğrenci_adi, Ev_adresi, Telefon, Doğum_tarihi

Araç : Sasi_No, Renk, Ağırlik, Motor_gücü

Şekil 2.3

Varlık kümeleri ve öznitelikleri



Öznitelikleri adlandırırken ilk harfi büyük karakter ile başlayıp ikinci kelime var ise alt çizgi ile birleştirerek yazılır. İzleyen kesimde anlatılacak ilişkilerinde öznitelikleri tanımlanabilir. Geleneksel varlık ilişki modelinde bir varlık kümesi ilişkili olduğu varlık kümesinin öznitelliğini içermez. Bu özellik veritabanında verinin tekrarlanması önerilir.

Bir öznitelik sadece bir varlık kümesi veya bir ilişki ile birleştirilebilir.



DİKKAT

Zorunlu ve Seçimli Öznitelikler

Bir özniteligin değeri varlık kümesi içindeki tüm varlık örnekleri için girilmesi zorunlu ise zorunlu öznitelik olarak adlandırılır. Varlık kümesi içindeki her bir örnek için değer girilmesi zorunlu olmayan öznitelikler ise seçimi öznitelik olarak adlandırılır. Model gösteriminde zorunlu özniteligin önüne "+" seçimi özniteligin önüne "o" sembolü konulabilir. Ancak pek çok gösterimde zorunlu ve seçimi öz nitelikler işaretlenmez ve ek dokümanlarda açıklanır.

Basit ve Birleşik Öznitelikler

Basit (atomik) öznitelik organizasyon için anlamlı daha küçük parçalara bölünmez. Örneğin Araç varlığının özniteliklerinin tamamı (Şasi_No, Renk, Ağırlık ve Motor_gücü) basit özniteliklerden oluşur. Ancak bazı öznitelikler daha detay bilgileri içeren anlamlı alt parçalara bölünebilir. Bu tip öznitelikler **birleşik öznitelikler** (çok parçalı) olarak adlandırılır. Örneğin Şekil 2.2'de verilen Çalışan varlığının adresi birleşik özniteliktir. Birleşik özniteligin alt parçalarına erişerek kullanmak gerekirse birleşik öznitelik anlamlı alt parçalara bölünmelidir. Adres özniteligi mahalle, cadde, kapı_no, postakodu ve şehir olarak anlamlı alt özniteliklere bölünebilir. Veritabanı tasarımcısı veritabanı kullanımında özniteliklerin kullanım desenlerini öngörerek özniteligi basit ya da birleşik olarak tanımlamalıdır. Varlık ilişki diyagramında birleşik özniteliklerin adı ve alt parçaları oval şecline içine yazılır.

Birleşik öznitelik(Composite attribute): Anlamlı alt parçaları olan bir özniteliktir.

Türetilen Öznitelikler

Bazı öznitelik değerleri veritabanında depolanan diğer ilişkili özniteliklerden hesaplanabilir veya türetilabilir. Bu oluşturulan yeni özniteligi **türetilen öznitelik** (derived attribute) denir. Örneğin, bir işletmede çalışanın işe başlama tarihini gösteren bir öznitelik (İş_başlama_tarihi) olsun. Bir çalışanın kaç yıl çalıştığı belirlenmek istenirse Çalışan_yılı özniteligi İş_başlama_tarihi özniteligi ve sorgu tarihi kullanılarak hesaplanabilir. Öğrenci varlık kümesinde bulunanların Doğum_tarihi özniteligi kullanılarak yaş özniteligi elde edilebilir. Türetilen öznitelikler varlık ilişki diyagramında kesikli çizgili oval şekillerle gösterilir.

Türetilen öznitelik (Derived attribute): Değer iğili olduğu diğer özniteliklerden hesaplanabilen özniteliktir.

Tek ve Çok Değerli Öznitelikler

Bir varlığın özniteligi sadece tek değer aliyorsa buna tek değerli öznitelik (single valued attribute) denir. Örneğin Çalışan varlık kümesi için Çalışan_no özniteligi tek değerli özniteliktir. Bir varlığın özniteligi birden çok değer aliyorsa buna **çok değerli öznitelik** (multivalued attribute) denir. Örneğin Çalışan tablosunda çalışanların telefon bilgilerini saklamak için kullanılan Telefon özniteliginde çalışan ait birden fazla telefon bilgisi tutulursa çok değerli öznitelik olur. Varlık ilişki diyagramında çok değerli öznitelik çift oval şekil ile ifade edilir. Şekil 2.2'deki telefon özniteligi çok değerli olarak gösterilmiştir. Çok değerli ve birleşik öznitelikler farklı kavramlardır. Çok değerli öznitelikte bir varlık örneğine ait birden fazla veri tutulur. Birleşik öznitelikte ise bir varlığa ait bir değer tutulur. Örneğin Çalışan varlığına ait adres alanında her bir çalışan için sadece bir adres vardır ancak her adres alt özniteliklerin birleşiminden oluşur (Mahalle, Cadde, No, Posta_kodu, Şehir).

Çok değerli öznitelik (Multivalued attribute): Bir varlık veya ilişki örneğine ait öznitelikte birden fazla değer yer almazdır.

Anahtar Öznitelik

Varlık kümesindeki her bir örnek için farklı değer alan öznitelijke anahtar öznitelik denir. Yani anahtar öznitelikte iki örneğin değeri aynı olamaz. Anahtar öznitelik tanımlandığı varlık kümesinde bulunan tüm örnekler için benzersiz değerler alır. Örneğin Çalışan_no Çalışan varlık kümesi için anahtar özniteliktir. Vatandaşlık numaraları, öğrenci numaraları, hastane dosya numaraları, evrak numaraları, kütüphanedeki kitap numaraları her biri kendi varlık kümesi için birer anahtar özniteliktir. Anahtar öznitelik tek bir öznitelikten oluşabileceği gibi birden fazla özniteligin birleşimi ile de tanımlanabilir. Bu tip öznitelijke birleşik anahtar denir.

Etki Alanı

Bir özniteligin alabilecegi değerlerin tümünün oluşturduğu kümeye etki alanı (domain) denir. Örneğin bir şirkette Çalışan_Ayrıcalıkları varlık kümesinde bulunanların Ayrıcalık_no öznitelığının değerleri Yönetici için 1, İdari personel için 2, İşçi için 3 olsun. Bu durumda Ayrıcalık_no öznitelığının etki alanı 1, 2 ve 3 değerlerinden oluşur. Etki alanı varlık ilişki diyagramı içinde gösterilmez.

SIRA SIZDE

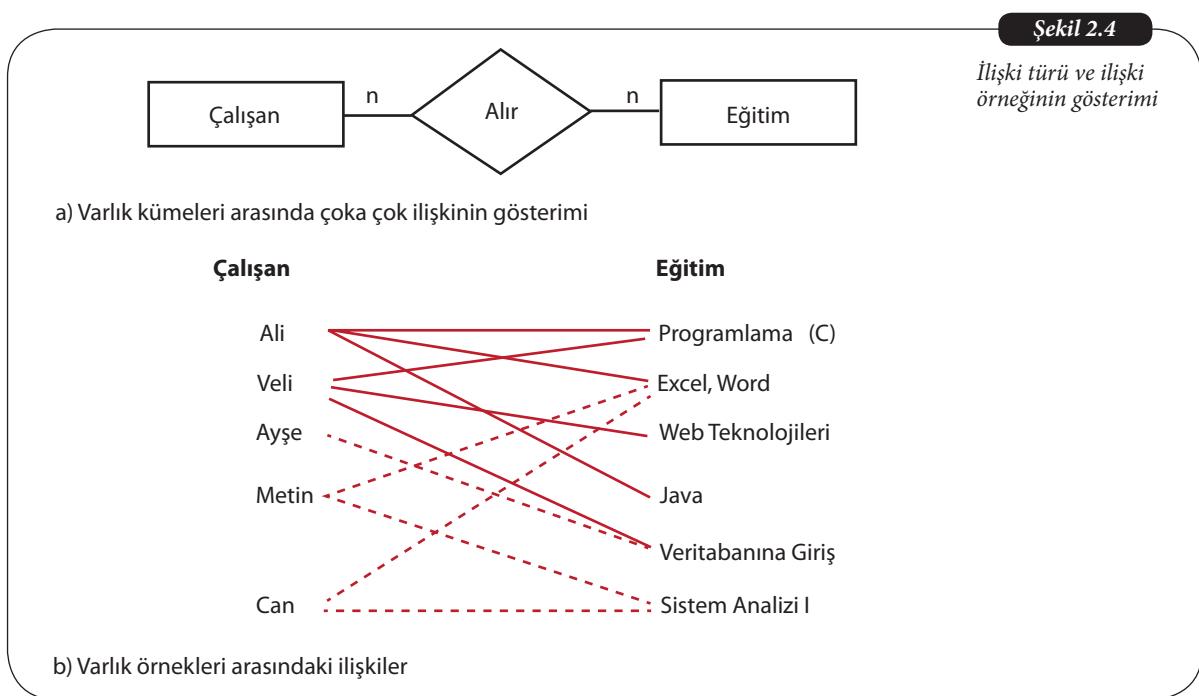


2

Anahtar özniteligin önemi nedir açıklayınız?

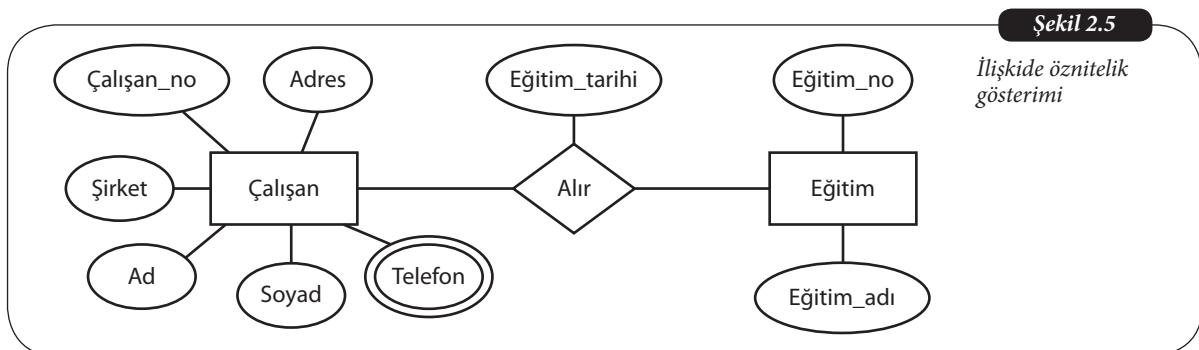
İlişki

İlişkiler veri modelinde farklı öğeleri bir arada tutar. En az iki varlık kümesi arasındaki etkileşimi gösteren bağlantıya ilişki (relationship) denir. İlişkiler iki varlık kümesi arasında olabileceği gibi çoklu şekilde de olabilmektedir. İlişkiler varlık ilişki diyagramında eşkenar dörtgen yani baklava dilimi şeklinde gösterilir ve içerisine ilişkinin adı yazılır. İlişkileri örneklemek üzere bir işletmede çalışanlar ve aldığı eğitimleri saklayan Çalışan ve Eğitim varlık kümeleri tanımlansın. Hangi çalışanın hangi eğitimi aldığıni izlemek üzere Çalışan ve Eğitim varlık kümeleri arasında Alır ilişkisi oluşturulur. Şekil 2.4 (a) şıkkında bu varlık kümeleri ve aralarındaki ilişki gösterilmektedir. Bu çokça çok ilişki türüdür. İlişki türleri izleyen kesimlerde detaylı olarak açıklanacaktır. Verilen çokça çok ilişkide, bir çalışan birden fazla eğitim alabilir ve aynı şekilde bir eğitim birden fazla çalışan tarafından alınabilir. Şekil 2.4 (b) şıkkında varlık örnekleri arasında bu ilişkinin detayı verilmiştir. (b) şıkkında görüldüğü gibi; Ali: Programlama (C), Excel, Word, Java eğitimleri alırken, Excel, Word eğitimini Ali, Metin ve Can almıştır. Benzer şekilde diğer çalışanların aldığı eğitimlerde gösterilmiştir.



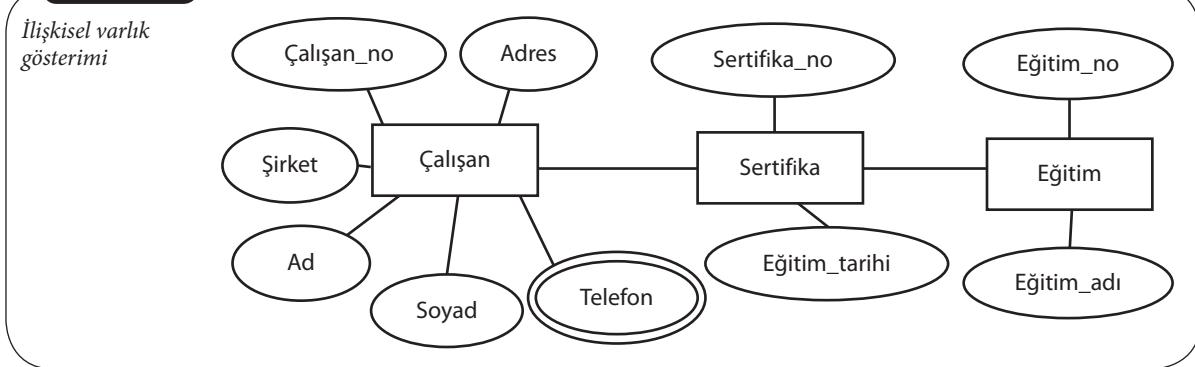
İlişki türü ve ilişki örneği farklı kavramlardır. Bir ilişki türü Şekil 2.4 (a) şıklıkta olduğu gibi varlık kümeleri arasındaki bağlantıyı gösterir. İlişki örneği ise varlık örnekleri arasındaki bağlantıdır. Her bir ilişki örneği bağlantılı olduğu varlık kümelerinden birer varlık arasında oluşur. Şekil 2.4 (b) şıklıkta verilen her bir çizgi farklı bir ilişki örneğini gösterir.

Şekil 2.4 (a) şıklıkta verilmiş olan ilişki Alır çalışanların almış olduğu eğitimleri ifade etmektedir. Çalışanların aldığı eğitimlerin tarihleri de saklanmak istenirse Alır ilişkisine Eğitim_tarihi özniteligi eklenir. İlişkilere eklenen öznitelikler de modelde oval içerisine özniteligin adı yazılarak gösterilir. İlişkiye eklenmiş Eğitim_tarihi öznitelik örneği Şekil 2.5'te verilmiştir.



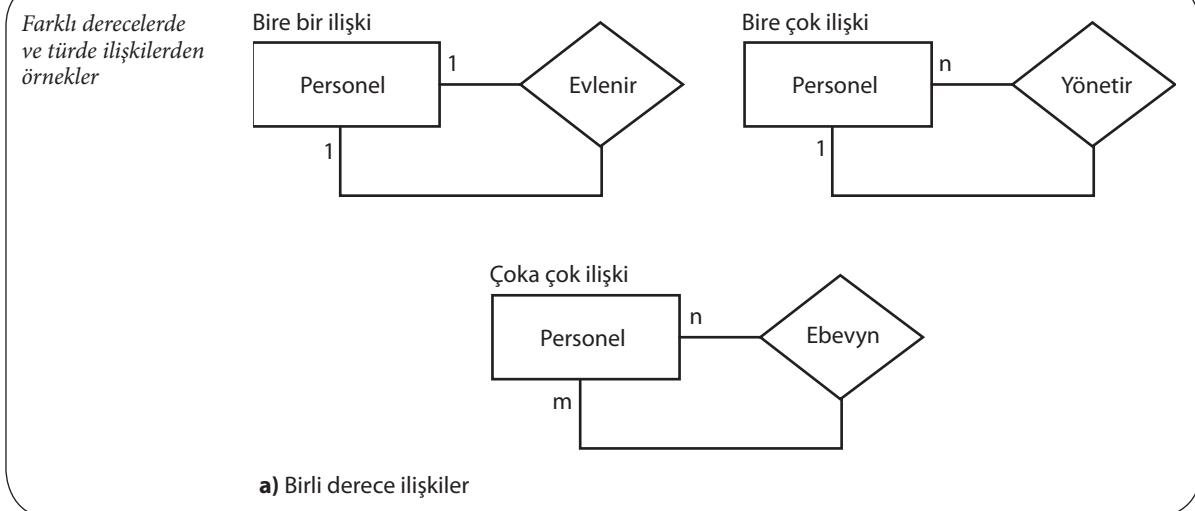
İlişkisel Varlıklar

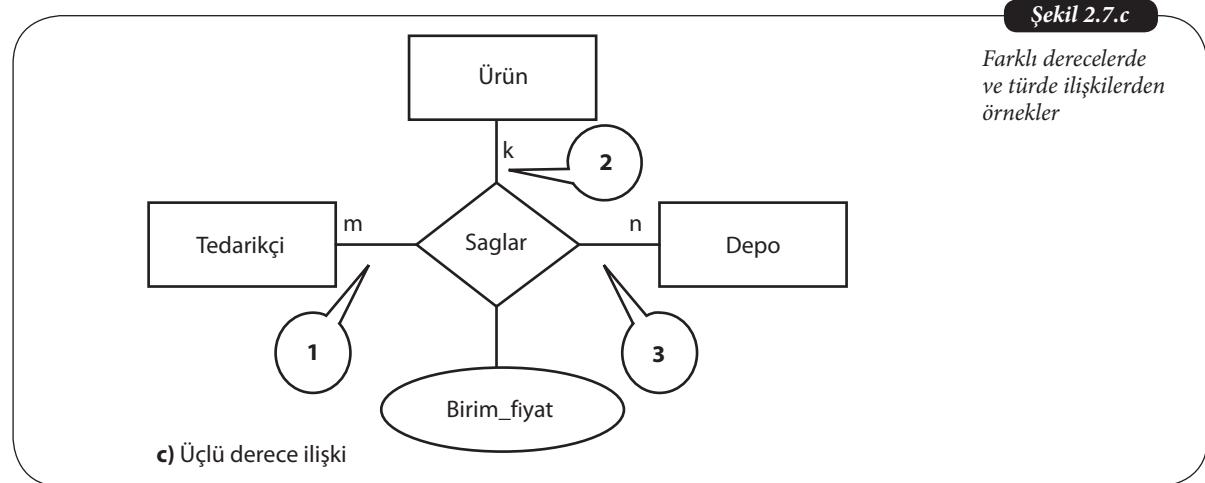
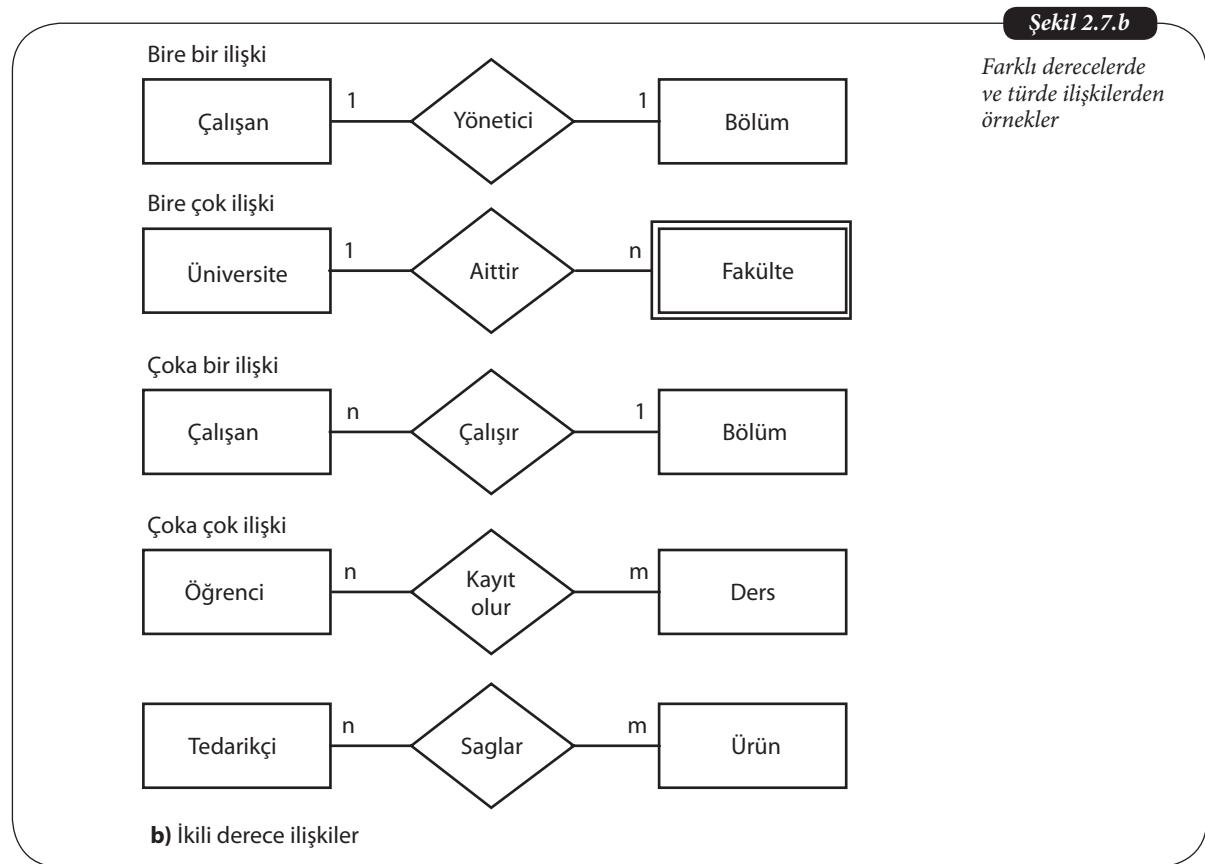
Bir ilişkinin özniteligi birden fazla ise bu ilişki bir varlık olarak gösterilebilir. Bu tip varlıklar ilişkisel varlık olarak adlandırılır. Şekil 2.5'te Çalışan ve Eğitim varlık kümeleri arasındaki bağlantı ilişki ve üzerinde özniteligi ile gösterilirken Şekil 2.6'da ilişkisi tanımlayan öznitelik sayısı arttılarından ilişkisel varlık gösterimi kullanılmıştır. Örnekte Sertifika ilişkisel varlığı oluşturulmuştur.

Şekil 2.6

İlişki Derecesi

İlişkide yer alan varlık kümelerinin sayısı ilişkinin derecesini gösterir. Şekil 2.7'de (a) birli, (b) ikili ve (c) üçlü derecede ilişkiler gösterilmiştir. Örnekte (a) şıkkında her bir modelde tek varlık vardır; Personel, Personel, Personel. (b) şıkkında verilen beş modelin her birinde ikişer varlık vardır; Çalışan-Bölüm, Üniversite-Fakülte, Çalışan-Bölüm, Öğrenci-Ders, Tedarikçi-Ürün. (c) şıkkında ise üç varlık arasında ilişki kurulmuştur; Tedarikçi-Ürün-Depo. Bu örnekte Sağlar ilişkisi Birim_fiyat özniteligi sahiptir. Üçten daha yüksek dereceli ilişkiler tanımlanabilmekle birlikte pratikte nadir olarak kullanılırlar. Bu nedenle üçüncü dereceden daha yüksek ilişkiler (3'ten fazla varlığı içeren ilişkiler) gösterilmemiştir.

Şekil 2.7.a



İlişki Türleri

Varlık kümeleri arasında üç tür ilişkiden söz edilebilir. Bunlar,

- Bire Bir İlişki (One to One 1:1)
- Bire Çok İlişki (One to Many 1:N) veya Çoka Bir İlişki (Many to One N:1)
- Çoka Çok İlişki (Many to Many N:N)dir.

Bire Bir İlişki (One to One 1:1): Bir varlık kümelerinin elemanı diğer bir varlık kümelerinin elemanlarından sadece biri ile ilişki kurabiliyorsa buna bire bir ilişki denir. Şekil 2.7'de değişik derecelerde bire bir ilişki örnekleri verilmiştir. (a) şıkkında Personelin

mevcut evlilik bilgisi tutulduğu varsayılmıştır. Eğer bir personel evli ise bu verinin saklandığı gösterilmiştir. (b) şıkkında ise ikinci dereceden ilişki olan Çalışan ve Bölüm varlıklarları arasında bire bir ilişki tanımlanmıştır. Bu örnekte bir çalışanın yönetici olduğu bölüm gösterilmiştir.

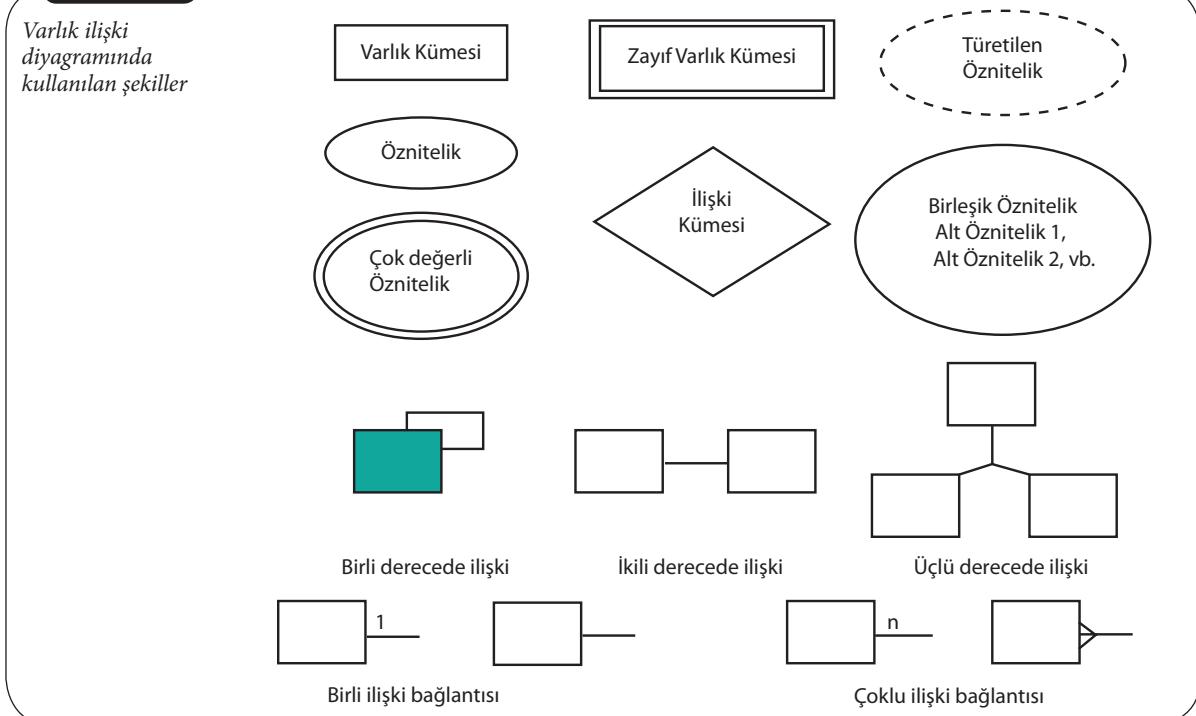
Bire Çok İlişki (One to Many 1:N) ve **Çoka bir ilişki (Many to one N:1)**: Bir varlık kümesinin elemanı diğer bir varlık kümesinin elemanlarıyla birden çok ilişki kurabiliyorsa buna bire çok ilişki denir. Bire çok ve çoka bir ilişki benzer ilişkiler olup kaynak varlık kümesinden bir varlık hedef varlık kümesinden birden fazla varlık ile ilişkilidir. Şekil 2.7 (a) ve (b) şıkklarında bire çok ve çoka bir ilişkilere örnekler verilmiştir. (a) şıkkındaki bire çok ilişki bir kişinin (yönetici) birden fazla personeli yönettiğini (veya birden fazla personel bir kişinin yönetiminde olabilir) gösterir. (b) şıkkında verilen bire çok ilişkide bir üniversitenin birden fazla fakültesi vardır. İkinci model de ise (çoka bir) bir bölümde birden fazla çalışan vardır.

Çoka Çok İlişki (Many to Many N:N): Bir varlık kümesinin birden fazla elemanı diğer bir varlık kümesinin elemanlarıyla birden fazla ilişki kurabiliyorsa buna çoka çok ilişki denir. Şekil 2.7 (c) şıkkında çoka çok ilişki örneğindeki her bir ilişki küçük cember içinde 1, 2 ve 3 olarak numaralanmıştır. 1 nolu ilişkide; her bir Tedarikçi birden fazla Ürünü birden fazla Depodan sağlanabilir, fakat herhangi bir Ürün sağlamak zorunda da değildir. 2 nolu ilişkide; her bir Ürün herhangi bir sayıda Tedarikçi tarafından birden fazla depodan sağlanabilir, fakat her bir Ürün en az bir Tedarikçi tarafından bir depodan sağlanabilmelidir. 3 nolu ilişkide; her bir Depodan herhangi bir sayıdaki Ürün birden fazla Tedarikçiden sağlanabilir, fakat her depodan en az bir Ürün sağlanabilmelidir.

Varlık İlişki Diyagramlarında Kullanılan Şekiller

Varlık ilişki diyagamlarında kullanılan işaretler Şekil 2.8'de verilmiştir. İşaretler için bir endüstriyel standart bulunmamaktadır. Gösterilen işaretler günümüzde kullanılan varlık ilişki modelleme diyagamlarında ortak kullanılan işaretlerdir. Pratikte karşılaşılan pek çok durumu modellemek için yeterli olabilecektir. Günümüzde kullanılan varlık ilişki modelleme araçlarında (Oracle Designer, Microsoft Visio vb.) bazı işaretler olmayabilir veya küçük farklar ile kullanılıyor olabilir.

Şekil 2.8



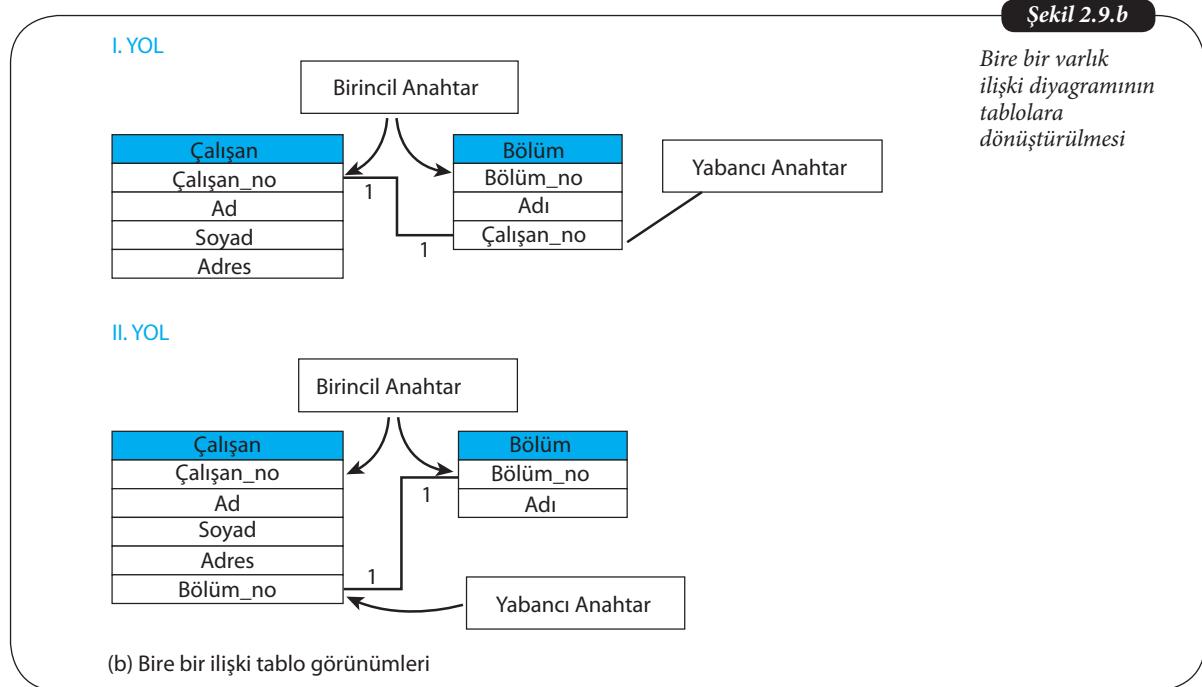
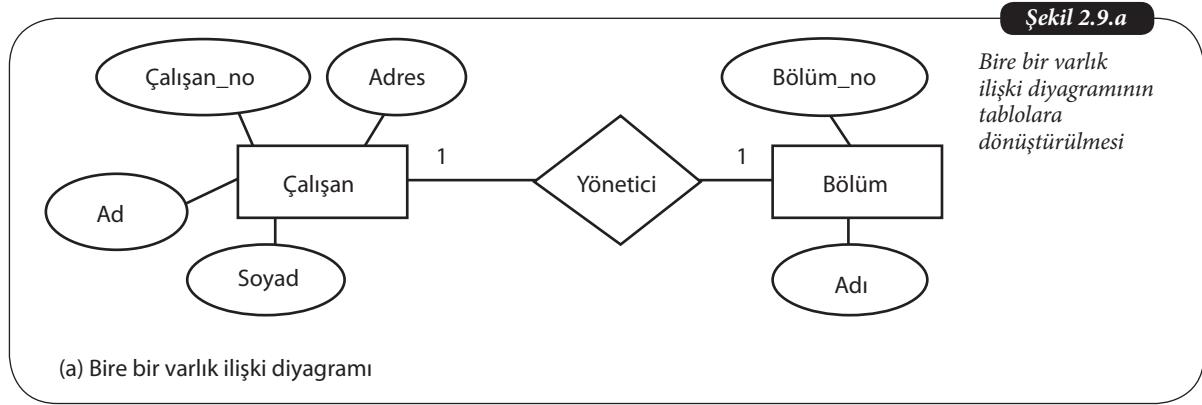
VARLIK İLİŞKİ DİYAGRAMININ TABLOLARA DÖNÜŞTÜRÜLMESİ

Varlık ilişki modelleri tablolara dönüştürülürken varlık kümeleri tablolara, öznitelikler ise alanlara dönüştürülür. Anahtar öznitelikler ise tabloda birincil anahtara dönüştürülür.

Bire Bir İlişkilerin Tablolara Dönüşürülmesi

Bire bir ilişkideki varlık kümeleri tablolara dönüştürülür. Öznitelikler ise tabloların sütunlarını oluşturur. İlişkileri oluşturmak için, bir varlık kümesinin anahtar özniteligi tabloda birincil anahtar olarak tanımlanır ve bu birincil anahtar diğer tabloda **yabancı anahtar** olarak belirlenir. İlişkideki anahtar özniteliklerin hangisinin birincil anahtar veya yabancı anahtar olacağına tabloların içereceği bilgilere göre karar verilir. Şekil 2.9 (a) şıklıkta çalışanlar ve yöneticiği bölümlerin bilgilerini tanımlayan varlık ilişki diyagramı verilmiştir. (b) şıklıkta Çalışan varlık kümesi Çalışan tablosuna Bölüm varlık kümesi Bölüm tablosuna dönüştürülmüştür. Anahtar öznitelikler **Çalışan_no** ve **Bölüm_no** kendi tablolarında birincil anahtar olarak tanımlanmıştır. Bu modelde iki tablo arasındaki ilişkiyi oluşturmak için iki yol kullanılabilir. Birinci yolda Çalışan tablosundaki **Çalışan_no** birincil anahtar Bölüm tablosuna yabancı anahtar olarak eklenmiştir. İkinci yolda **Bölüm_no** birincil anahtarı Çalışan tablosuna yabancı anahtar olarak eklenmiştir.

Yabancı anahtar: ilişkisel veri tabanlarında bir tablonun birinci anahtarının diğer bir tabloya daha kolay bağlanmak amacıyla ikinci tablo üzerinde bir sütun olarak yer almıştır.

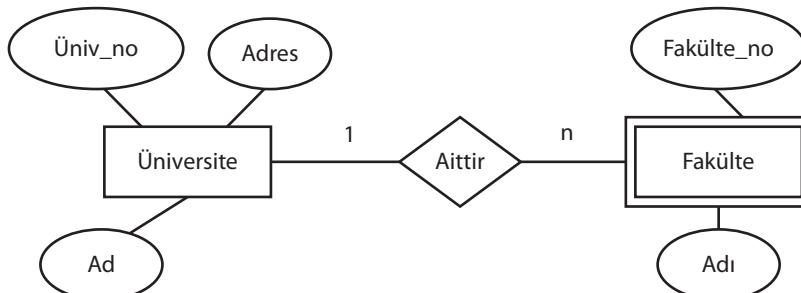


Bire Çok veya Çoka Bir İlişkilerin Tablolara Dönüşürlmesi

Varlık kümeleri tablolara dönüştürülür. Öz nitelikler tabloların sütunları olarak tanımlanır. İlişkinin birli (1) tarafındaki tablodaki öz nitelik birincil anahtar olarak tanımlanır ve ilişkinin çoklu (n) tarafındaki tabloya yabancı anahtar olarak eklenir. Şekil 2.10 (a) şekilde bir üniversite ve üniversiteye ait fakülteleri gösteren varlık ilişki modeli verilmiştir. Üniversiteye ait birden fazla fakülte vardır. Bu modelde Fakülte zayıf varlık kümesidir. Şekil 2.10 (b) şekilde gösterildiği gibi öncelikle varlık kümeleri ve öz nitelikleri tablolara dönüştürülür. İlişkinin birli tarafı olan Üniversite tablosunun birincil anahtarı (Üniv_no) ilişkinin çoklu tarafındaki Fakülte tablosuna yabancı anahtar olarak eklenir.

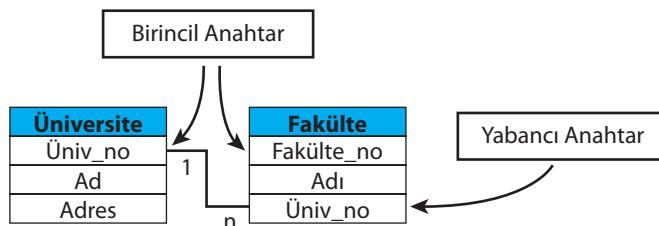
Şekil 2.10.a

Bire çok varlık ilişki diyagramı



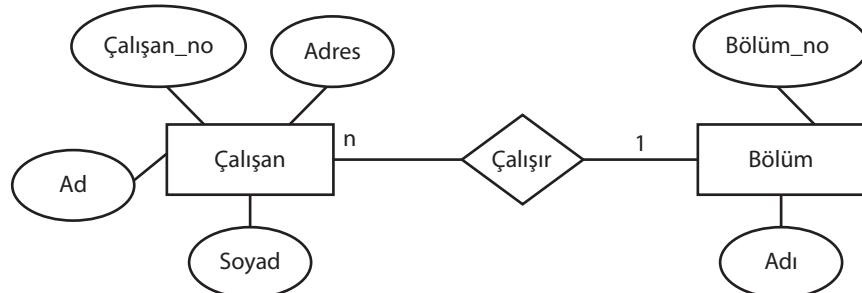
Şekil 2.10.b

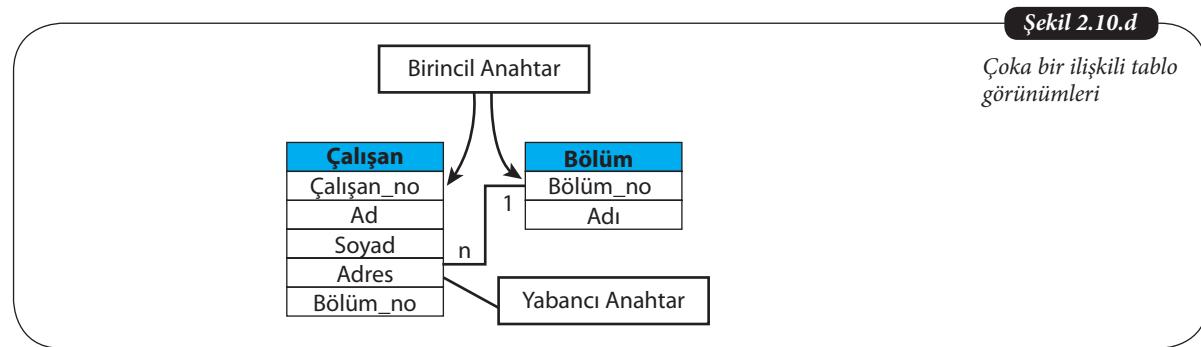
Bire çok ilişkili tablo görünümüleri



Şekil 2.10.c

Çoka bir varlık diyagramı





Şekil 2.10 (c) şıklıkta bir işyerindeki bölümler ve bu bölümlerde çalışanların ilişkisini gösteren çoka bir ilişki türü içeren varlık ilişki diyagramı verilmiştir. Her çalışanın bir bölümü vardır (veya her bölümde birden fazla çalışan vardır). Şekil 2.10 (d) şıklıkta Çalışan ve Bölüm varlık kümeleri ve öznitelikleri tablolara dönüştürülmüştür. İlişkinin birli tarafı olan Bölüm tablosunun birincil anahtarı (Bölüm_no) ilişkinin çoklu tarafındaki Çalışan tablosuna yabancı anahtar olarak eklenerek ilişki oluşturulmuştur.

Yabancı anahtarın önemi nedir açıklayınız?



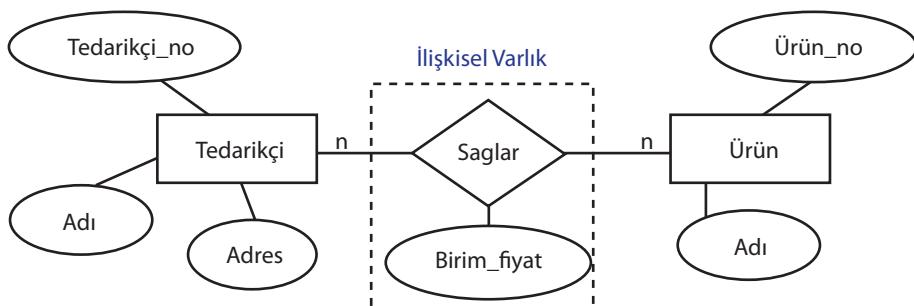
SIRA SİZDE

Çoka Çok İlişkilerin Tablolara Dönüşürtlmesi

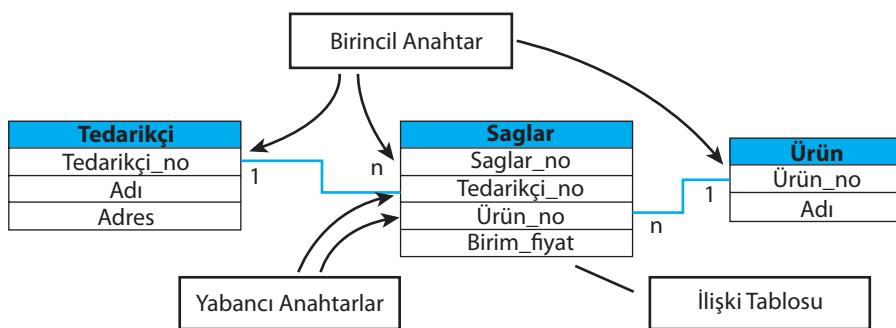
Varlık kümeleri tablolara dönüştürülür. Varlık kümeleri arasındaki ilişki (ilişkisel varlık) içinde tablo oluşturulur. Öznitelikler tabloların sütunları olarak tanımlanır. Her bir tablo da birincil anahtarlar tanımlanır. Varlık kümelerinin birincil anahtarları ilişkiye oluşturan tabloya sütun olarak eklenir. İlişkiye karşılık oluşturulan tablonun sütunları; kendi birincil anahtarlarından, diğer kümelerden gelen yabancı anahtarlardan ve tabloya ait özniteliklerden oluşur. Böylece tablolar arasında çoka çok ilişki, bire çoklu ve çoka birli ilişkilere dönüştürülmüş olur. Şekil 2.11 (a) şıklıkta çoka çok Saglar ilişkisi ile bağlı Tedarikçi ve Ürün varlık kümelerinden oluşan varlık ilişki modeli verilmiştir. Bu modele göre bir tedarikçi birden fazla ürün sağlar. İlave olarak bir ürün birden fazla tedarikçiden sağlanabilir. Çoka çok ilişkisi dönüştürme için Saglar ilişki tablosu eklenmiş ve ilişkisel varlık olarak işaretlenen öznitelikler bu tabloya eklenmiştir. İlave olarak bu tabloya Tedarikçi ve Ürün tablolarının birincil anahtarları yabancı anahtarlar olarak eklenmiştir. Saglar tablosuna tanımlanan Saglar_no birincil anahtarının eklenmesi zorunlu değildir. Bununla birlikte kullanılan her tabloya birincil anahtar eklemek iyi bir yaklaşımdır. Bu nedenle ilave olarak Saglar ilişki tablosuna Saglar_no birincil anahtarları tanımlanmıştır.

Şekil 2.11

Çoka çok varlık ilişkisi diyagramının tablolara dönüştürülmesi



(a) Çoka çok varlık ilişkisi diyagramı

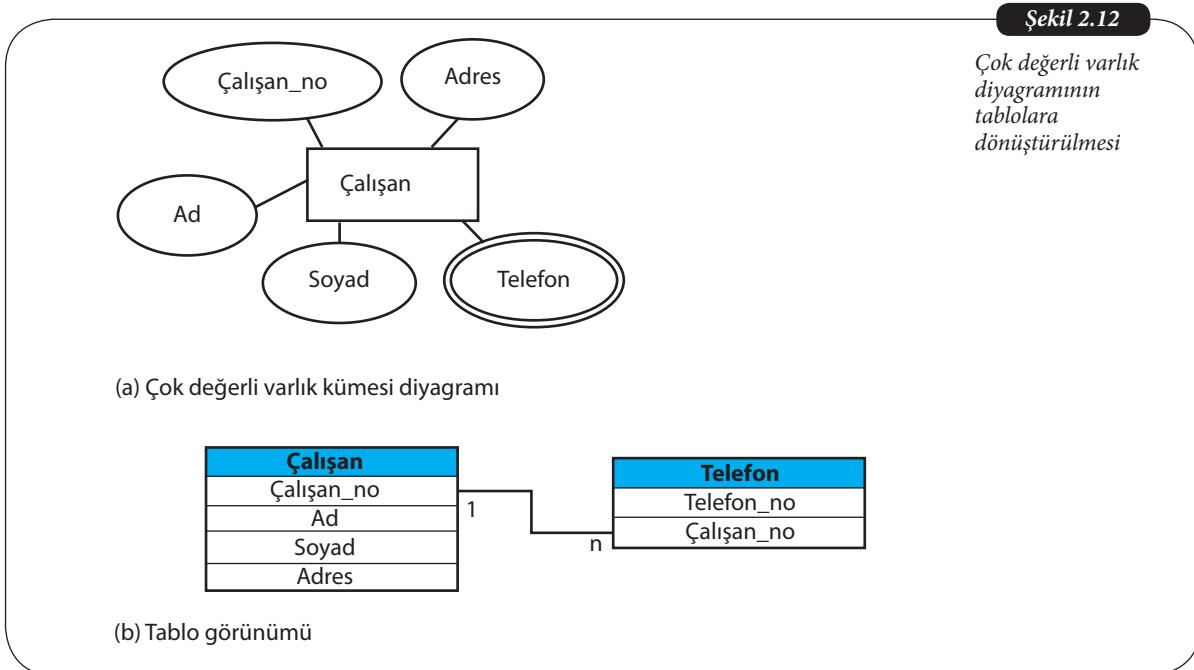


(b) Çoka çok ilişkinin dönüştürüldüğü tablo görüntümeleri

Çok Değerli Öz niteliklerin Tablolara Dönüştürülmesi

Varlık kümesi ve öznitelikleri tabloya dönüştürülür. Çok değer içeren öznitelik için yeni bir tablo oluşturulur. Bu tabloya çok değerli özniteligiin bağlı olduğu tablonun birincil anahtarı yabancı anahtar olarak eklenir. Bu sayede çok değerli özniteligiin her bir alt değeri yeni tabloya ayrı satır olarak eklenmiş olur. Şekil 2.12'de çok değerli Telefon özniteligi olan Çalışan varlık kümescinin diyagramı ve tablolara dönüştürülmüş şeması verilmiştir. Çok değerli öznitelik olan Telefon alanında çalışanların herbiri için 1 veya daha fazla telefon numarası kaydı mevcuttur. Bazı çalışanlarda telefon numarası kaydı olmayabilir. Bu dönüşüm ile bir çalışana ait her bir telefon kaydı yeni tabloda ayrı satırda saklanmış olacaktır.

Şekil 2.12



VERİTABANI YAŞAM DÖNGÜSÜ

Veritabanı projelerinin asıl kaynaklarından birisi bilgi sistemlerinin geliştirilmesidir. İşletmenin ihtiyaçlarını karşılayan bilgi sistemleri yeni veritabanı geliştirilmesine ihtiyaç duyar. Bilgi sistemi kullanıcıları yapacakları iş ile ilgili verileri depolamak istediginde veya bilgi sistemi uzmanları veri yönetimini iyileştirmek istediginde veritabanı kullanımlı çözüm olacaktır. Veritabanı geliştirme süreci bire bir olmasa bile bilgi sistemi geliştirme süreci ile büyük ölçüde örtüşür. Bu nedenle izleyen kesimde öncelikle bilgi sistemi geliştirme süreci fazları kısaca açıklanacaktır. Daha sonra da bilgi sistemi geliştirme fazlarıyla büyük ölçüde örtüşen veritabanı geliştirme yaşam döngüsü (VTYD) fazları hakkında bilgi verilecektir.

Geleneksel bilgi sistemi geliştirme süreci sistem geliştirme yaşam döngüsü (SGYD) olarak adlandırılır. Sistem geliştirme yaşam döngüsü bilgi sistemlerinin planlanması, geliştirilmesi, kullanım ve bakım faaliyetlerini içeren fazlardan oluşur. Bu fazlar;

Kapsam ve Planlama: Problemin kapsamı, fırsatları ve hedefleri tanımlanır.

Gereksinim Analizi: Sistem analisti, sosyal araç ve tekniklerin yardımıyla gereksinimleri tanımlar. Bu gereksinimler analiz edilerek işlevsel, işlevsel olmayan ve ortam gereksinimleri olarak sınıflanır. Bu fazın çıktısı alternatif çözüm önerisidir.

Tasarım: Sistem tasarımda analist, alternatif çözüm önerilerini kavramsal, mantıksal ve fiziksel sistem özelliklerine dönüştürür.

Gerçekleştirme ve Test: Özellikleri ortaya konulmuş olan sistem geliştirilir ve testleri tamamlanarak çalışır hale getirilir.

Kurulum ve Dağıtım: Sistem geliştiricileri, uygulama yazılımını (ve veritabanını), geliştirme ortamından üretim ortamına geçirmek üzere mevcut ya da yeni donanım üzerine kurarlar. Sistem analistleri sistem kullanıcılarını eğitir, eski verilerin yeni sisteme aktarımını ve en son sistem testlerini yapar.

Operasyon ve Bakım: Kullanıcıların talepleri ya da iş süreçlerindeki değişimler doğrultusunda sistem desteğinin sağlanmasıdır.

Veritabanı Yaşam Döngüsü Fazları

Veritabanı yaşam döngüsü fazları verilmiş olan SGYD fazları ile büyük ölçüde örtüşür. VTYD fazları;

1. Gereksinim Analizi
2. Veritabanı Tasarımı
3. Uygulama ve Yükleme
4. Operasyon
5. Bakımdır.

Gereksinim Analizi

Bu fazda yapılan işlemler;

- Mevcut veri işleme süreçlerinin analizi,
- Genel işletme işlevleri ve onların veritabanı ihtiyaçlarının analizi,
- İşletmede kullanılan bilgi sisteminin yeni veritabanına ihtiyacının ortaya konulması,
- Önerilen bilgi sistemi için gereken veritabanı kapsamının tanımlanması,
- İşletmedeki işlevlerin veritabanı destegine ihtiyaç duyduğu tüm veri gereksinimlerinin analizidir.

Veritabanı Tasarımı

Tasarım fazında kavramsal, mantıksal ve fiziksel veri modelleri oluşturulur. Bu modeller ve özellikleri hakkında açıklamalar ünitenin ilk kesiminde verilmiştir.

Uygulama ve Yükleme

Bilgi sisteminin bir parçası olarak veritabanı uygulama ekiplerince geliştirilir ve çalışacağı hedef sisteme yüklenir. Bu işlemler;

- Veritabanı işleme programlarının geliştirilmesi ve test edilmesi,
- Veritabanı dökümantasyon ve eğitim materyallerinin oluşturulması,
- Veritabanı yönetim sisteminin kurulması ve mevcut verilerin dönüştürülerek veritabanına aktarılması,
- Kullanıma alınan veritabanının uygulama yazılımları ile birlikte çalışmasının test edilmesinden oluşur.

Operasyon

Bu fazda bilgi sistemi ve geliştirilen veritabanı uygulamaya alınmış ve kullanılmıştır. Veritabanı kullanıcılar, uygulama yazılımı ve donanım ile ortaklaşa çalışmaktadır. Olası bir hata durumda tüm sistemin kullanılamaz olma ihtimali yüksektir. Bu nedenle veritabanı kullanımı süresince izlenmelidir.

Bakım

Bilgi sistemi kullanıma alındıktan sonra kullanıcılar tarafından yeni isterlerin gerçekleştirilemesi ve iyileştirmeler talep edilir. Bu değişiklik isteklerinin uygulamaya alınması için veri modelindeki ve dolayısıyla veritabanındaki değişikliklerin yapılması gerekecektir. Ayrıca veritabanı ve uygulamaların kullanımı süresince bilgi ihtiyaçlarını karşılayıp karşılamadığı da izlenmeli ve gerekirse düzenlemeler yapılmalıdır. Veritabanında saklanılan veriler artıkça veritabanı performans için uyarlanmalıdır. Zaman içinde ortaya çıkan hatalardan dolayı oluşabilecek bozuk veya kalıntı veriler düzenlenmeli ve gerekirse kaldırılmalıdır.

Özet



Veritabanı tasarım aşamalarını tanımlamak

Veritabanında kullanılan tasarımlar üç aşamadan meydana gelmektedir.

Kavramsal veri modelleme: Kavramsal veri modeli (conceptual data model) veritabanında saklanacak verilerin, kısıtlarının ve ilişkilerinin bir gösterimidir.
Mantıksal veri modelleme: Mantıksal veri modeli (logical data model) Kavramsal modelin seçilen veritabanı yönetim sisteminin veritabanı şemasına dönüştürülmesidir.

Fiziksel veri modelleme: Fiziksel veri modeli (physical data model) veri süreklilik teknolojisinin belirli bir süremü ile bağlı, tamamiyla bağlanmış veri modelidir.



Varlık ilişki modelleme kavramlarını ve varlıklar arasındaki ilişkileri sıralamak

Varlık ilişki modelleme kavramsal veri modelinde kullanılan yöntemlerden biridir. Varlık ilişki modelleme ilk olarak Peter Chan tarafından 1976 yılında bulunmuştur. Varlık ilişki modeli ilişkisel veritabanı yönetim sistemleri için tablo ve tablolar arası ilişkiyi ortaya çıkarır böylece veritabanı tasarımının temel yapısını oluşturmuş olur. Varlık ilişki modelinde bire bir, bire çok (veya çok bir) ve çok çok ilişki türleri vardır.



Varlık ilişki modelini tablolara dönüştürmek

Varlık ilişki modelleri tablolara dönüştürülürken varlık kümeleri tablolara, öznitelikler ise tabloların alanlarına dönüştürülür. İlişkide ise bir varlık kümесinin anahtar özniteliği tabloda birincil anahtar olarak tanımlanır. İlişkideki anahtar özniteliklerin hangisinin birincil anahtar veya yabancı anahtar olacağına tablolardan içereceği bilgilere göre karar verilir. İlişki türüne göre bu birincil anahtar diğer tablolara yabancı anahatlar olarak eklenir. Bazı ilişki türlerinin dönüşümü ise ilişki tablosu oluşturularak sağlanır.



Veritabanı yaşam döngüsünü aşamalarını sıralamak

Veritabanı yaşam döngüsü beş aşamadan meydana gelmektedir.

- Gereksinim Analizi
- Veritabanı Tasarımı
- Uygulama ve Yükleme
- Operasyon
- Bakım

Kendimizi Sınayalım

1. Varlık ilişki diyagramının tasarılanması aşağıdaki veri modellerinden hangisi içinde yer alır?

- a. İç model
- b. Dış model
- c. Mantıksal model
- d. Kavramsal model
- e. Fiziksnel model

2. Aşağıdakilerden hangisi veritabanında yer alacak olan tablolar ve tabloların birbirlerine olan bağlantılarının özeliliklerini ortaya çıkarır?

- a. Varlık
- b. Etki alanı
- c. İlişki
- d. Öznitelik
- e. Varlık ilişki diyagramı

3. Aşağıdaki özniteliklerden hangisi bulunduğu varlık kümesi için benzersiz değerler alır?

- a. Türetilen öznitelik
- b. Çok değerli öznitelik
- c. Tek değerli öznitelik
- d. Birleşik öznitelik
- e. Anahtar öznitelik

4. “Doğum tarihi özniteliği ve günün tarihi kullanılarak yaş özniteliği elde edilebilir.” açıklaması ile bahsedilen öznitelik aşağıdakilerden hangisidir?

- a. Anahtar öznitelik
- b. Türetilen öznitelik
- c. Birleşik öznitelik
- d. Tek değerli öznitelik
- e. Çok değerli öznitelik

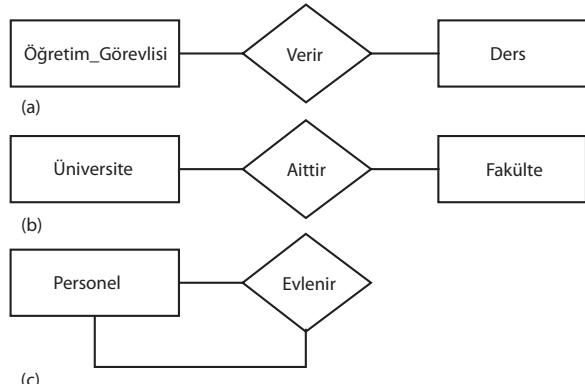
5. Aşağılardan hangisi veritabanı yaşam döngüsü aşamalarından biri **değildir**?

- a. Veritabanı tasarıımı
- b. Gereksinim Analizi
- c. Operasyon
- d. Uygulama ve yükleme
- e. Veritabanı oluşturma

6. Aşağıdaki veri modellerinin hangisinde yabancı anahtar yer alır?

- a. Mantıksal model - Fiziksnel model
- b. Kavramsal model - Fiziksnel model
- c. Mantıksal model - Kavramsal model
- d. Sadece Mantıksal model
- e. Sadece Kavramsal model

7, 8, 9 ve 10. soruları Şekil 2.13'e göre cevaplayınız.



Şekil 2.13. Varlık ilişki modelleri

7. Şekil 2.13 (a) şıkkında verilen varlık ilişki modelinde bir öğretim görevlisine birden çok ders verilme durumunda olacak ilişki türü aşağıdakilerden hangisidir? (Her ders sadece bir öğretim görevlisince verilir)

- a. Bire bir ilişki(1:1)
- b. Çoka bir ilişki(N:1)
- c. Bire çok ilişki(1:N)
- d. Çoka çok ilişki(N:N)
- e. Tekil ilişki(1)

8. Şekil 2.13 (c) şıkkında verilen Personel varlık kümesi tablolara dönüştürüldüğünde kaç tablo oluşur?

- a. 1 tablo
- b. 2 tablo
- c. 3 tablo
- d. Dönüşüm yapılamaz
- e. Hiç tablo oluşmaz

9. Şekil 2.13'te verilen varlık ilişki modellerinde olan zayıf varlık kümesi aşağıdakilerden hangisidir?

- a. Öğretim_Görevlisi
- b. Ders
- c. Üniversite
- d. Fakülte
- e. Personel

10. Şekil 2.13 (c) şıkkında verilen varlık ilişki modelinde her personelin mevcut evlilik verisi tutuluyorsa ilişki türü nedir?

- a. Bire bir ilişki(1:1)
- b. Çoka bir ilişki(N:1)
- c. Bire çok ilişki(1:N)
- d. Çoka çok ilişki(N:N)
- e. Tekil ilişki(1)

Kendimizi Sınayalım Yanıt Anahtarı

- | | |
|-------|--|
| 1. d | Yanıtınız yanlış ise “Veritabanı Tasarım Aşamaları” konusunu yeniden gözden geçiriniz. |
| 2. e | Yanıtınız yanlış ise “Varlık İlişki Modelleme” konusunu yeniden gözden geçiriniz. |
| 3. e | Yanıtınız yanlış ise “Varlık İlişki Modelleme” konusunu yeniden gözden geçiriniz. |
| 4. b | Yanıtınız yanlış ise “Varlık İlişki Modelleme” konusunu yeniden gözden geçiriniz. |
| 5. e | Yanıtınız yanlış ise “Veritabanı Yaşam Döngüsü” konusunu yeniden gözden geçiriniz. |
| 6. a | Yanıtınız yanlış ise “Veritabanı Tasarım Aşamaları” konusunu yeniden gözden geçiriniz. |
| 7. c | Yanıtınız yanlış ise “Varlık İlişki Modelleme” konusunu yeniden gözden geçiriniz. |
| 8. b | Yanıtınız yanlış ise “Varlık İlişki Diyagramının Tablolara Dönüşürülmesi” konusunu yeniden gözden geçiriniz. |
| 9. d | Yanıtınız yanlış ise “Varlık İlişki Modelleme” konusunu yeniden gözden geçiriniz. |
| 10. a | Yanıtınız yanlış ise “Varlık İlişki Modelleme” konusunu yeniden gözden geçiriniz. |

Sıra Sizde Yanıt Anahtarı

Sıra Sizde 1

Varlık ilişki modellemesinin amacı veritabanında bulunan tabloları, tabloların birileri ile olan bağlantılarını ve özelliklerini ortaya çıkarmak için varlık ilişki diyagramları oluşturmaktr.

Sıra Sizde 2

Anahtar öznitelik bir varlık kümelerindeki her varlık için benzersiz değer olan niteliktir. Anahtar öznitelik benzersiz değer olması nedeniyle varlık kümelerindeki her bir varlığın göstergesi olur. Buna en iyi örnek Türkiye Cumhuriyeti vatandaşlık numarası olacaktır. Vatandaşlık numarası her birey için benzersiz değer alır.

Sıra Sizde 3

Yabancı anahtar, ilişkisel veritabanı yönetim sistemlerinde birden fazla tabloyu birbirine bağlamakta kullanılmaktır. İlişki türüne göre bir tablonun benzersiz anahtarı diğer tabloya yabancı anahtar olarak eklenmektedir. Böylece iki tablo arasında ilişki kurulmuş olur.

Sıra Sizde 4

Veritabanı tasarımı üç aşamadan meydana gelmektedir. Bu- lar kavramsal model, mantıksal model ve fiziksel model olarak adlandırılırlar.

Yararlanılan ve Başvurulabilecek Kaynaklar

Chen, P. (1976). **The Entity-Relationship Model - Towards a Unified View of Data**, ACM Transactions on Database Systems Vol. 1.

Garcia-Molina H., Ullman J.D., Widom J. (2008). **Database Systems: The Complete Book**, 2nd Edition. New Jersey, Pearson Education Inc.

Hoffer J.A., Prescott M.B., Mcfadden F.R. (2007). **Modern Database Management**, 8th Edition. New Jersey, Pearson Education Inc.

Özseven, T. (2011). **Veritabanı Yönetim Sistemleri**, 2. Baskı. Trabzon, MuratHan Yayınevi.

3

Amaçlarımız

- Bu üniteyi tamamladıktan sonra;
- 🕒 İlişkisel veritabanı modelini tanımlayabilecek,
 - 🕒 İlişkisel cebirin temel işlemlerini kullanabilecek,
 - 🕒 Veritabanı nesnelerinden tablo, kısıtlar, görünüm, indeks ve saklı yordamları tanımlayabilecek,
 - 🕒 Veri tiplerini ve kullanımını anlayabilecek bilgi ve becerilere sahip olacaksınız.

Anahtar Kavramlar

- İlişkisel Veritabanı Modeli
- İlişkisel Cebir
- Veri Tabanı Nesneleri: Tablo, İndeks, Görünüm, Saklı Yordam
- Veritabanı Kısıtları ve Anahtarlar

İçindekiler

Veritabanı Sistemleri

İlişkisel Veritabanı Modeli

- GİRİŞ
- İLİŞKİSEL CEBİR
- İLİŞKİSEL VERİTABANI NESNELERİ VE KAVRAMLAR
- İLİŞKİSEL VERİTABANI ÖRNEĞİ

İlişkisel Veritabanı Modeli

GİRİŞ

İlişkisel veritabanı modeli IBM'de araştırmacı olarak çalışan Dr. E. F. Codd tarafından geliştirilmiştir. Matematik alanında uzmanlığını yapmış olan Dr. Codd ilişkisel veri modeli çalışmasını “Büyük Paylaşımlı Veribankaları için Verinin İlişkisel Modeli” adıyla Haziran 1970 yılında yayımlamıştır. Daha sonra diğer araştırmacılarında katkılarıyla günümüzde kullanılmakta olan ilişkisel veritabanı modeli ortaya çıkmıştır.

İlişkisel veritabanı veriyi **ilişkisel örnekler** içinde depolar. İlişkisel örnekler veritabanı kullanıcıları tarafından tablolar olarak adlandırılır. Her bir ilişkisel örnek kayıtlar ve alanlar şeklinde oluşturulur. Bu ünitede veritabanı nesneleri anlatılırken terminoloji olarak tablo, kayıt ve alan isimlendirmeleri kullanılacaktır.

Bir tablo içindeki kayıtların veya alanların fiziksels sırasının önemi yoktur. Tablo içindeki her bir kayıt kendi değerini taşıyan alanlardan oluşur. İlişkisel veritabanlarının bu iki özelliği sayesinde veri, bilgisayardaki fiziksels depolamadan bağımsız olarak kullanılabilir maktedir. Bu sayede kullanıcı fiziksels olarak nerede ve nasıl depolandığını bilmesine gerek kalmadan veriye erişebilmektedir.

İlişkisel modelde tablolar arası ilişkiler bire-bir, bire-çok ve çoka-çok olarak sınıflanır (Tablolar arasındaki ilişkiler ünite 2'de detaylı olarak anlatılmıştır). İki tablo arasındaki bir ilişki bir ortak alanın eşleşen değerleri üzerinden direk olarak kurulur. Örnek olarak, Şekil 3.1'de [*Siparişler*] ve [*Sipariş Ayrıntıları*] tabloları [*Sipariş No*] alanı üzerinden ilişkilendirilmiştir. Böylece satış işlemlerinin detaylarına [*Sipariş No*] bağlantısı ile [*Sipariş Ayrıntıları*] tablosundan erişilebilir. Benzer şekilde [*Siparişler*] ve [*Müşteriler*] tabloları, [*Sipariş Ayrıntıları*] ve [*Ürünler*] tabloları da ilgili alanlar üzerinden ilişkilendirilmiştir.

Veritabanı için **İlişkisel örnek** (Relational instance): Tablo olarak, Tuple: Kayıt (Record) ve Satır (Row), Alan (Fields): Sütun (Columns) ve Öz nitelik (Attributes) olarak adlandırılır.

Tabloların birbiri ile ilişkisini sağlayan alanların ismi aynı olmak zorunda değildir. Veritabanı tasarım sırasında bu isimlere karar verilir. Örneğin Şekil 3.1'de [*Siparişler*] ve [*Müşteriler*] tabloları [*Siparişler*].[*Müşteri No*]= [*Müşteriler*].[*No*] şeklinde ilişkilendirilmiştir.



DİKKAT

Sekil 3.1

Kullanıcı veritabanındaki tablolar arasındaki ilişkileri biliyorsa veriye kolayca erişebilir. Veriye içinde bulunduğu tablolardan doğrudan ya da ilişkili tablolardan dolaylı olarak erişilebilir. Şekil 3.1'de verildiği gibi [*Müşteriler*] tablosu [*Ürünler*] tablosu ile dolaylı olarak ilişkili olsa da kullanıcı bir müşteriye satılan ürünlerin listesini üretебilir. Bu örnekteki dolaylı bağlantı; [*Müşteriler*] tablosu [*Siparişler*] tablosuyla, [*Siparişler*] tablosu [*Sipariş Ayrintilari*] tablosuyla ve [*Sipariş Ayrintilari*] tablosu da [*Ürünler*] tablosu ile doğrudan ilişkili olduğu için oluşturulabilmiştir.

İlişkisel veritabanında veriye Yapılandırılmış Sorgu Dili (Structured Query Language, SQL) ile erişilebilir. SQL, ilişkisel veritabanı oluşturma, düzenleneme, bakım ve sorgulama işlemlerinin yapılabildiği standart bir dildir. İzleyen ünitelerde SQL dilinin komutları detaylı olarak anlatılacaktır.

Günümüzde yaygın kullanılan veritabanı yazılımlarında SQL komutları komut ekranı veya grafik sorgu oluşturma ekranları üzerinden kullanılılmaktedir. Örneğin bir İVTYS (İlişkisel Veritabanı Yönetim Sistemi) olan Microsoft Access ile tablolar ve tablolar arasındaki ilişkiler grafik ekranlarda görülebilir. SQL sorguları da yine grafik ekranlar ile oluşturulabilir. Oluşturulan SQL komutları daha sonra kullanmak üzere saklanabilir.

SIRA SİZDE



1

Şekil 3.1'de verilen tablolara göre hangi ürünlerin hangi şehirlere sevk edildiğine ulaşmak için tablolara arasında hangi bağlantılar kurulmalıdır?

İLİŞKİSEL CEBİR

İlişkisel veritabanı modeli matematiğin iki dalı olan Küme teorisi ve birinci derece yüklem aritmetiğine dayanmaktadır. Modelin adı bile küme teorisindeki ilişki teriminden türetilmiştir. İlişkisel örnekler (tablolar) üzerine yapılacak işlemleri öğrenmeden önce verilen ilişkisel örneklerden yeni ilişkisel örnekler oluşturulmasını sağlayan ilişkisel cebir hakkında bilgi sahibi olunmalıdır. İlişkisel cebirde tüm işlenen veriler ve sorguların sonuçları kümeler hâlinde ifade edilir. Geleneksel ilişkisel cebirde işlemler dört sınıfta toplanabilir.

- Tablolara uygulanan genel küme işlemleri: birleşim (\cup), kesişim (\cap), farklılık (\setminus) ve fark ($-$) işlemleridir.

- b) Bir tablodan parçalar getiren işlemler: seçim (selection, σ) bazı kayıtları elerken, yansıtma (projection, π) bazı alanları eler.
- c) İki tablonun kayıtlarını bitişiren işlemler: Kartezyen çarpım (Cross-product, x) birinci kümenin her satırı ikinci kümenin her bir satırıyla eşleşir. Şartlı Bitişme (Conditional Join, \bowtie_θ) iki tablonun kartezyen çarpımı sonucundan verilen şartla uygun kayıtlar getirilir. Kartezyen çarpımın getirdiğinden daha az çoklu kayıt içermesi daha verimli bir biçimde hesaplamayı sağlar. Gösterimde şart yerine θ sembolü kullanıldığı için Teta (Theta, θ) bitişmesi diye de adlandırılır. Doğal bitişme (Natural Join, \bowtie) her ortak alanda eşit bitişme yapılarak bulunur. Bölme (Division, \div) işlemi; eğer B tablosundaki y kümesi A tablosundaki bir x ile ilişkilendirilmişse ve A tablosundaki x ile ilişkili bir şekilde B tablosundaki tüm y'leri kapsiyorsa, bu x A-B kümesi içerisindeidir.
- d) Yeniden adlandırma (Renaming, ρ) tablolardaki kayıtları etkilemez ancak erişilen alanların veya tabloların adlarını kullanım sırasında değiştirir.

Izleyen kesimde İşlemler Şekil 3.2'de verilen R ve S tablolarındaki veriler kullanılarak örneklenenecektir.

Şekil 3.2

Örnek Tablolar R
ve S

Tablo R

İsim	Ünvanı	Adres	Cinsiyet	Şehir	Ülke
Ayşe Kara	Marketing Manager	Kirmizi toprak mh. Gul sk. No:1	F	Eskisehir	Turkiye
Ali Can	Sales Manager	Seker Mh. 4883 sk No:9	M	Eskisehir	Turkiye

Tablo S

İsim	Ünvanı	Adres	Cinsiyet	Şehir	Ülke
John Smith	Marketing Manager	1900 Duh Drv.	M	Voncouver	Canada
Ayşe Kara	Marketing Manager	Kirmizi toprak mh. Gul sk. No:1	F	Eskisehir	Turkiye

Birleşim (\cup)

Birleşim işlemi iki tablodaki tüm kayıtları getirir. Tekrar eden kayıtlar elenir. $R \cup S$ olarak gösterilir ve Şekil 3.2'deki tablolar kullanıldığında $R \cup S$ 'nin getirdiği kayıtlar;

İsim	Ünvanı	Adres	Cinsiyet	Şehir	Ülke
Ayşe Kara	Marketing Manager	Kirmizi toprak mh. Gul sk. No:1	F	Eskisehir	Turkiye
Ali Can	Sales Manager	Seker Mh. 4883 sk No:9	M	Eskisehir	Turkiye
John Smith	Marketing Manager	1900 Duh Drv.	M	Voncouver	Canada

Kesişim (\cap)

Kesişim işlemi iki tablonun ortak kayıtlarını getirir. $R \cap S$ ile gösterilir. $R \cap S$ 'nin getirdiği kayıtlar;

İsim	Ünvanı	Adres	Cinsiyet	Şehir	Ülke
Ayşe Kara	Marketing Manager	Kirmizi toprak mh. Gul sk. No:1	F	Eskisehir	Turkiye

Fark (-)

Bu işlemde iki tablodan birincisinde olan ancak ikincisinde olmayan kayıtlar getirilir. $R - S$ ile gösterilir. Bu durumda $R - S$ 'nin getirdiği kayıtlar;

İsim	Ünvanı	Adres	Cinsiyet	Şehir	Ülke
Ali Can	Sales Manager	Seker Mh. 4883 sk No:9	M	Eskisehir	Turkiye

Yansıtma (π)

Yansıtma işlemi bir tablonun sadece seçilen alanlarından oluşan yeni bir tablo oluşturur. $\pi_{a_1, a_2, \dots, a_n}(R)$ ile gösterilir. İfadeden a_1, a_2, \dots, a_n R tablosundan seçilen alanları gösterir. $\pi_{\text{isim}, \text{Adres}}(R)$ 'nin getirdiği kayıtlar;

İsim	Adres
Ayşe Kara	Kirmizi toprak mh. Gul sk. No:1
Ali Can	Seker Mh. 4883 sk No:9

Seçim (σ)

Seçim işlemi verilen bir tablonun kayıtlarının alt kümesi olarak yeni bir tablo üretir. Getirilen alt küme verilen bir şartla göre seçilir. Seçim şartını gerçekleyen satırlar getirilir. Sonuç başka bir ilişkisel cebir işlemi için girdi olarak kullanılabilir. $\sigma_{\text{Cinsiyet} = F}(R)$ 'nin getirdiği kayıtlar;

İsim	Ünvanı	Adres	Cinsiyet	Şehir	Ülke
Ayşe Kara	Marketing Manager	Kirmizi toprak mh. Gul sk. No:1	F	Eskisehir	Turkiye

Kartezyen Çarpım (x)

İki tablonun alanlarının tüm olası eşleşmelerini getirir. Tablo1×Tablo2 ile gösterilir. Şekil 3.3'de yer alan Tablo T (a) ve Tablo V (b) tablolarının kartezyen çarpımının ($T \times V$) getirdiği kayıtlar Sonuç (c) tablosunda gösterilmektedir. Sonuç tablosunda aynı isimli alanların olması nedeniyle bu alanların karışmaması için önlernerine tablo ismi eklenmiştir. Örneğin T tablosundan gelen B alanı için alan ismi "T.B" olarak gösterilmiştir.

Şekil 3.3

Tablo T			Tablo V			Sonuç TxV					
A	B	C	B	C	D	A	T.B	T.C	V.B	V.C	D
11	2	3	2	3	12	11	2	3	2	3	12
15	4	8	2	3	14	11	2	3	2	3	14
19	4	8	4	8	21	11	2	3	4	8	21

TxV Kartezyen Çarpım İşlemi

Şartlı Bitişme (\bowtie_θ)

Sonuç şeması kartezyen çarpımdaki gibidir. İki tablonun Kartezyen çarpım sonucundan verilen şartta uygun olanlar getirilir. Theta (Theta) bitişmesi diye de adlandırılır. Tablo1 \bowtie_θ Tablo2 olarak gösterilir. Sonuç kümesinde Tablo1 ve Tablo2'nin (Kartezyen çarpımında olduğu gibi) alan adları değiştirilmeden getirilir. Şekil 3.3 T ve V tabloları kullanılarak A < D şartlı bitişim sonucu ($T \bowtie_{A < D} V$) getirdiği kayıtlar Şekil 3.4'te gösterilmiştir. Şekil 3.4'te anlaşılırlığı artırmak için tablolardan gelen benzer alanlar önüne tablo adı yazılmıştır. Örneğin T tablosundan gelen B alanı için "T.B" olarak gösterilmiştir.

Şartlı bitişmede eşitlik operatörü kullanılırsa eşit bitişme (Equijoin, \bowtie_e) olarak adlandırılır. Eşit bitişme sonucu Tablo1'deki alanları (aynı adlarıyla) ve Tablo2'de olup şart içinde yer almayan alanları getirir.

Şekil 3.5'te verilen şartlı bitişmede $T \bowtie_{A < D} V = \sigma_{A < D}(T \times V)$ dir.

Şekil 3.4
T $\bowtie_{A < D}$ V Şartlı Bitişme İşlemi

A	T.B	T.C	V.B	V.C	D
11	2	3	2	3	12
11	2	3	2	3	14
11	2	3	4	8	21
15	4	8	4	8	21
19	4	8	2	3	12
19	4	8	2	3	14
19	4	8	4	8	21

DİKKAT

Doğal Bitişme (\bowtie)

Bitişme işleminin özel bir durumudur. Bu bitişme işleminde sonuç iki tablonun eşit bitişmede tüm ortak alanlarının kullanılmasıyla bulunur. İki tablo arasında en az bir alan ortak ise uygulanabilir. Tablo1 \bowtie Tablo2 olarak gösterilir. Eşit bitişmede olduğu gibi bitişmede kullanılan alanlar Şekil 3.3 verilen T ve V tablolarının doğal bitişim sonucu ($T \bowtie V$) getirdiği kayıtlar Şekil 3.5'te gösterilmektedir.

Şekil 3.5
T $\bowtie V$ Doğal Bitişme İşlemi

A	B	C	D
11	2	3	12
11	2	3	14
15	4	8	21
19	4	8	21

Bölme (\div)

$M(x,y)$ ve $N(y)$ iki tablo olmak üzere $M \div N$ işlemi alan değeri y'ye eşit olan M tablosu içindeki (x) alan değerlerini verir. Aşağıdaki Şekil 3.6'da "tüm ayakkabılardan almış kadınları bulmak" için bölme işlemi (TabloM : TabloN) kullanılmaktadır. Sonuçta sadece "Ayşe" nin tüm ayakkabılardan aldığı bulunur.

Sekil 3.6

Tablo M ÷ Tablo N
Bölme İşlemi

Tablo M		Tablo N		Sonuç	
İsim	Ayakkabı	İsim	Ayakkabı	İsim	Ayakkabı
Ayşe	Kırmızı				
Gül	Mavi				
Ayşe	Mavi				
Dilek	Yeşil				
Hatice	Sarı				
Ayşe	Yeşil				
Ayşe	Sarı				

Yeniden Adlandırma (ρ)

İlişkisel cebir işlemleri uygulanarak elde edilen kayıtların oluşturduğu tabloların alan adları değiştirilebilir. Yeniden adlandırma $\rho_{(A_1, A_2, \dots, A_n)}$ (Tablo1) ile ifade edilir. Tablo1'den gelen alanlar A1, A2, ..., An başlıklarını ile gösterilir. Şekil 3.3 c şıklıkında verilen Kartezyen çarpım işleminde V tablosundan getirilen alanlara yeniden adlandırma uygulandığında $(T \times \rho_{(E, G, D)}(V))$ getirilen kayıtlar Şekil 3.7'de gösterilmektedir.

Sekil 3.7

$T \times \rho_{(E, G, D)}(V)$
Yeniden Adlandırma
İşlemi

A	B	C	E	G	D
11	2	3	2	3	12
11	2	3	2	3	14
11	2	3	4	8	21
15	4	8	2	3	12
15	4	8	2	3	14
15	4	8	4	8	21
19	4	8	2	3	12
19	4	8	2	3	14
19	4	8	4	8	21

İlişkisel cebirde verilen bu basit işlemlerin yanı sıra, istenilen karmaşıklıkta yeni ifadelerin oluşturulmasına izin verir. Böylece karmaşık işlemler ya da karmaşık işlemler sonucunda gelen kümelerinde kullanıldığı yeni tablolar tanımlamak mümkündür. Bu basit işlemlerin yanı sıra ilişkisel cebirde ilave işlemler vardır. Bunlar çiftleri eleme, kümeler üzerinde yapılan işlemler (minimum, maksimum, toplam, vb.), gruplama, genişletilmiş yansıtma işlemleri, sıralama işlemleri ve dış bitişme işlemleridir. Bu ilave işlemler ilişkisel cebir kapsamında anlatılmayacaktır.

İlişkisel cebirde verilen işlemlerin ilişkisel veritabanlarında uygulanması SQL dili ile yapılır. SQL ilişkisel cebirde anlatılan bu işlemleri ve bunların karmaşık yapılarını destekler. Bu üitede anlatılmayan diğer genişletilmiş işlemler ve kısıtlar SQL komutları anlatılırken gözlemlenebilecektir.

SIRA SİZDE



2

Şekil 3.3'te verilen T ve V tablolarına göre $\sigma_{A=1} (T \bowtie_{A=D} V)$ işleminin sonucu nedir?

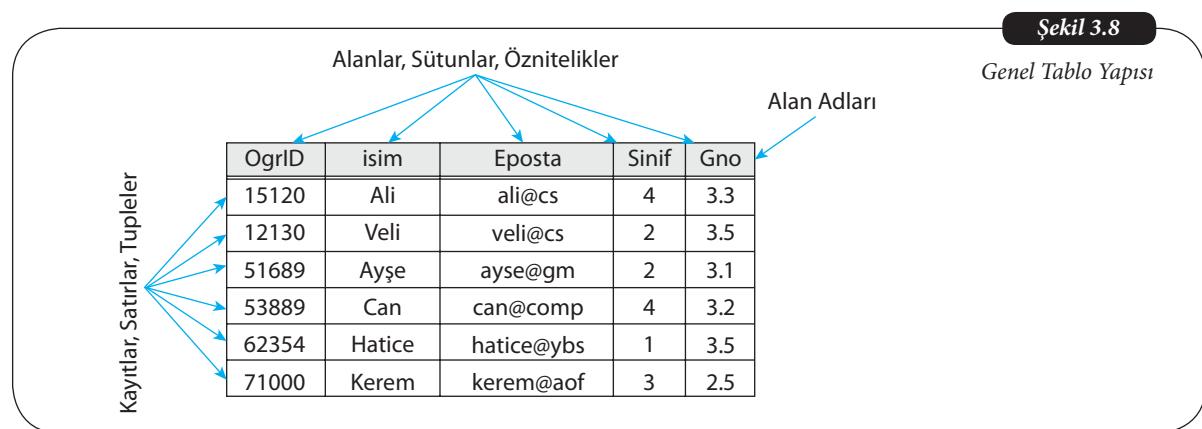
İLİŞKİSEL VERİTABANI NESNELERİ VE KAVRAMLAR

İVTYS ilişkisel veritabanı modelini esas alan yönetim yazılımıdır. İVTYS ilişkisel veritabanının yönetimini, veritabanı ile iletişimini sağlayan arayüzlerden oluşturur. İVTYS'de veri tablo adı verilen veri nesnelerinde tutulur. Bu bölümde temel veritabanı nesneleri olan tablolar (tables), görünümler (views), indeksler (indexes) ve saklı yordamlar (stored procedures) anlatılacaktır. Aynı zamanda tabloları oluşturan kayıtlar, alanlar, alanlarda saklanabilecek değerlerin veri tipleri ve kısıtlardan bahsedilecektir.

Tablolar, İlişkisel Örnek

İlişkisel modele göre veri tabloları içinde tutulur. Her bir tablo kayıtlar ve alanlardan oluşur. Şekil 3.8'de tipik bir tablo yapısı gösterilmiştir. Bu tabloda öğrenci bilgileri saklanmaktadır.

İlişkisel veritabanı yönetim sistemi (İVTYS): Relational Database Management System (RDBMS).



Tablolar veritabanının baş yapılarıdır ve her bir tablo tek ve özel bir temayı temsil eder. Daha önceden de ifade edildiği gibi tablo içindeki kayıtların ve alanların sıralamasının bir önemi yoktur. Böylece veri, ilişkisel veritabanında sunucudaki fiziksel depolama alanından bağımsız olarak tutulmuş olur. Her tabloda tablo içindeki satırları benzersiz olarak ifade etmeye çalışan en az bir alan bulunur.

Verilen tablo içinde tutulan tema bir nesne veya hareket olabilir. Tablo içindeki tema bir nesne ise, bir kişi, bir yer gibi somut bir şey temsil ettiği anlamına gelir. Bu tablolar birden fazla kullanılan değerleri tutmak için kullanılır ve doğrulama tabloları (validation, lookup) olarak adlandırılır. Örneğin özneler, aktörler, şehir adları, yetenek kategorileri, ürün kodları ve proje belirleme kodları gibi veriler doğrulama tablolarında tutulur. Bu tablodaki veriler neredeyse statik olup veri ekleme, düzenleme ve çıkartma işlemleri nadiren yapılır. Bu tablolarda yer alan birincil anahtar bağlılığı olduğu hareket tablosunda yabancı anahtar olarak kullanılır. Birincil ve yabancı anahtar tablolar arasında bağlantı kurmak için kullanılır ve ilerleyen kesimlerde detaylı olarak açıklanacaktır. Örnek veritabanındaki [Müşteriler], [Sağlayıcılar], [Taşıyıcılar] doğrulama tablolarına örnek olarak verilebilir.

Hareket (transaction) tablolarında ise dinamik veri tutulur. Tablo hareketin zaman içinde oluşunu gösteren karakteristik özelliklerden oluşur. Hareket tablolarında yapılan işlem bilgileri ve nesne tablolarına ilişki kuran yabancı anahtarlar bulur. Örnek veritabanındaki [Siparişler], [Sipariş Ayırtıları] tabloları hareket tablolarına örnek olarak verilebilir. Örneğin [Siparişler] tablosundaki her bir kayıt bir satış işlemi hareketini ifade eder.

Büyük miktarda verinin tutulduğu Veri ambarı sistemleri boyutsal veri modeli kullanırlar. Bu modelde de veriler tablolar ile tutulur. Hareket verileri gerçek tablolarda (fact table), hareket ile ilişkili olan nesneler ise boyut tablolarında (dimension table, lookup table) tutulur.



DİKKAT

Kayıtlar, Satırlar ve Tuplelar

Kayıt, satır ve tuple hep aynı anlamda kullanılmaktadır. Şekil 3.8'de bir tablodaki kayıtlar görülmektedir. Kayıt bir tabloda verilen temanın benzersiz bir örneğini temsil eder. Bir kayıt tablodaki (dolu ya da boş olup olmadığına bakılmaksızın) tüm alanlardan oluşur.

Şekil 3.8'de verilen tabloda her bir kişi bir [OgrID] ile veritabanında benzersiz olarak belirlenir. Aynı zamanda tablodaki her bir kayıt bir kişinin farklı özelliklerini gösterir. Örneğin Kerem'i ele alırsak; verilen tablodaki Kerem'e ait kayıt onunla ilişkili özellikleri içermektedir.

Alanlar, Sütunlar, Öznitelikler

Alan, sütun ve öznitelik ifadelerinin hepsi veritabanı tablosundaki alanları belirmek için kullanılan terimlerdir. Alanlar her bir tekrarlanan kayıt içindeki veri yoğun yapısını ve tanımını gösterir. Bu nedenle tekrar eden kayıtlardaki her bir alandaki veri farklı olabilir. Doğru tasarılmış bir veritabanında her bir alanda sadece bir değer tutulur. Alan içinde tuttuğu veriye göre isimlendirilir. Böylece bir alana veri girişi yapılrken sezgisel olarak içerik anlaşılmış olur. Örneğin alan adları *Ad*, *Soyad*, *Adres* ve *Sehir* verilirse her bir alana girilecek değer hakkında kullanıcı kesin bilgi sahibi olur. Mehmet isimli bir kullanıcıyı aramak için ya da şehir bazında sıralama yapmak için kullanılacak alanların *Ad* ve *Sehir* olduğu kolayca anlaşılmış olur.

İyi tasarlanmış bir veritabanı için alanlar tanımlanırken aşağıda verilen üç durumdan kaçınmak gereklidir. Bu üç durum aşağıda açıklanmış ve şekil 3.9'da örnekleri gösterilmiştir.

- *Çok parçalı alan*: Aynı alanda iki veya daha fazla farklı bilgiyi tutar. Örneğin Şekil 3.9'da gösterildiği gibi *postakodu* ve *sehir* bilgilerini saklamak için [*PostakoduSehir*] adında tek bir alan tanımlamak yanlış olacaktır. Bu durumda tablo içinden *posta-kodu* ve *sehir* adına göre arama veya sıralama yapmak güçleşecektir.
- *Çok değerli alan*: Aynı tip verinin farklı örneklerini aynı alanda tutar. Yine alandaki farklı değerler bazında işlem yapmak güçleşmiş olacaktır. Bu tip alanlar bir satırda bir değer olacak şekilde yani aynı kişinin farklı kayıtları olarak düzenlenir.
- *Hesaplanan alan*: Birleştirilmiş metin ya da ayrıştırmak için matematiksel hesap gereken değerleri tek bir alanda tutar. Bu durumda da örnekte verilen alanın iki ayrı alan olarak tanımlanması sorunu giderecektir.

Şekil 3.9

Çok Parçalı, Çok Değerli, Hesaplanan Alanları Gösteren Bir Tablo

Hesaplanan Alan		Çok Parçalı Alan		Çok Değerli Alan	
id	MüşteriAdSoyad	Sınıf	Gno	PostakoduSehir	Dersler
15120	Ali Durmaz	4	3.3	26480, Eskisehir	Mat, Fizik
15130	Veli Demir	2	3.5	06380, Ankara	Türkçe
51689	Ayşe Kara	2	3.1	34120, İstanbul	İşletme, Maliye
53889	Can Kale	4	3.2	35222, İzmir	Mat, Fizik
62354	Hatice Kosan	1	3.5	11130, Balikesir	Girisimcilik
71000	Kerem Kozan	3	2.5	42341, Konya	Hukuk, İşletme

SIRA SİZDE



3

Şekil 3.9'da verilen tablodaki çok parçalı alan [*PostakoduSehir*] nasıl değiştirilirse “*Sehir* adına göre aramak daha kolay olacaktır?

Veri Tipleri

Bu eleman tablonun alanlarında depolanan verinin yapısını gösterir. Bu kesimde veri tipleri Basit, Karmaşık, Özelleştirilmiş olarak gruplanarak anlatılacaktır.

Basit Veri Tipleri

Bu veri tiplerinde tek bir değer üzerinde bir örtüsü veya değer kısıtlaması yapılır. Sayılar buna örnek olarak verilebilir.

Karakter: Bu veri tipi sabit veya değişken boydaki karakter dizilerinde (bir veya daha fazla yazılabilir) karakterleri depolamakta kullanılır. Sabit boyutlu karakter veri tipi **CHARACTER** ya da **CHAR** olarak bilinir. Değişken boydaki karakter dizisi veri tipi ise **CHARACTER**, **CAHR VARYING** veya **VARCHAR** olarak bilinir.

Sabit boydaki karakter dizisinde yabancı dildeki karakter kümelerinden karakterleri depolamak için **NATIONAL CHARACTER**, **NATIONAL CHAR** ve **NCHAR** veri tipleri kullanılır. Değişken boydaki yabancı dildeki karakter dizilerini depolamak için ise **NATIONAL CHARACTER VARYING**, **NATIONAL CHAR VARYING** ve **NCHAR VARYING** kullanılır.

Bit: Bu veri tipi ikili sayı sistemindeki sayılarından (binary number) oluşan dizileri depolamakta kullanılır. İkili sayı sistemindeki verilere örnek olarak dijital görüntü ve ses verisi verilebilir. Bu veri tipi **BIT**, **BIT VARYING**, **BINARY** veya **VARBINARY** olarak kullanılır.

Tam Sayısal: Bu veri tipi tüm sayıları ve ondalık ayırcılı sayıları depolar. İVTYS yazılımlarının büyük çoğunluğu tam sayısal veri tipi olarak **NUMERIC**, **DECIMAL(DEC)**, **INTEGER (INT)** ve **SMALLINT** kullanır.

Yaklaşık Sayısal: Bu veri tipinde ondalık ayırcılı sayılar ve üslü sayılar depolanır. İVTYS yazılımlarında yaklaşık sayısal veri tipi için **FLOAT**, **REAL** ve **DOUBLE PRECISION** kullanılır.

Tarih ve Zaman: Bu veri tipi genelde **TIMESTAMP** olarak bilinir ve tarihleri, saatı veya her ikisini birden depolamakta kullanılır. Bu veri tipi farklı İVTYS yazılımlarında çok farklı uygulanabilir. Bu nedenle bu veri tipini kullanmadan önce İVTYS yazılımının tarih ve saatleri nasıl kullandığı öğrenilmelidir.

Karmaşık Veri Tipleri

Karmaşık veri tipleri nesne veri tiplerini kapsar. Bu veri tipleri nesnelerin kullanımı ve ilişkisel veritabanları arasında bir köprü görevi görür. İkili sayı ile temsil edilen nesneler (binary objects), veri grup dizileri (collection arrays) bu veri tipindendir. Örneğin bir resim ikili sayı sisteminde bir alanda tutulabilir. Bu veri tipi genelde nesne tabanlı veritabanı sistemlerinde yaygın olarak kullanılır.

Farklı İVTYS yazılımlarında nesne veri tipleri değişir. Bazı ilişkisel veritabanları diğerlerinden daha fazla nesne-ilişkisel veri tipleri sunar. Bazı karmaşık veri tipleri aşağıda verilmiştir:

İkili (binary) nesneler: İkili nesneler olağan ilişkisel veritabanı kayıt yapılarından ikili veriyi ayırmak için oluşturulmuştur. Resim gibi büyük nesneleri içeren tablolar, karakterler ve sayılar içeren ortalama bir tablo kaydından çok daha büyütür. Bu nedenle depolama problemi ortaya çıkabilir. Veritabanı tablo kayıtlarını fiziksel olarak 2 ila 8 kilobayt büyülüklükte bloklarda tutar. Bu alana karakter ve sayısal alanlardan oluşan tablo kayıtları kolayca sıyrılmak, resim, ses benzeri ikili veriler (çok büyük olabileceğiinden) sıyrılmaz ve veritabanının fiziksel veri depolama özelliğini zorlayarak verimsiz kullanımına sebep olacaktır. Bu nedenle ikili nesneler genelikle tablo kayıt değerlerinden farklı olarak oluşturulmuştur. Büyük karakter dizileri, video, XML veri, ses vb. ikili nesneler olarak tutulur.

Referans işaretçileri (pointers): Bazı ilişkisel veritabanları veritabanı dışına depolamış nesne veya dosyaları işaretlemek (point) için bu veri tipini kullanır. Veritabanı dışındaki nesne veya dosyaları gösteren bilgi tablonun kayıtlarındaki bir alanda tutulur. Tablonun ilgili alanında tutulan bilgi sadece adresi içerir. Böylece büyük nesneler veritabanı dışına depolandığı için veritabanının fiziksel depolama alanını yönetmesindeki verimlilik bozulmamış olur.

Koleksiyon diziler: Bu veri tipinde tablo alanında dizi tutulur. Koleksiyon diziler sabit veya dinamik uzunlukta olabilir.

Kullanıcı tanımlı: Bazı İVTYS yazılımları kullanıcının kendi tanımladığı veri tiplerini kullanmasına izin verir. Kullanıcı tanımlı veri tipleri yazılım ile oluşturulabilir ve tablolara alanlarının veri tipi olarak atanabilir.

Özelleştirilmiş Veri Tipleri

Bu veri tipleri daha gelişmiş ilişkili veritabanı sistemlerinde görünür. Özelleştirilmiş veri tipleri verinin yapısına uygun depolama sağlamak için oluşturulur. Örneğin MS SQL Server İVTYS yazılımındaki bazı özelleştirilmiş veri tipleri;

XML: Hiyerarşik düzende verilerin saklanması sağlayan bir veri saklama sistemi-ğidir. XML yapısı içerisinde saklanan verilere göre farklılaşabilir.

Geography: Coğrafi verilerin saklanması için özelleştirilmiş veri türleridir. Enlem, meridyen, yükseklik gibi konumsal verilerin saklanması için oluşturulmuştur.

Geometry: Geometrik şekillerin standart bir tanımı yapılarak yeniden çizilmesine yönelik verilerin saklanması sağlar. Şekillerin en, boy, yükseklik ve koordinat gibi verilerini barındırır.

Hierarchyid: Birbirleri ile ilişkili nesneleri hiperarşik bir ağaç yapısında saklayan veri türüdür. Bu yapı sadece veri değil verilerin birbiri olan ilişkilerini saklayan bir veri türüdür.

Null Değerinin Kullanımı

NULL bilinmeyen veya olmayan değeri gösterir. NULL sayısal değerlerdeki sıfır ya da karakter dizilerindeki bir veya daha fazla boş karaktere karşılık gelmez. Şekil 3.10'da verilen [Ürünler] tablosunda [No] alanındaki değeri 5 ve 6 olan kayıtlarda herhangi bir nedenle [Standart Maliyet] değeri boş (NULL) bırakılmıştır. Bilinmeyen değerler tablolarda çeşitli nedenlerle yer alabilir. İlgili değerin henüz belirlenmemiş olması, alanın tabloya yeni eklenmesi ve eski kayıtlar için henüz tanımlanmamış olması ya da bazı durumlarda varlığın yapısı gereği herhangi bir değer alamayabilmektedir.

NULL değerin dezavantajı matematiksel operasyonlarda ya da fonksiyonlarda hataya sebep olabilmesidir. NULL değeri içeren bir işlemin sonucu yine NULL olur. Örneğin;

$$\begin{aligned} (25 \times 3) + 4 &= 79, \\ (\text{Null} \times 3) + 4 &= \text{Null}, \\ (25 \times \text{Null}) + 4 &= \text{Null}, \\ (25 \times 3) + \text{Null} &= \text{Null} \text{ değer alır.} \end{aligned}$$

Benzer şekilde Şekil 3.10'da yer alan [Standart Maliyet] alanı üzerinde gerçekleştirilecek aritmetik bir işlem NULL satırlar için sayısal bir değer değil NULL değeri ile sonuçlanacaktır. Ancak SQL komutları ile ilgili alanın toplamı, ya da ortalaması gibi bir birleştirme işlemi yapılrsa yazılım NULL değeri olan satırları hesaba katmayacaktır.

Sekil 3.10

Null Değer İçeren Bir Veri Tablosu

No	Ürün Kodu	Ürün Adı	Standart Maliyet	Liste Fiyatu
1	NWTB-1	Hint Çayı	13.5	18
3	NWTCO-30	Traders Şurup	7.5	10
4	NWTCO-4	Cajun Çeşnilik	16.5	NULL
5	NWTO-5	Zeytin Yağı	NULL	21.35
6	NWTJP-6	Böğürtlen Reçeli	NULL	25
7	NWTDFN-7	Kurutulmuş Armut	22.5	30
8	NWTS-8	Köri Sosu	30	40
14	NWTDFN-14	Ceviz	17.4375	23.25
17	NWTCFN-17	Meyve Kokteyli	29.25	39
19	NWTBGM-19	Çikolatalı Bisküvi Karışımı	6.9	NULL
20	NWTJP-6	Marmelat	60.75	81
21	NWTBGM-21	Kurabiye	7.5	10

Kısıtlar (Constraint)

Kısıtlar tablodaki alanlar üzerine kuralların uygulanmasını mümkün kılar. Tablo içine giren veri türünün kısıtlanması veritabanı içindeki verinin doğruluk ve güvenirliliğini sağlar. Kısıtlar tablo veya alan seviyesinde olabilir. Alanlar üzerindeki kısıtlayıcılar sadece ilgili alana uygulanırken, tablo seviyesi kısıtlar tüm tabloya uygulanır. Tablo kısıtı alan tanımlarından ayrı olarak yapılır ve tablonun birden fazla alanına uygulanabilir. Birden fazla alan üzerine kısıt konulmak istenirse tablo kısıtı kullanılmalıdır. Örneğin bir tabloda iki alanı içeren birincil anahtar tanımlanmak istenirse, birincil anahtar kısıtına iki alanın adı da yazılır. İzleyen kesimde çok kullanılan kısıtlar açıklanacaktır.

NOT NULL Kısıtı

Veritabanı tablosunun alanlarının mutlaka dolu olması gerekiğinde boş bırakılmaması için uygulanan bir zorlamadır. Eğer bir alanın özelliği NOT NULL olarak belirlenmiş ise tabloya yeni bir kayıt eklenirken bu alana ait değerin boş bırakılmasına izin verilmez. Kullanıcı bir hata mesajı ile bilgilendirilir ve kaydın tamamlanmasına izin verilmez.

DEFAULT Kısıtı

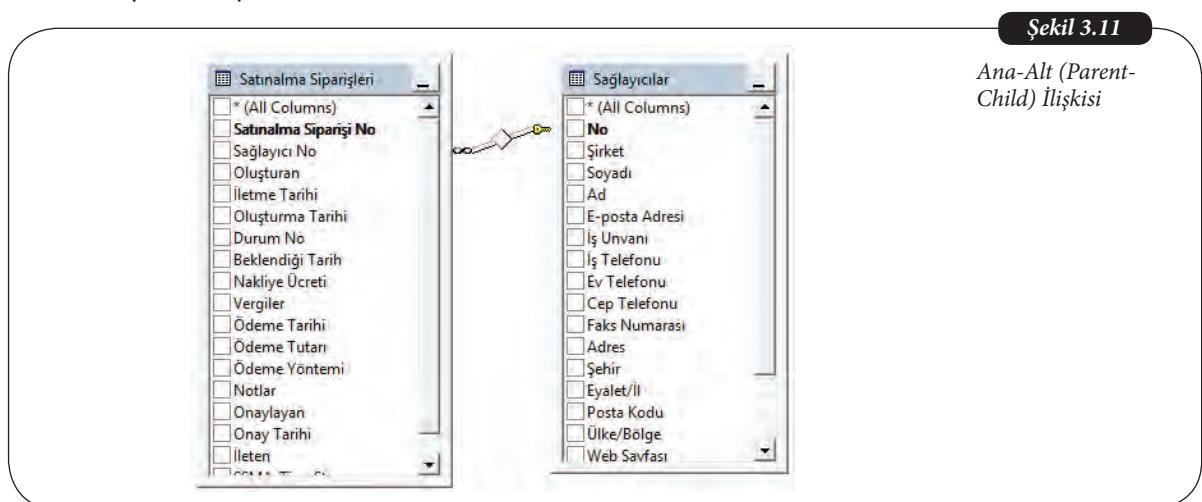
Tabloya bir kayıt eklendiği veya değiştirildiği zaman **DEFAULT** kısıtlı alana değer belirtmemiş ise ilgili alana varsayılan bir değer atanmasını sağlar. Özellikle bir alanda **NULL** değerlere izin verilmiyor ve **DEFAULT** tanımı yapılmamış ise ilgili tabloya kayıt eklenirken ilgili alan dolu olmalıdır aksi hâlde veritabanı hata verecektir. **DEFAULT** kısıtı tablo oluşturulurken varsayılan değerin atanacağı alan tanımına eklenir.

Anahtar Kısıtı

Anahtar kısıtları birincil anahtar (primary key), benzersiz anahtar (unique key) ve yabancı anahtardan (foreign key) oluşur. Anahtar kısıtları farklı tablolardaki alanlar arasındaki değerlerin kontrol ve doğrulanmasına izin verir. Birincil ve yabancı anahtarlar ana-alt (parent-child) tablolar arasında ilişki kurmak için kullanılırlar. Şekil 3.11'de [*Satınalma Şiparişleri*] ana tablosunda satınalma ile ilgili bilgiler saklanırken onun alt tablosu olan [*Sağlayıcılar*] tablosu içinde satınalma işlemi ile ilgi sağlayan bilgisi saklanmaktadır. İki tablo arasındaki ilişki tablolar arasındaki bağlantı çizgisi ile gösterilmiştir. [*Sağlayıcılar*] tablosundaki [*No*] "Birincil Anahtar", [*Satınalma Şiparişleri*] tablosundaki [*Sağlayıcı No*] "Yabancı Anahtar"dır. Birincil Anahtar, Benzersiz Anahtar ve Yabancı Anahtar izleyen kesimde açıklanacaktır. Ancak MS SQL Server'da oluşturulması ve kullanımı ilerleyen ünitelerde detaylı olarak yer alacaktır.

Sekil 3.11

Ana-Alt (Parent-Child) İlişkisi



Birincil Anahtar (Primary Key)

Bu anahtar tablo içindeki bir kaydı benzersiz olarak belirlemek için kullanılır. Örneğin Şekil 3.11'deki [*Satınalma Şiparişleri*] tablosunda [*Satınalma Şiparişi No*], [*Sağlayıcılar*] tablosunda [*No*] birincil anahtarlardır. Bu alanda tutulan değerler benzersiz olup, [*Satınalma Şiparişleri*] tablosunda [*Satınalma Şiparişi No*] biliniyorsa ürünün tüm bilgilerine kesin olarak erişilebilir. Diğer alanlardaki veriler ilgili ürüne erişmek için yeterli olmamıştır. Örneğin [*Oluşturan*] alanındaki değer ilgili kayına erişmek için yeterli değildir. Aynı bireyin oluşturduğu birçok satınalma siparişi olabilir. Birincil anahtar tek bir alan ile tanımlanabilecegi gibi birden fazla alan ile de oluşturulabilir. Birden fazla alan ile oluşturulan birincil anahtara *kompozit birincil anahtar* denir. Bir kayttaki birincil anahtarın değeri kaydın tüm veritabanında belirlenmesini sağlar. Birincil anahtarın alanı sayesinde de tanımlandığı tablo tüm veritabanı tarafından belirlenmiş olur. Birincil anahtar tablo seviyesi tutarlığı sağlar ve veritabanındaki diğer tablolar ile ilişkilendirilmesine yardımcı olur. Birincil anahtarın özellikleri;

- Tabloda bir kaydı belirlemek için kullanılır,
- Tabloda sadece bir Birincil anahtar tanımlanabilir,
- Birincil anahtar içindeki değerler benzersizdir,
- **NULL** değeri alamaz,
- Birincil anahtar kümelenmiş (clustered) indeks kullanır (ünite 5'te açıklanacaktır).

Benzersiz Anahtar (Unique Key)

Bu anahtar bir alanın değerlerinin benzersizliğini belirler. Bu sayede tablodaki tüm kayıtlar içinde benzersiz tanımlı alanda birden fazla aynı değer bulunamaz. Benzersiz anahtarın özellikleri;

- Bir tabloda birden fazla benzersiz anahtar olabilir,
- Benzersiz anahtara **NULL** değeri atanabilir,
- Benzersiz anahtar kümelenmemiş (Non-clustered) indeks kullanır.

Yabancı Anahtar (Foreign Key)

Yabancı anahtarlar ana tablodaki birincil anahtarın alt tablodaki karşılık gelen kopyasıdır. Yani tablolar arası ilişkilerde (ana-alt tablolarında) bağlantının alt tarafını belirtir. Yabancı anahtar adı; ana tablo zaten kendi birincil anahtarına sahip olduğu için, alt tablonun birincil anahtarının ana tabloda "yabancı" olmasından gelir. Yabancı anahtar değeri bağlı olduğu tablodaki birincil anahtar değerlerine uygun olmak zorundadır. Örneğin Şekil 3.11'de [*Sağlayıcılar*] tablosunda [*No*] alanı birincil anahtar iken bağlantının karşı tarafındaki [*Satınalma Şiparişleri*] tablosundaki [*Sağlayıcı No*] alanı yabancı anahtar olarak tanımlanır.

Yabancı anahtar ile tablolar arasında kurulan ilişki sayesinde veritabanı seviyesinde ilişki seviyesi tutarlılık sağlanmış olur. Yabancı anahtar her iki tablonun uygun bir şekilde ilişkilendirilmesini ve veri tutarlığının korunmasına yardımcı olurlar. Örneğin [*Sağlayıcılar*] tablosunda yer alan bir sağlayıcının silinmesi, hâlen ilgili sağlayıcı numarasının yer aldığı [*Satınalma Şiparişleri*] tablosunda belirsiz bir kayıt oluşturabilecektir. Ya da [*Sağlayıcılar*] tablosunda yer almayan bir sağlayıcı numarasının [*Satınalma Şiparişleri*] tablona eklenmesi tanımsız sağlayıcı oluşturulmasına neden olabilir. Bu nedenle sistemde oluşturulacak yabancı anahtar ilişkileri bu hataları engelleyecek şekilde düzenlenebilir. Bu sayede tutarsız veri oluşumuna engel olunabilir. Yabancı anahtarın bağlı olduğu ana tabloda eklenilen/güncellenen değer var ise ya da alt tabloya eklenilen yabancı anahtar değeri **NULL** ise kayıt eklenebilir. Yabancı anahtarın özellikleri;

- Tablolar arası geçiş ve sorgulama işlerini kolaylaştırırlar,
- Benzersiz olmak zorunda değildir,
- Bir tabloda birden fazla tanımlanabilir.

CHECK Kısıtı

Bu kısıt ile bir veya daha fazla alanına girilebilecek değerlerin sınırlanması sağlanır. Örneğin [*Ürünler*] tablosunda [*Hedef Düzey*] alanının alabileceği değerler 0 ile 100 arasında sınırlanır. Bu **CHECK** kısıtının tanımdan sonra bir ürün eklenirken [*Hedef Düzey*] alanına sadece 0-100 aralığındaki değerler girilebilir. Bir alan üzerine birden fazla **CHECK** kısıtı tanımlanabilir. Aynı şekilde tablo seviyesinde tanımlanması durumunda birden fazla alan üzerine tek bir **CHECK** kısıtı tanımlanabilir. Örneğin [*Sağlayıcılar*] tablosunda [*Ülke/ Bölge*] ve [*Posta Kodu*] alanlarına CHECK kısıtı tanımlanarak ülke değeri olarak Türkiye girildiğinde [*Posta Kodu*] alanına da sayısal değer girilmesi zorlanabilir.

CHECK ve yabancı anahtar kısıtları tanımlanan alanaya girilecek değerin kontrol edilmesi açısından benzerdir. Farkları ise, yabancı anahtar girilecek değerin bağlı olduğu birincil anahtar alanında olup olmadığını kontrol ederken, **CHECK** kısıtı girilen değerin tanımlanan mantıksal ifadeye uygunluğu kontrol eder.

Şekil 3.16'da verilmiş olan örnek veritabanı şemasında [*Sağlayıcılar*], [*Satınalma Siparişleri*] ve [*Çalışanlar*] tablolarında birincil ve yabancı anahtarlar hangileri olabilir?



Görünümler (Views)

Görünüm, veritabanından bir veya daha fazla tablonun alanlarından oluşturulan sanal tablodur. Görünümler birden fazla tablodan verilerin aynı anda kullanılmasını sağlar. Bu şekilde görünüm oluşturabilmek için tablolar arası bağlantılar anahtarlar ile oluşturulmuş olmalıdır. Görünüm içinden bir kayıt istendiğinde veri görünümünün oluşturulduğu tabloların getirilir, görünüm içindeki veriler ayrı bir alanda depolanmaz. Bu nedenle görünümler sanal tablolar olarak adlandırılır. Görünüm için veritabanında tutulan bilgi görünümün yapısıdır. Görünümler veritabanındaki bilgileri farklı yönlerden görenmenizi sağlar.

Aşağıdaki örnekte hangi müşterinin hangi üründen ne kadar satın aldığıını bulmak için dört tablo kullanılarak bir görünüm yer almaktadır. Veritabanı yönetim sistemleri ilgili görünüm için bu görünümü elde edecek sorgu sizsini saklarlar. Ve ilgili görünüm kullanıcı ya da sistem tarafından talep edildiğinde mevcut tablolar ilişkilendirilerek görünüm türetilir.

Şekil 3.12

Görünüm Oluşturma

The screenshot shows the Object Explorer on the left with the view 'dbo.Sekil3_12_view' selected. The SQL Query window in the center contains the following code:

```

SELECT TOP (100) PERCENT dbo.Müşteriler.Sirket, dbo.Ürünler.[Ürün Adı], SUM(dbo.[Sipariş Ayırtıları].Miktar) AS 'ToplamMiktar'
FROM dbo.Siparişler INNER JOIN
dbo.[Sipariş Ayırtıları] ON dbo.Siparişler.Siparis No = dbo.[Sipariş Ayırtıları].[Siparis No] INNER JOIN
dbo.Müşteriler ON dbo.Müşteriler.Sirket = dbo.Siparişler.Sirket
GROUP BY dbo.Müşteriler.Sirket, dbo.Ürünler.[Ürün Adı]

```

The results grid shows the following data:

Sirket	Ürün Adı	ToplamMiktar
Sirket J	Badem	20
Sirket J	Böğürtlen Reçeli	10
Sirket Z	Böğürtlen Reçeli	90
Sirket J	Cajun Çeşnilik	10
Sirket Y	Cajun Çeşnilik	30
Sirket CC	Çikolata	10

Görünümlerinin kullanılmasının avantajları şunlardır:

- Çok uzun ve karmaşık sorgulardan bir görünüm oluşturarak karmaşık sorguyu çalıştırın yerine görünümdeki basit bir seçim işlemi ile sorguyu tamamlayabiliriz.
- Kullanıcıların tablo veya tabloların yapısını görmesini engeller. Böylece güvenlik anlamında, tabloların yapısını bilmeyen kullanıcıların tablolara müdahale etmesi engellenmiş olur. İlave olarak uygulama geliştirici veya kullanıcılarla sadece görenümlere erişim hakkı verip tablolara erişim hakkı vermeyerek ekstra güvenlik sağlanabilir.
- Birden fazla fiziksel veya mantıksal veritabanına erişmek yerine, tek bir veritabanından veri alınıyormuş gibi veri getirilebilir.
- Gelen veriler üzerinde matematiksel işlemler veya biçimlendirme işlemleri yapılarak alan değeri gibi gösterilebilir.
- Veritabanını mantıksal gruptara ayırıp (yıllar, modeller vb.) daha kolay raporlama ve daha yüksek performans alınabilir.

İndeksler (Indexes)

İndeks çoğunlukla bir tablonun küçük bir kısmının (tablonun bir alanı gibi) kopyasıdır. İndeks oluşturulurken tablonun depolandığı alan dışında farklı bir yere indekste kullanılan alan (veya alanların) kopyası taşınır. Bazı veritabanlarında indeksler tamamen farklı veri dosyalarında saklanır.

İndeksler bir kitabın başındaki içindekiler veya sonundaki indeks bölümü gibi davranışır. Özel bir konu aranırken kitabın indeks bölümünden nasıl doğrudan ilgili sayfa no bulunup konuya erişilirse, aynı şekilde tablodaki indeks ile de istenilen değer indeksten bulunur ve tablodaki kayıta hızlı olarak erişilir. İndeks tanımlı olmayan tabloda bir değer aranırken tüm tablodaki kayıtlar incelenmek zorundadır. İlişkisel veritabanları birden fazla indeksleme teknüğünü destekler. Böylece erişim örüntüsüne, ilişki boyutuna ve verinin dağılımına uygun en iyi arama desteklenmiş olur. İndeksler genellikle B-trees, R-trees ve bitmap olarak uygulanırlar. B-tree ile indekslenmiş tabloda n kayıt var ise indekslenmiş alanda bir değeri aramak için kontrol edilecek kayıt sayısı $\log(n)$ gibi küçük bir değer olacaktır. İndeksli olmayan bir alanda arama yapmak için ise n kayıt kontrol edilmek zorundadır.



DİKKAT

Birincil anahtar ve yabancı anahtarlar üzerine tanımlanan indeksler performansı büyük ölçüde arttırmır.

İVTYS yazılan SQL komutunu çalıştırırken bir eniyileyici yazılım (optimizer) kullanır. Bu optimizer yazılımı SQL komutuna göre tablolar arasında bağ kurarken mümkün olduğunda indeksli alanları kullanır. Bu nedenle indeksler uygun şekilde tanımlanmaz ise SQL komutların yavaş çalışmasına sebep olacaktır. Aşağıdaki durumlarda indeksleme tercih edilmemelidir.

Cok fazla indeks oluşturmayın. Verilerin eklenmesi/değiştirilmesi/silinmesi sırasında indekslerin tekrar düzenlenmesi gerekebileceğinden çok fazla indeks kullanımı veritabanının yavaşlamasına sebep olacaktır.

Bir indekste çok fazla alan kullanmayın. Çok fazla alan bir indekste kullanıldığından indeks içindeki tüm alanlar sorgu içinde kullanılmak zorunda olduğundan, sorgu yapma komutları daha karmaşık hâle gelecektir. İndeksler normalde tablonun kapladığı depolama alanından daha az yer kaplamalıdır. Birden fazla alan ile yapılan indeksler çok fazla alan kaplarlar.

Farklı değeri az olan alanları indekslemeyiniz. İndeks yapılan alan içindeki değerler birbirlerinden çok farklı değil ise indeksleme yapmayınız. İndeksin yararlı olabilmesi için indeksleme yapılacak alan içindeki değerler ayırıcı olmalıdır. Örneğin bir alanda sadece cinsiyet bilgisi tutuluyor ise (erkek, kadın gibi) bu alanda indeksleme yapılmaz. Çünkü böyle bir indekse göre arama yapıldığında, indeks arama uzayını sadece iki parçaya bölecek ve ilgili kayda erişmek için neredeyse kayıtların yarısında tüm değerlere tek tek bakmak gerekecektir.

Saklı Yordamlar (Stored Procedures)

İVTYS içinde yapılan programlamanın büyük kısmı saklı yordamlar ile yapılır. Genellikle saklı yordamlar veri üzerinde ön işlem yaparak veritabanı dışına gönderilecek verinin miktarını düşürmeye kullanılır. Ayrıca güvenliği artırmak için sistemler tasarılanırken kullanıcıların sadece saklı yordamlara erişimi açılır ve tablolara erişim izni verilmez. Bu durumda kullanıcılar veriye erişmek istediği sakin yordama başvurur, sakin yordamda kullanıcının istediği veriler tablolardan derlenir/işlenir ve kullanıcıya dönülür.

Saklı yordamlar veritabanında saklanan ve verileri işleyen çalıştırılabilir program kodudur. Saklı yordamlar ilişkisel veri tabanlarının bir parçası olmadığı hâlde neredeyse tüm modern İVTYS'ler tarafından desteklenir.

Şekil 3.16'da verilmiş olan [Ürünler] tablosuna kayıt eklerken [Ürün Adı] alanına boş değer girilememesi ve [Birimdeki Miktar] alanının eksi değer alamaması için hangi kısıtlar kullanılmalıdır?



SIRA SİZDE

5

İLİŞKİSEL VERİTABANI ÖRNEĞİ

İVTYS ilişkisel veritabanı modelini esas alan yönetim yazılımıdır. İVTYS ilişkisel veritabanının yönetimini ve veritabanı ile iletişimini sağlayan arayüzlerden oluşur ve tüm modern veritabanı sistemleri (MS SQL Server, IBM DB2, Oracle, Sysbase, MySQL, ve Microsoft Access) tarafından desteklenir. İzleyen ünite MS Access yazılımı konusunda bilgi aktarımaktadır. Daha sonraki üniteler ise MS SQL Server kullanılarak ilişkisel veritabanı sistemlerinin yönetimi, oluşturulması, düzenlenmesi, bakımı için kullanılan SQL komutları örnekler ile anlatılacaktır.

Bu kitapta SQL komutları anlatılırken Microsoft SQL veritabanları için üretilmiş olan örnek veritabanı olan Northwind kullanılacaktır. Microsoft tarafından dağıtılan bu veritabanının içerisinde oluşturulmuş birçok içi dolu tablo bulunmaktadır. Bu tablolar olabildiğince detaylidir ve neredeyse her tür değişken tipine sahiptir. Örneklerin anlaşılır olmasına yönelik olarak Northwind veritabanının Türkçe sürümüne yer verilmiştir.

Northwind Veritabanı

Bu kitaptaki pek çok örnek doğrudan bu örnek veritabanında uygulanabilir. Bu veritabanı uluslararası satış yapan bir firmanın personel bilgileri, satışları, müşteri bilgileri, satılan ürünleri ve tedarikçilerinin bilgilerini tutulacak şekilde tasarlanmıştır.

Bu üitede kullanılacak MS Northwind veritabanı dosyalarına ulaşmak için <https://bit.ly/30UFiLp> adresinde yer alan Northwind.rar dosyasına ulaşabilirsiniz.



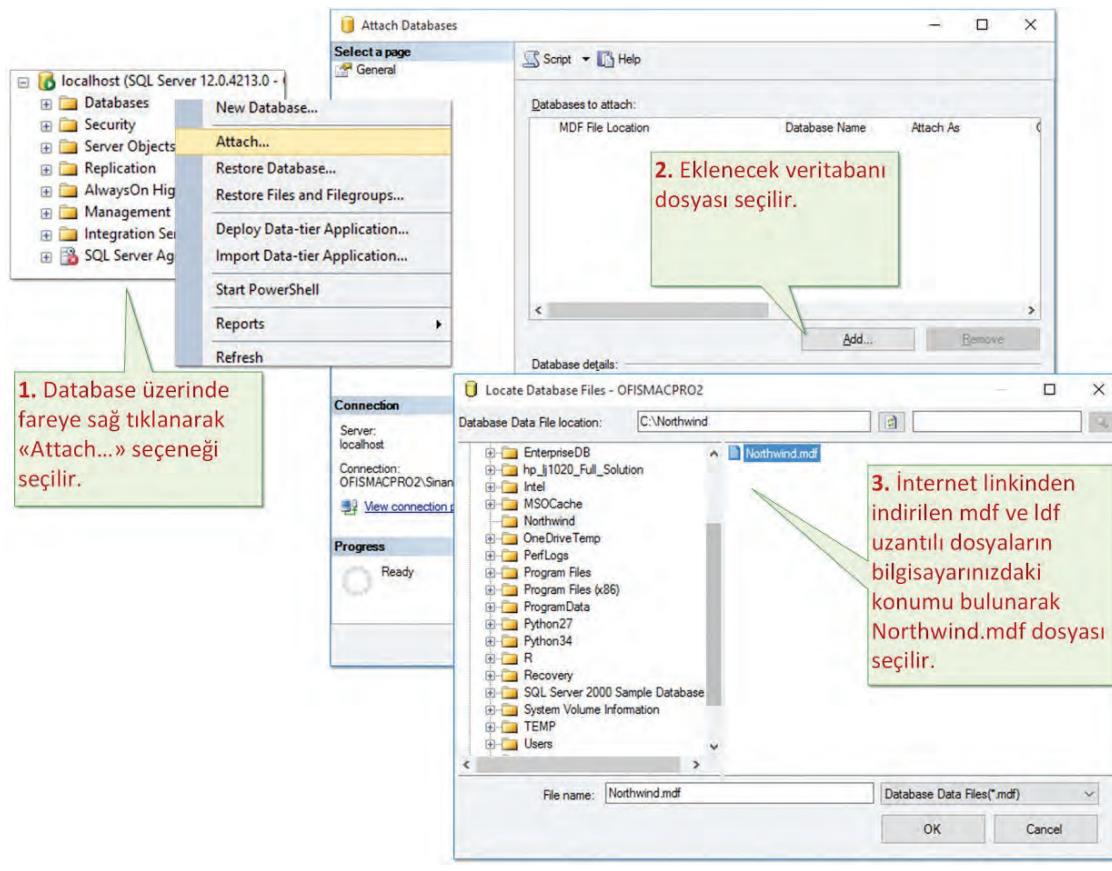
INTERNET

Kaynakta verilen bağlantından indirilen veritabanı sisteminizde kullandığınız MS SQL Server altına eklenmelidir (Şekil 3.13). Bu işlem için; MS SQL Server Management Studio uygulamasında Database/Attach seçeneğinden gelen pencerede eklenecek veritabanı dosyası seçilir. Bunun dışında Northwind veritabanını aynı linkte verilen "Northwind_Gene-

rate_Script.sql” dosyası ile SQL betik (script) olarak da kurulabilir. Northwind kurulum SQL betiği MS SQL Server Management Studio uygulamasında çalıştırıldığında örnek veritabanı sisteminize eklenecektir (Şekil 3.14). Farklı sürüm MS SQL Server kullanılması hâlinde örnek Northwind veritabanının SQL betik ile kurulması daha kolay olacaktır.

Şekil 3.13

Örnek Northwind Veritabanı MS SQL Server Yazılımına Ekleme



Şekil 3.14

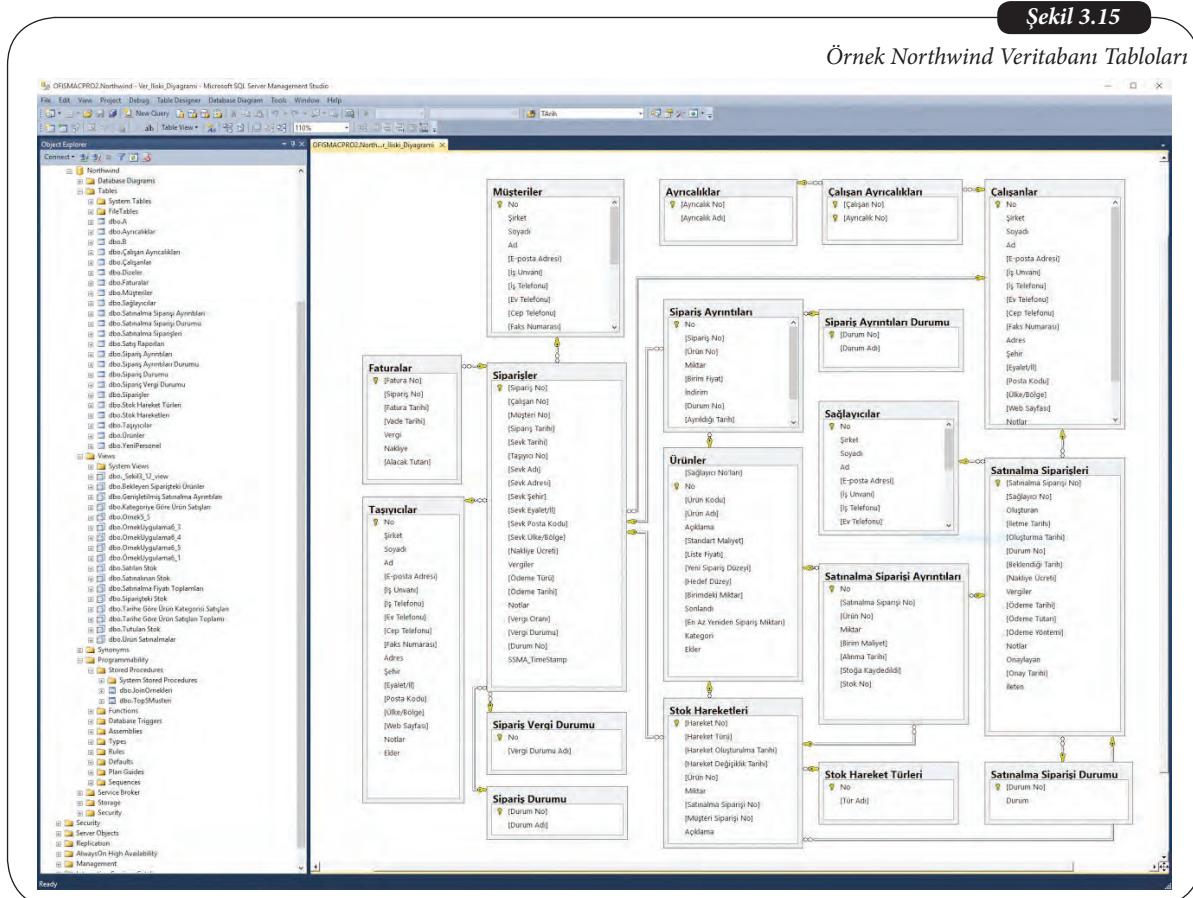
MS SQL Server Management Studio'da SQL Betik ile Örnek Northwind Veritabanı Kurulumu

```

USE [Northwind]
GO
***** Object: Table [dbo].[Ayricaliklar] Script Date: 21.05.2016 15:11:33
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Ayricaliklar](
    [Ayricalik No] [int] IDENTITY(1,1) NOT NULL,
    [Ayricalik Adi] [nvarchar](50) NULL,
    CONSTRAINT [Ayricaliklar$PrimaryKey] PRIMARY KEY CLUSTERED
        ([Ayricalik No] ASC)
)

```

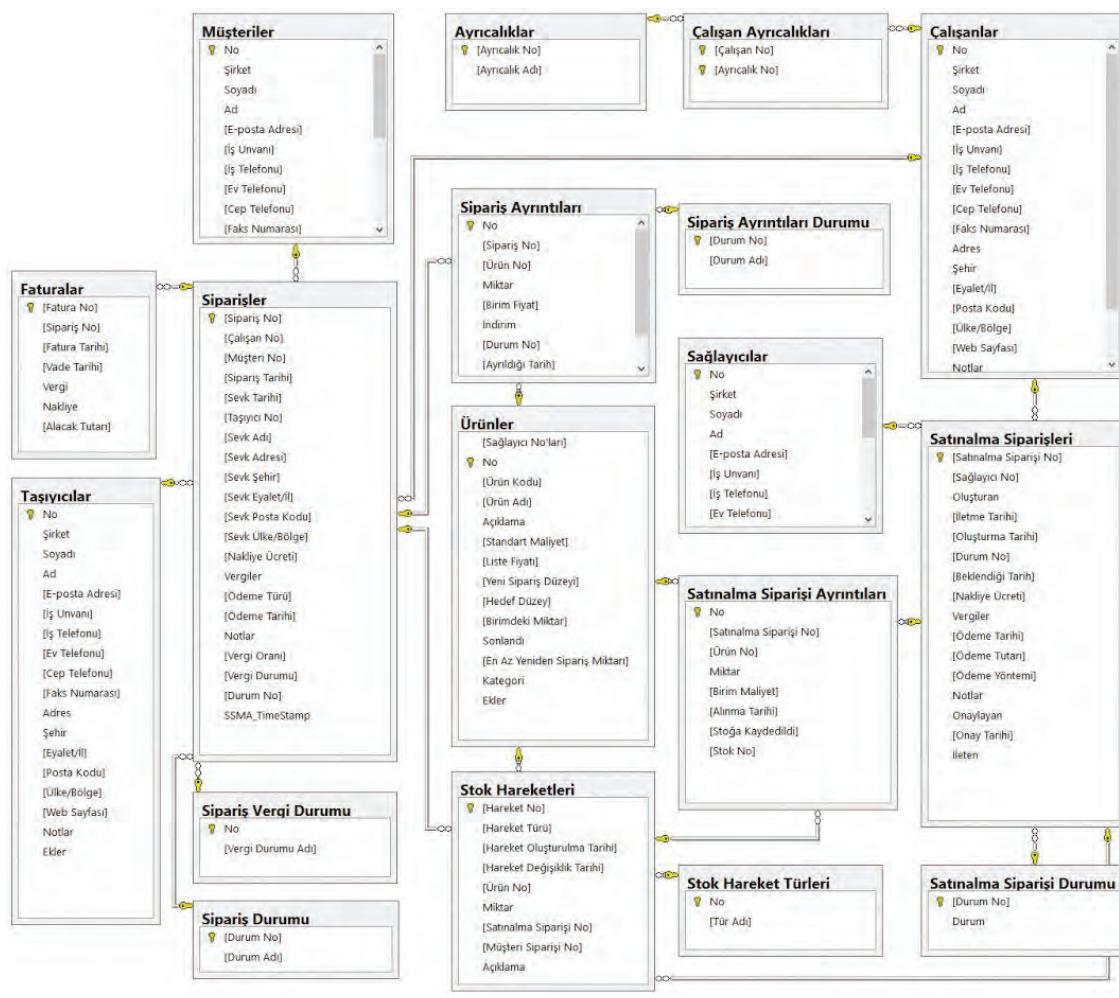
Northwind veritabanı eklendikten sonra MS SQL Server Management Studio uygulamasında veritabanının nesnelerinin (tablolar, görünümler, diyagramlar vb.) görünümü Şekil 3.15'te verilmiştir.



Örnek veritabanında tabloların yapılarının ve bağlantılarının görüldüğü veritabanı diyagramı Şekil 3.16'da verilmiştir. Bu diyagramdan da görülebileceği gibi Northwind veritabanında uluslararası bir firmanın satış işlemleri kayıt altına alınmaktadır. Tüm satış hareket kayıtları [Siparişler] tablosunda depolanırken, firma personel bilgileri [Çalışanlar], satış yapılan müşteri bilgileri [Müşteriler], satılan ürünler [Ürünler] tablolarında depolanmaktadır. Ayrıca personel, müşteri, ürün ve sağlayıcı detaylarını gösteren ilave tablolar mevcuttur. Bu tablolar arası ilişkiler de birincil ve yabancı anahtarlar oluşturularak kurulmuştur. Şekil 3.16'te oluşturulan bu bağlantılar tablolar arasındaki çizgiler olarak görülmektedir. Çizgilerin uçlarında tabloya bitişik olarak verilen anahtar simgeleri birincil anahtar (benzersiz) alanları, sonsuzluk simgesi ise ilgili alanda birden fazla kayıt olabileceğini gösterir.

Şekil 3.16

Northwind Veritabanında Tabloların ve İlişkilerin Gösterildiği Diyagram



Özet



1 İlişkisel veritabanı modelini tanımlamak

Veritabanları verilerin tutulduğu, depolandığı sistemlerdir. İlişkisel veritabanı veriyi ilişkisel örnekler (tablolar) içinde depolar. İlişkisel sözcüğü veritabanında yer alan herhangi bir kaydın tek bir konu hakkında ve sadece o konuya ilgili bilgileri içermesi gerektiğini ifade eder. İlişki örnekleri tablolar olarak adlandırılır. Tablolar kayıtlar ve alanlardan oluşur. Bir tablo içindeki kayıtların veya alanların fiziksel sırasının önemi yoktur. Bu özelliği ile veriye bilgisayardaki fiziksel depolamadan bağımsız olarak erişilebilmektedir. İlişkisel modelde ilişkiler bire bir ve bire çok olarak oluşturulabilir. Tablolar arasındaki ilişkiler anahtarlar ile belirtilir. İki tablo arasındaki ilişki bir ortak alanın eşleşen değerleri üzerinden direk olarak kurulur. Birden fazla tablodaki direk ilişkiler üzerinden dolaylı ilişkiler oluşturulur.



2 İlişkisel cebirin temel işlemlerini kullanmak

İlişkisel veritabanı modeli matematiğin iki dali olan Küme teorisi ve birinci derece yüklem aritmetiğine dayanmaktadır. İlişkisel cebirde tüm işlenen veriler ve sorguların sonuçları kümeler hâlinde ifade edilir. Geleneksel ilişkisel cebirde işlemler dört sınıfta toplanabilir:

- Tablolara uygulanan genel küme işlemleri: birleşim (union, \cup), kesişim (intersection \cap) ve fark (difference, $-$)
- Bir tablodan parçalar getiren işlemler: seçim (selection, σ), yansıtma (projection, π)
- İki tablonun kayıtlarını bitişten işlemeler: Kartezyen çarpım (Cross-product, \times), Şartlı Bitişme (Conditional Join, \bowtie_θ), Doğal bitişme (Natural Join, \bowtie), Bölme (Division, \div)
- Yeniden adlandırma (Renaming, ρ)



3 Veritabanı nesnelerinden tablo, kısıtlar, görünüm, indeks ve saklı yordamları tanımlamak

İlişkisel veritabanında veri tablolar içinde tutulur. Tablolar arasındaki bağlantılar çeşitli anahtarlar vasıtası ile oluşturulur. Tablolar alan ve kayıtlardan oluşur. Kayıt bir tabloda verilen temanın benzersiz bir örneğini temsil eder. Tablo içinde temaya uygun öğelerin her biri farklı bir satırda tutulur. Alanlarda ise tabloda tutulan temanın özellikleri tutulur. Doğru tasarılmış bir veritabanında her bir alanda sadece bir değer tutulmalıdır. Ayrıca verilerin tablolarda saklanması sırasında alan ve tablo seviyesi kısıtlar (**NOT**

NULL, **DEFAULT**, **CHECK**, Anahtarlar: birincil, benzersiz ve yabancı) veritabanı yazılımı tarafından uygulanarak verilerin tutarlılığı sağlanır. Görünüm, veritabanından bir veya daha fazla tablonun alanlarından oluşturulan sanal tablodur. Görünüm içinden bir kayıt istendiğinde veri görünümünün oluşturulduğu tablolardan getirilir, görünüm içindeki veriler ayrı bir alanda depolanmaz. Görünüm için veritabanında tutulan bilgi Görünümün yapısıdır. Indeks ve saklı yordamlar ilişkisel veritabanı öğeleri olmamakla birlikte neredeyse tüm modern ilişkisel veritabanı yönetim sistemlerince desteklenir. İndeksler tablo içinde bir değeri ararken performans artımı için kullanılır. Saklı yordamlar ise hem güvenlik ve performans sağlamak için hem de kullanıcının erişmek istediği verinin veritabanı içinde işlenerek sunulması amacıyla kullanılır.



4 Veri tiplerini ve kullanımını anlamak

Veritabanı içerisinde tablolar tasarlanırken, saklanacak her veri alanı için ihtiyacı karşılayan en uygun veri tipinin belirlenmesi gereklidir. Veri tipleri üç farklı sınıfta toplanabilir. Bunlar;

Basit veri tipleri: Karakter, Bit, Tam Sayısal, Yaklaşık Sayısal, Tarih ve Zaman

Karmaşık veri tipleri: İkili (binary) nesneler, Referans işaretçileri (pointers),

Koleksiyon diziler, Kullanıcı tanımlı veri tipleri

Özelleştirilmiş veri tipleri: xml, geography, geometry, hierarchy vb.

Kendimizi Sınayalım

Bankacılık ilişkisel veritabanında aşağıdaki 3 tablonun ve alan adlarının olduğunu varsayıyoruz. İzleyen 1-4 soruları verilen tablolara göre cevaplayınız.

Krediler (KrediNo, KrediMiktari): Krediler tablosunda müşterilere ait kredi no ve verilen kredilerin miktarları tutulur,

KrediHesabi (MusteriAdi, KrediHesapNo): KrediHesabi tablosunda kredi hesabı olan müşteri adları ve kredi hesap noları tutulur,

MevduatHesabi (MusteriAdi, MevduatHesapNo): MevduatHesabi tablosunda Mevduat hesabı olan müşteri adları ve mevduat hesap noları tutulur.

1. 2000 TL den fazla kredi alanların tamamını getiren ifade aşağıdakilerden hangisidir?

- a. $\sigma_{\text{KrediMiktari} > 2000}(\text{Krediler} \cup \text{KrediHesabi})$
- b. $\pi_{\text{KrediNo}}(\text{Krediler})$
- c. $\pi_{\text{KrediMiktari} > 2000}(\text{Krediler})$
- d. $\sigma_{\text{KrediMiktari} > 2000}(\text{Krediler})$
- e. $\sigma_{\text{Krediler} > 2000}(\text{KrediMiktari})$

2. 2000 TL den fazla kredi alanların **KrediNo**'larını getiren ifade aşağıdakilerden hangisidir?

- a. $\sigma_{\text{KrediMiktari} > 2000}(\text{Krediler})$
- b. $\pi_{\text{KrediNo}}(\sigma_{\text{Krediler} > 2000}(\text{KredilerMiktari}))$
- c. $\pi_{\text{KrediNo}}(\sigma_{\text{KrediMiktari} > 2000}(\text{Krediler}))$
- d. $\sigma_{\text{KrediNo}}(\pi_{\text{KrediMiktari} > 2000}(\text{Krediler}))$
- e. $\pi_{\text{KrediMiktari} > 2000}(\text{Krediler})$

3. Bankada kredi hesabı veya mevduat hesabı olan müşterilerin adlarını getiren ifade aşağıdakilerden hangisidir?

- a. $\pi_{\text{MusteriAdi}}(\text{KrediHesabi}) \cup \pi_{\text{MusteriAdi}}(\text{MevduatHesabi})$
- b. $\sigma_{\text{MusteriAdi}}(\text{KrediHesabi}) - \sigma_{\text{MusteriAdi}}(\text{MevduatHesabi})$
- c. $\sigma_{\text{MusteriAdi}}(\text{KrediHesabi}) \div \sigma_{\text{MusteriAdi}}(\text{MevduatHesabi})$
- d. $\pi_{\text{MusteriAdi}}(\text{KrediHesabi}) \cap \pi_{\text{MusteriAdi}}(\text{MevduatHesabi})$
- e. $\pi_{\text{MusteriAdi}}(\text{KrediHesabi}) \times \pi_{\text{MusteriAdi}}(\text{MevduatHesabi})$

4. Bankada hem kredi hesabı hem de mevduat hesabı olan müşterilerin adlarını getiren ifade aşağıdakilerden hangisidir?

- a. $\pi_{\text{MusteriAdi}}(\text{KrediHesabi}) \cup \pi_{\text{MusteriAdi}}(\text{MevduatHesabi})$
- b. $\sigma_{\text{MusteriAdi}}(\text{KrediHesabi}) - \sigma_{\text{MusteriAdi}}(\text{MevduatHesabi})$
- c. $\sigma_{\text{MusteriAdi}}(\text{KrediHesabi}) \div \sigma_{\text{MusteriAdi}}(\text{MevduatHesabi})$
- d. $\pi_{\text{MusteriAdi}}(\text{KrediHesabi}) \cap \pi_{\text{MusteriAdi}}(\text{MevduatHesabi})$
- e. $\pi_{\text{MusteriAdi}}(\text{KrediHesabi}) \times \pi_{\text{MusteriAdi}}(\text{MevduatHesabi})$

5. İlişkisel veritabanında kullanılan “tuple, ilişkisel örnek, öznitelik” kavramlarının eşdeğer ifadesi aşağıdakilerden hangisinde doğru sıra ile verilmiştir?

- a. Tablo, Kayıt, Alan
- b. Tablo, Anahtar, Kayıt
- c. Kayıt, Tablo, Alan
- d. Kayıt, Anahtar, Alan
- e. Alan, Anahtar, Kayıt

6. Aşağıdakilerden hangisi Basit veri tiplerinden biridir?

- a. Tarih ve Zaman
- b. İkili (binary) nesne
- c. Koleksiyon dizi
- d. Geography
- e. Geometry

7. I. $(12 \times 3) + 0$

II. $(\text{Null} \times 3) + 4$

III. $(12 \times 3) + \text{Null}$

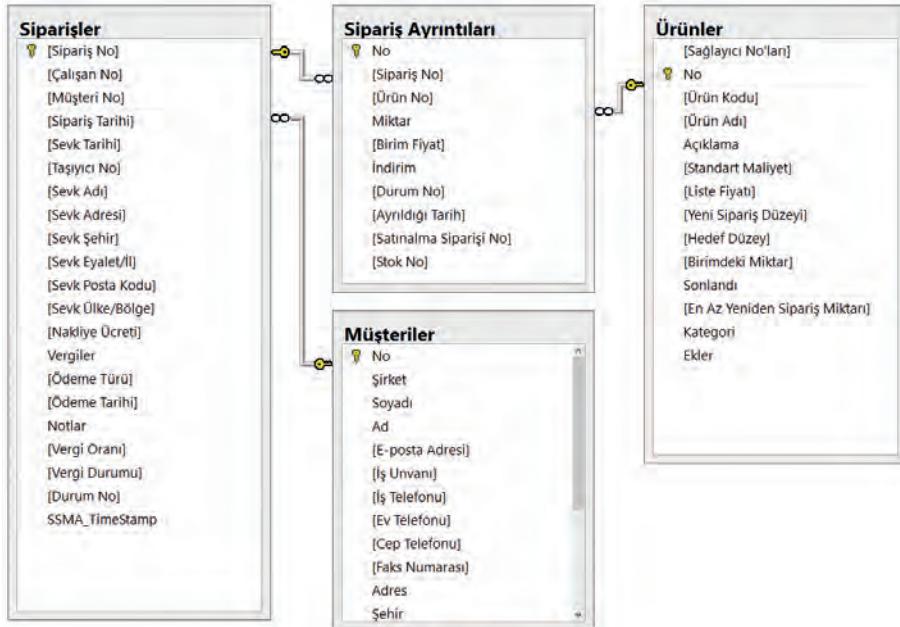
İlişkisel veritabanı işleminde yukarıda verilen ifadelerin sonucu aşağıdakilerden hangisidir?

- a. I: 36, II: 4, III: 36
- b. I: 0, II: Null, III: Null
- c. I: 36, II: 4, III: Null
- d. I: 36, II: Null, III: Null
- e. I: Null, II: Null, III: Null

8. Veritabanında bir veya daha fazla tablonun alanlarından oluşturulan sanal tabloya ne ad verilir?

- a. Hareket tablosu
- b. Görünüm
- c. Tuple
- d. Saklı Tablo
- e. Yabancı anahtar

9. ve 10. Soruları aşağıdaki tabloya göre yanıtlayınız.



9. Yukarıdaki şekilde [Siparişler] tablosunda Birincil anahtar (veya anahtarlar) hangi seçenekte doğru verilmiştir?

- a. [Sipariş No], [No]
- b. [Sipariş no],[Çalışan No]
- c. Sadece [Ürün No]
- d. Sadece [Sipariş No]
- e. Sadece [Sipariş Tarihi]

10. Yukarıdaki şekilde [Siparişler] tablosunda Yabancı anahtar (veya anahtarlar) hangi seçenekte doğru verilmiştir?

- a. [Sipariş No], [Müşteri No]
- b. Sadece [Müşteri No]
- c. [Sevk Adı]
- d. Sadece [Sipariş No]
- e. Sadece [Ürün No]

Kendimizi Sınayalım Yanıt Anahtarları

1. d Yanınız yanlış ise “İlişkisel Cebir - Seçim” konusunu yeniden gözden geçiriniz.
2. c Yanınız yanlış ise “İlişkisel Cebir - Seçim ve Yansıtma” konusunu yeniden gözden geçiriniz.
3. a Yanınız yanlış ise “İlişkisel Cebir - Birleşim” konusunu yeniden gözden geçiriniz.
4. d Yanınız yanlış ise “İlişkisel Cebir - Kesişim” konusunu yeniden gözden geçiriniz.
5. c Yanınız yanlış ise “Tablolar” konusunu yeniden gözden geçiriniz.
6. a Yanınız yanlış ise “Veri Tipleri” konusunu yeniden gözden geçiriniz.
7. d Yanınız yanlış ise “Null Değerinin Kullanımı” konusunu yeniden gözden geçiriniz.
8. b Yanınız yanlış ise “Görünümler” konusunu yeniden gözden geçiriniz.
9. d Yanınız yanlış ise “Anahtar Kısıtı” konusunu yeniden gözden geçiriniz.
10. b Yanınız yanlış ise “Anahtar Kısıtı” konusunu yeniden gözden geçiriniz.

Sıra Sizde 3

Tablonun çok parçalı [PostakoduSehir] alanı [Postakodu] ve [Sehir] olmak üzere iki ayrı alan hâlinde bölünerek tekrar tasarlanmalıdır. Böylece kullanıcı ek işlem yapmaya gerek kalmaksızın sadece [Sehir] alanı üzerinden arama yapabilir.

Sıra Sizde 4

Tablolarda tanımlanan anahtarların tipleri aşağıdaki tabloda verilmiştir.

Tablo Adı	Birincil Anahtar	Yabancı Anahtar
[Sağlayıcılar]	[No]	-
[Satınalma Siparişleri]	[Satınalma Siparişi No]	[Oluşturan], [Sağlayıcı No]
[Çalışanlar]	[No]	-

Sıra Sizde 5

Bir alanda boş değer (NULL) girilmesini önlemek için NOT NULL, tanımlanmış bir kısıtı uygulamak içinde CHECK kullanılır. Bu durumda [Urunler] tablosunda [Ürün Adı] alanı NOT NULL kısıtıyla ve [Birimdeki Miktar] alanı da CHECK (Birimdeki Miktar>0) kısıtıyla tanımlanmalıdır.

Sıra Sizde Yanıt Anahtarları

Sıra Sizde 1

Hangi ürünlerin hangi şehrre sevk edildiğine ulaşmak için [Ürün Adı] alanının bulunduğu [Ürünler] tablosu ile [Sevk Şehir] alanının bulunduğu [Siparişler] tablosunun ilişkisi bulunmalıdır. Bu iki tablo arasında ilişki [Sipariş Ayrıntı] tablosudur. [Sipariş Ayrıntı] tablosu her bir siparişe hangi üründen kaçar adet sipariş edildiği gibi ayrıntıları saklamaktadır. Dolayısıyla [Siparişler].[Sipariş No] ile [Sipariş Ayrıntı].[No], [Sipariş Ayrıntı].[Ürün No] ile [Ürünler].[Ürün No] ilişkileri kullanılarak istenilen veri kümesine ulaşılabilir.

Sıra Sizde 2

Şekil 3.4'te T $\bowtie_{A < D}$ V işleminin sonucu verilmiştir. Bu tablodan A=11 ile seçim yapılrsa aşağıdaki kayıtlar gelir.

A	T.B	T.C	V.B	V.C	D
11	2	3	2	3	12
11	2	3	2	3	14
11	2	3	4	8	21

Yararlanılan ve Başvurulabilecek Kaynaklar

Garcia-Molina H., Ullman J. D., Widom J. (2008). **Database Systems: The Complete Book, 2nd Edition**. New Jersey, Pearson Education Inc.

Gavin Powell (2006). **Beginning Database Design**, Indianapolis, Wiley Publishing, Inc.

Hernandez M.J. (2003). **Database Design for Mere Mortals™**, Second Edition. Boston, Addison-Wesley.

Ramakrishnan R., Gehrke J. (2003). **Database Management Systems**, 3rd Edition. New York, McGraw-Hill Companies Inc.
Silberschatz, A., Korth, H. F., Sudarshan, S., (2006). **Database System Concepts**, McGraw-Hill.

Yarımagañan, Ü (2010). **Veri Tabanı Yönetim Sistemleri**, Akademî Yayıncılık.

4

Amaçlarımız

- Bu üniteyi tamamladıktan sonra;
- 🕒 Masaüstü veritabanı kullanım alanlarını tanımlayabilecek,
 - 🕒 Veritabanında bulunan tablo, form, rapor nesnelerinin işlevlerini açıklayabilecek,
 - 🕒 Grafik arayüz ile basit sorgu oluşturma işlemlerini uygulayabilecek bilgi ve becerilere sahip olacaksınız.

Anahtar Kavramlar

- Masaüstü Veritabanı
- Veritabanı Yönetim Sistemleri
- Tablo
- Veri Türü
- Bire Bir İlişkisi
- Bire Çok İlişkisi
- Çoka Çok İlişkisi
- Birincil Anahtar
- Yabancı Anahtar
- Bilgi Tutarlılığı
- Form
- Raporlama
- Sorgulama

İçindekiler

Veritabanı Sistemleri

Masaüstü Veritabanı Sistemleri

- GİRİŞ
- MASAÜSTÜ VERİTABANLARI
- TABLOLAR, FORMLAR, RAPORLAR
- GRAFİK ARAYÜZ İLE VERİ SORGULAMA

Masaüstü Veritabanı Sistemleri

GİRİŞ

İnsanlar tarih boyunca bilgiyi kaydetmek, bilgiyi aktarmak ve bilgiye erişmek için çeşitli yöntemler geliştirmiştir. Matbaanın icadı ile birlikte bilgi, çoğaltılabılır ve kolay erişilebilir hâle gelmiş kısacası toplumsallaşmıştır. İnsanlar bilgiye ulaşıkça bu bilgileri saklama, depolama ihtiyacı duymuşlardır. 20. Yüzyıla kadar kitaplar bu bilgilerin depolanmasında yeterli olsa da bilginin çoğalması ve bilgisayarların ortaya çıkması ile birlikte bilgi dijital ortamlarda saklanmaya başlanmıştır. İçinde bulunduğuuz bilgi çağında, bilgiyi üretme, kaydetme, arşivleme, koruma, paylaşma ve bilgiden yararlanma biçimlerinin farklılaşması, bilgi ve iletişim teknolojilerinde gerçekleşen değişim ve gelişimden kaynaklanmaktadır.

Veri, saklamaya ihtiyaç duyduğumuz her türlü bilgidir. Veritabanı ise birbirile ilişkili her türlü bilginin belli bir sistematik ile biraraya gelmesi ile oluşan bilgi topluluğudur. Veritabanını daha basit şekilde tanımlayacak olursak, kendi amacınız doğrultusunda düzenlemiş olduğunuz bilgi topluluğudur diyebiliriz. Örneğin iyi bir müzik arşivine sahip olmak ya da kitaplığınızdaki kitapların bilgilerini tutmak hatta cep telefonunuzda oluşturduğunuz kişilerin listesi bir çeşit veritabanı oluşturmak anlamına gelmektedir.

Veritabanları, çok büyük miktardaki bilgileri depolamada geleneksel yöntem olan dosya işleme sistemlerine alternatif olarak geliştirilmiştir. Basitten karmaşığa pek çok bilgi veritabanlarında depolanabilir. Günlük hayatımızda kullandığımız arama motorları, alışveriş sitelerine üyelik sistemleri, internet bankacılığı sistemleri veritabanı kullanımına örnek olarak verilebilir. Bir veritabanı oluşturabilmek için veritabanı yönetim sistemi yazılımlarına ihtiyaç duyulur ve veritabanları bu yazılımlar aracılığıyla yönetilir.

Veritabanları Sistemleri kitabından bu ünitesinde, masaüstü veritabanı kavramı üzerinde duracağız. Veritabanlarının olmazsa olmaz nesneleri tablolar, formlar, raporlar MS Access veritabanı yazılımindan yararlanılarak nasıl oluşturulur ve bu nesneler veritabanı için ne ifade eder sorularına cevap arayacağız. Son olarak herhangi bir programlama diline ihtiyaç duymadan oluşturulan sorgular hakkında bilgi vereceğiz.

MASAÜSTÜ VERİTABANLARI

Veritabanı, birbiri ile ilişkili veri dosyaları ve veri tabloları kümесinin çok amaçlı kullanıma olanak sağlayacak şekilde depolanmasıdır.

İlişkisel veritabanı yönetim sistemi, verilerinizi tanıma, onlarla çalışma ve başlıklar ile paylaşma konularını tümüyle denetlemenizi sağlar. Sitemin, Veri Tanımlama, Veri Yönetimi ve Veri Denetimi olmak üzere 3 ana becerisi vardır.

Belirli bir amaca göre düzenlenmiş kayıt ve dosyaların tümü **veritabanını** oluşturur. Verileri bilgisayar ortamında saklamadan yollarından biri olan veritabanlarını işlemek amacıyla Veritabanı Yönetim Sistemi (DBMS-Database Management System) yazılımları geliştirilmiştir. Kullanıcılar veritabanı oluşturmak ve oluşturdukları veritabanını sorunsuz yönetmek amacıyla Veritabanı Yönetim Sistemi yazılımlarından faydalnamaktadır.

Modern veritabanı yönetim sistemleri, **ilişkisel veritabanı yönetim sistemini** (Relational Database Management System - RDBMS) kullanırlar. İlişkisel veritabanı yönetim sistemlerinde, veritabanındaki her kayıt bir tek konu hakkındadır ve yalnızca o konuya ilişkili bilgileri içerir. Sistem tüm verileri tablolar içerisinde yönetir. Bu tablolar arasındaki veriler, çeşitli anahtarlar aracılığı ile birbirlerine bağlanır. Tablolarda, sütunlar arasında bir anahtar sütun yer alır ve bu anahtar sütun aracılığı ile birden çok tablo verileri birbirileyle bağlantı sağlayabilir, herhangi bir sorgulamada birlikte görüntülenebilir.

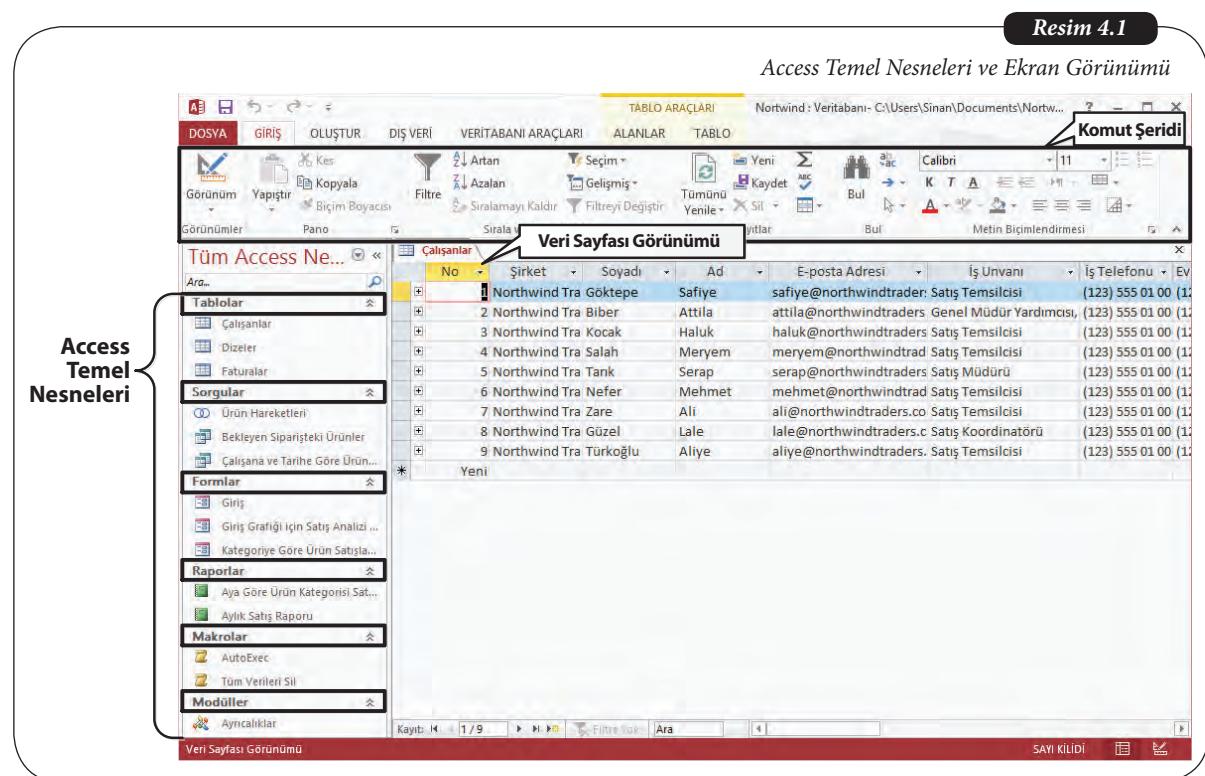
Veritabanı yazılımlarını sunucu veritabanı yönetim sistemleri ve masaüstü veritabanı yönetim sistemleri olarak iki gruba ayırmak mümkündür. Sunucu veritabanı yönetim sistemleri karmaşık ve pahalı sistemlerdir. Kullanımı üst düzey bilgi ve beceri gerektirir. Masaüstü veritabanı yönetim sistemleri ise basit ihtiyaçların karşılanması için kullanılan, kişisel bilgisayarlarda çalışan ve elde edilmesi kolay olan ucuz sistemlerdir. Karmaşık olmayan verilerin depolanması amacı ile rahatça kullanılabilir. Masaüstü veritabanı yönetim sistemi yazılımlarını kullanmak için üst düzey bilgi ve beceriye ihtiyaç yoktur.

Masaüstü veritabanı yönetim sistemi yazılımlarında hiç program kodu kullanmadan veritabanı hazırlamak mümkündür. Bu yazılımları kullanarak herhangi bir programcılık bilgisine sahip olmayan bir kullanıcı kendi veritabanı dosyalarını kolayca hazırlayabilir. Programcılık bilgisi olanlar ise hazırladıkları veritabanlarına kod yazarak daha ileri düzey işlemleri yapabilmektedir.

MS Access Veritabanı Yazılımı

Tablolar, sorgular, formlar, raporlar gibi Access veritabanı nesneleri arayüzün sol tarafındaki Gezinti Bölmesinde listelenir ve nesneler arasında geçiş bu bölümden sağlanır. Gezinti bölümünden bir veritabanı nesnesi seçildiğinde Access penceresinde görüntülenir ve seçilen nesne türü menüde işaretlenerek komut şeridi nesne türüne göre özelleşir.

Microsoft Access, veritabanı yönetim sistemleri arasına çok sonradan girmiş olmasına rağmen en yaygın kullanılan masaüstü veritabanı yönetim sistemleri yazılımlarından biridir. 90'lı yılların başında ortaya çıkan MS Access, ilk masaüstü ilişkisel veritabanı yönetim sistemidir. MS Access'in bu kadar yaygın kullanılmasının en önemli nedeni küçük ölçekli veritabanları için çok kullanışlı bir yazılım olması ve yazılım araçlarının yüksek kullanıcı kolaylığına sahip olmasıdır. MS Access'in içerisindeki *temel nesneler*, tablolar, sorgular, formlar, raporlar, veri erişim sayfaları, makrolar ve modüllerdir. Access'te bulunan nesnelerin bir çoğu şablonlar ve sihirbazlar yardımıyla kolayca hazırlanabilmektedir. MS Access'in temel nesneleri ve yazılımın arayüzü Resim 4.1'de yer almaktadır.



Veritabanı yönetim sistemleri yazılımları ile programcılık bilgisine gerek kalmadan veritabanları kullanılsa da veritabanı oluşturulmadan önce tasarımının iyi yapılması gereklidir. Veritabanının hangi amaçlar için oluşturulacağı, kimler tarafından kullanılacağı akla gelecek ilk sorulardır. Veritabanında hangi konular hakkında bilgi saklamaya gereksinim duyulacağı (tablolar) ve her konu için hangi bilgilerin saklanması gereki (tablodaki alanlar) belirlenmelidir. Veritabanında gereksinim duyulan tablolar ve alanlar belirlendikten sonra, alanlar her kayıttı benzersiz değerlerle tanımlanmalıdır.

Veritabanında yer alan tabloların birbiri ile doğru ilişkilendirilmesi çok önemlidir. Bir tablodaki verilerin başka tablolardaki verilerle ilişkilendirilmesinde herhangi bir sorun çıkarsa, tablolara alan ekleyerek ya da yeni tablolar oluşturarak veritabanı tasarımının gözden geçirilmesi gereklidir. Doğru tasarlanmış bir veritabanında gereksiz ya da tekrarlanan bilgiler yer almaz. Yinelenen bilgiler gereksiz yer kaplayacağı gibi hata ve tutarsızlık ihtimallerini de artırır. Veritabanında yer alan bilgilerin doğru ve tam olması bu bilgiler kullanılarak oluşturulacak sorguların ve raporların doğru bilgi barındırmamasını (çıktı üretmesini) sağlar. Aksi takdirde yanlış bilgilerle yanlış sonuçlara ulaşılması kaçınılmazdır.

Tasarımı iyi yapılarak MS Access'te oluşturulan bir veritabanı dosyası kolayca yönetilebilir. Microsoft Access'te veriler tablolar hâlinde saklanır. Tablo verileri çevrimiçi formlar kullanılarak görüntülenebilir. Tablolara yeni veri eklenebilir ve veriler güncelleştirilebilir. Veritabanında belirli bir kriterde uygun veriler aranırsa sorgular kullanılarak bu veriler süzülebilir. Raporlar kullanılarak veriler çözümlenebilir ve belirli bir düzende yazdırılabilir. Veri erişim sayfaları oluşturularak kullanıcılar veritabanlarındaki verileri internetten veya bir intranetten görüntüleyebilir, güncelleştirebilir ya da çözümleyebilir.

MS Access yazılımının bu kadar yaygın kullanılan bir masaüstü veritabanı sistemi olmasının nedeni ne olabilir?



SIRA SİZDE

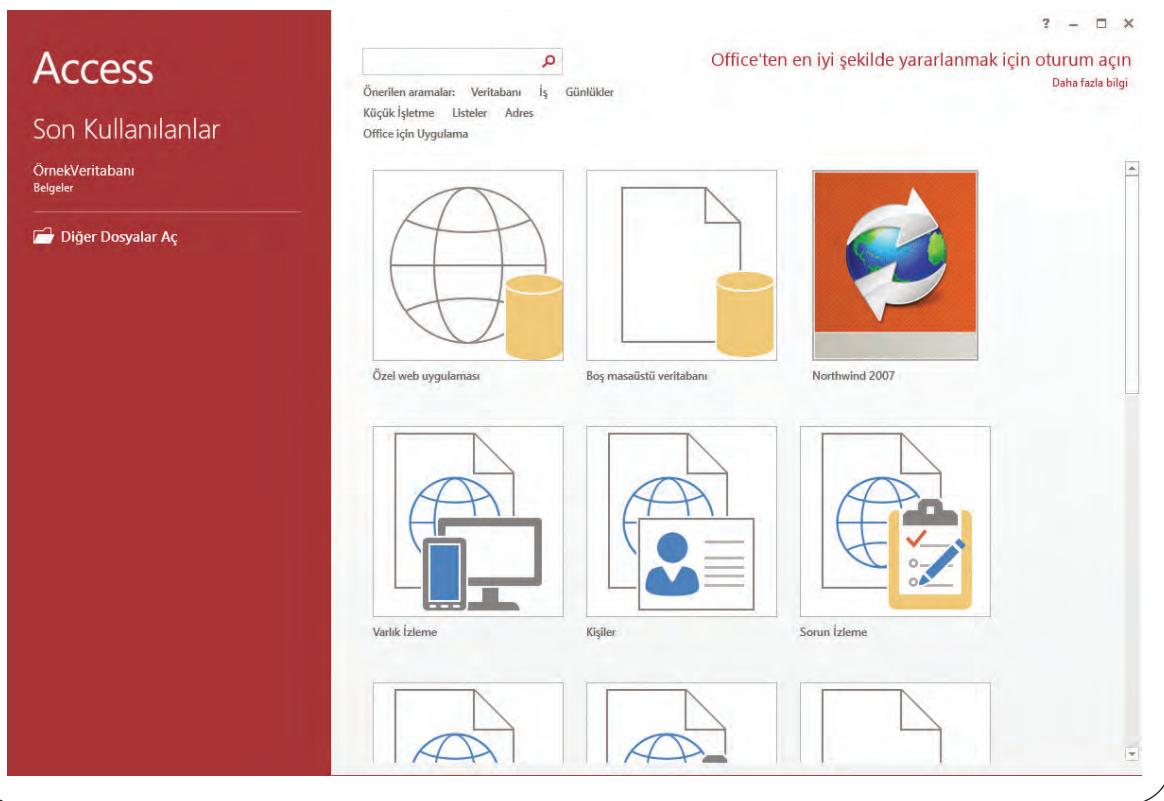
1

MS Access'te Masaüstü Veritabanı Oluşturma

Windows görev çubuğunun sol ucunda yer alan **Başlat** düğmesine tıklanarak görüntülenen menüden Microsoft Access seçeneğini işaretleyerek Access programını başlatabilirsiniz. Program açıldığında Resim 4.2'de gösterildiği gibi bir ekran karşınıza çıkar.

Resim 4.2

Microsoft Access Açılmış Ekranı



Bu açılış ekranında *Son Kullanılanlar* ve *Diğer Dosyaları Aç* olmak üzere iki farklı seçenek görüntülenir. Açılmak istenilen veritabanı *Son Kullanılanlar* bölümünde listeleniyorsa bu bölümden seçim yapılarak, listede görünmüyorsa *Diğer dosyaları aç* seçeneği ile daha önce oluşturulmuş tüm veritabanı dosyalarına erişilerek veritabanı dosyası açılabilir.

Yeni bir veritabanı dosyası oluşturulacaksa açılış ekranının solundan *Yeni* seçeneği işaretlenerek *Boş Masaüstü veritabanı* ya da veritabanı şablonlarından biri seçilebilir. Yeni veritabanı dosyası *Boş masaüstü veritabanı* seçeneği ile oluşturulduğunda Resim 4.3'te gösterildiği gibi veritabanı dosyasının adlandırılmasını isteyen Dosya adı kutusu açılır. Dosya adı kutusuna masaüstü veritabanı dosyasının adı yazıldıktan sonra konum seçilecek *Oluştur* simgesine tıklanmalıdır.

Masaüstü veritabanı dosyası oluşturulduğunda gezinti bölümünde Tablo 1 olarak adlandırılmış boş bir Tablo veri sayfası görüntülenir. Bundan sonraki işlem adımı açılan bu tabloya veri girişi yapmak ya da başka bir kaynaktan verileri alarak yapıştmaktır (Resim 4.4).

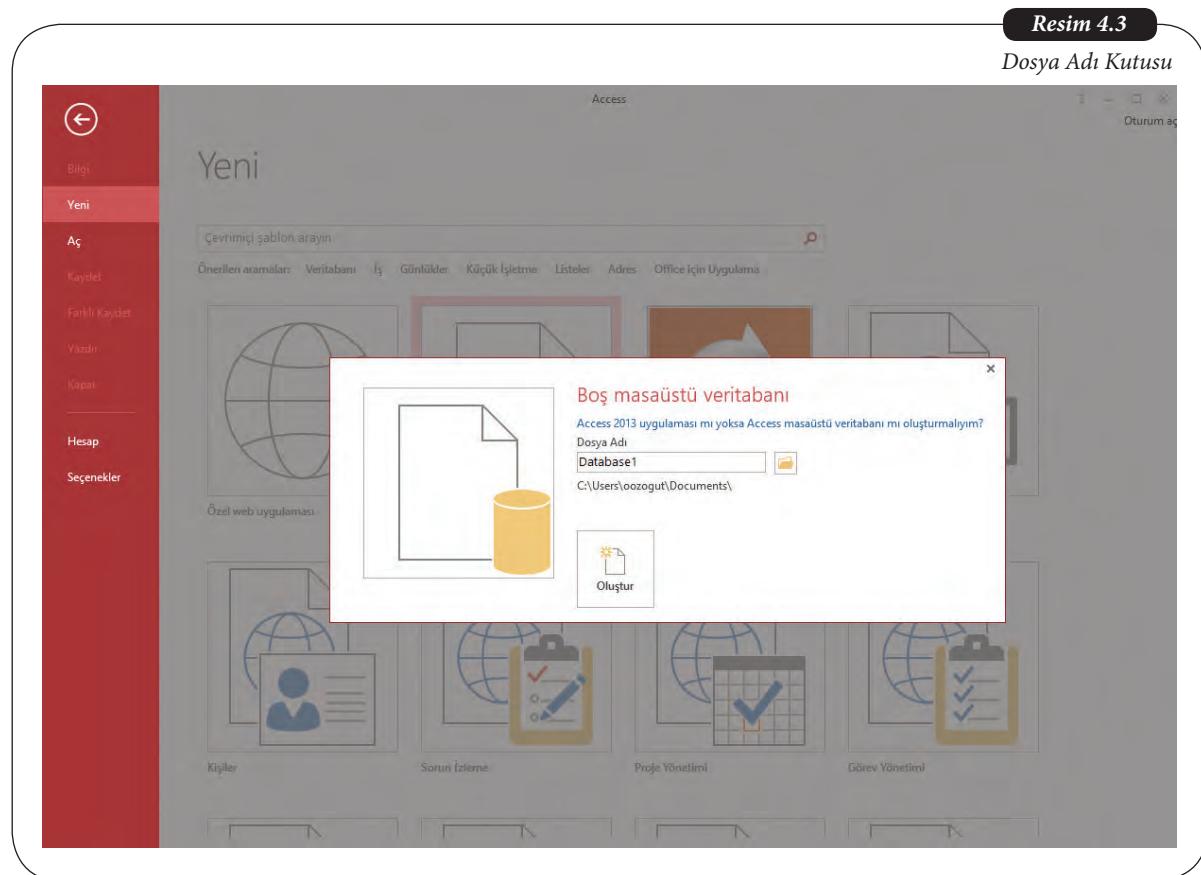
DİKKAT



Microsoft Access, oluşturulan masaüstü veritabanı dosya uzantısını .accdb olarak kaydeder.

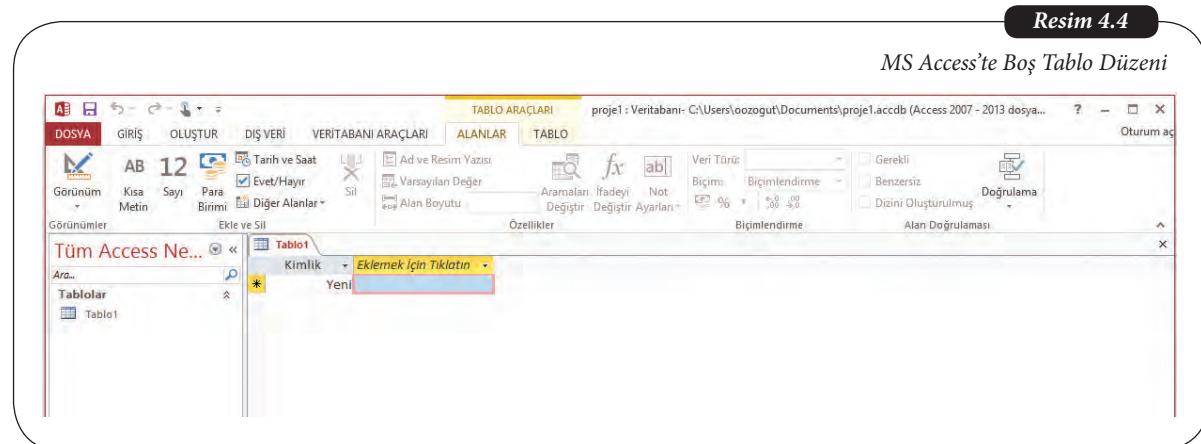
Resim 4.3

Dosya Adı Kutusu



Resim 4.4

MS Access'te Boş Tablo Düzeni



TABLOLAR, FORMULAR, RAPORLAR

Veritabanı sistemlerinin tümünde tablolar, formlar ve raporlar yer alır. Raporların ve formların oluşturulması MS Access veritabanı sistemi dışındaki veritabanlarında daha üst düzey bilgi gerektirir. Bu bölümde veritabanı nesnelerinden olan tablolar, formlar ve raporlar MS Access veritabanı sisteminde yer alan Northwind örnek veritabanı kullanılarak anlatılmıştır.

Bu ünitede örnek olarak kullanılan Northwind MS Access veritabanına yeni Access dosyası oluşturma ekranında aşağıdaki simge kullanılarak ulaşılabilir.



Northwind 2007

Tablolar

Tablolar belirli konular hakkındaki verilerin toplandığı bölümdür ve bir veritabanının en gerekli nesneleridir. Her konu için ayrı bir tablo oluşturmak işlemlerin daha hızlı yapılabilmesini ve verilere daha kolay erişilebilmesini sağlar. Veri girişlerindeki hata yapma oranını azaltarak karmaşık veri yiğinlarında çalışmayı kolaylaştır.

Bir veritabanı tablosu görünüm olarak elektronik tabloya benzer. Veriler satır ve süntular içerisinde saklanır. Bu nedenle elektronik tabloları veritabanı tablolarına almak oldukça kolaydır. Tablolar, verileri belli bir düzen içerisinde kaydeder. Tablodaki her satır bir kayıt her sütun ise bir alan olarak kabul edilir. Kayıtlar, bağımsız bilgi parçalarının saklandığı alanlarda bulunur. Örneğin, **Çalışanlar** isimli bir tablonuz varsa her kayıt (satır) farklı bir çalışanın bilgilerini içerir ve her alanda (sütunda) ad, soyad, e-posta adresi, telefon numarası vb. gibi farklı bilgi türleri bulunur. Resim 4.5'te örnek tablo görünümü verilmiştir.

Resim 4.5

Örnek Tablo Görünümü

No	Sirket	Soyadi	Ad	E-posta Adresi	İş Ünvanı	İş Telefonu	Ev Telefonu
1	Northwind Tra Göktepe	SafİYE	safие@northwindtrader.	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	
2	Northwind Tra Biber	Attila	attila@northwindtrader.	Genel Müdür Yardımcısı,	(123) 555 01 00	(123) 555 01 02	
3	Northwind Tra Kocak	Haluk	haluk@northwindtrader.	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	
4	Northwind Tra Salah	Meryem	meryem@northwindtrad.	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	
5	Northwind Tra Tank	Serap	serap@northwindtraders.	Satış Müdürü	(123) 555 01 00	(123) 555 01 02	
6	Northwind Tra Nefer	Mehmet	mehmet@northwindtrad.	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	
7	Northwind Tra Zare	Ali	ali@northwindtraders.co	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	
8	Northwind Tra Güzel	Lale	lale@northwindtraders.c	Satış Koordinatörü	(123) 555 01 00	(123) 555 01 02	
9	Northwind Tra Türkoglu	Aliye	aliye@northwindtraders.	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	

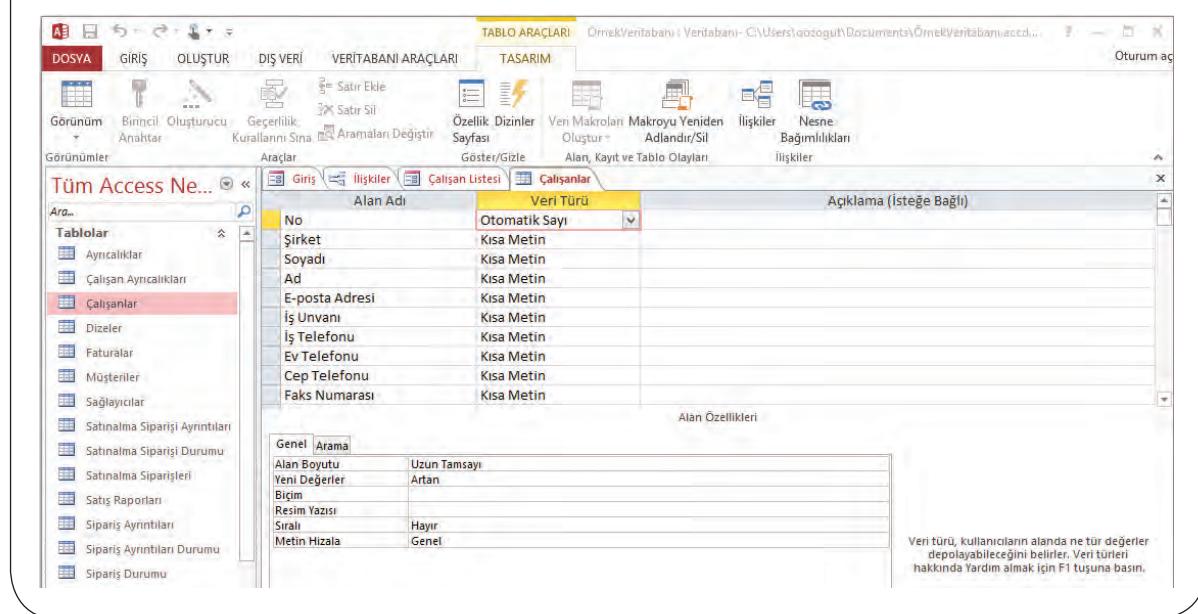
Alanda bulunan tüm değerlere uygulanan ve bu değerlerin ne tür veriler olabileceğini belirleyen bir dizi niteliğe veri türü denir. Alanların; metin, tarih, sayı gibi belli bir veri türü olarak atanması gereklidir. Access'te on farklı veri türü vardır:

- Ek:** Dijital fotoğraflar gibi dosyalar. Her kayda birden çok dosya eklenebilir.
- Otomatik Sayı:** Her kayıt için otomatik olarak üretilen numaralar.
- Para Birimi:** Para değerleri.
- Tarih/Saat:** Tarihler ve saatler.
- Köprü:** E-posta adresleri, web sayfaları gibi bağlantılar.
- Not:** Uzun metin blokları ve metin biçimlendirilmesinin kullanıldığı metinler. Örneğin adresler ya da ayrıntılı ürün açıklamaları.
- Sayı:** Sayısal değerler. Para birimleri için ayrı bir veri türü olduğunu unutmayın.
- OLE Nesnesi:** Word belgeleri gibi OLE (Object Link and Embedding–Bağlı ve gömülü nesneler) nesnesi.
- Metin:** Ad-Soyad ya da posta adresi gibi kısa alfasyasal değerler.
- Evet/Hayır:** 0 ya da 1 değeri olarak saklanan boolean değer.

Resim 4.6'da Çalışanlar tablosundaki alanların veri türleri tasarım görünümünde gösterilmektedir.

Resim 4.6

Tasarım Görünümünde Alan Veri Türleri



Veritabanında bulunan nesneler Veri sayfası görünümü ve Tasarım görünümü olmak üzere iki şekilde görüntülenebilir. Veri sayfası görünümünde, daha çok yeni girdiler oluşturulmasına olanak sağlarken, Tasarım görünümünde, kullanılan nesnenin tasarnımı ve biçimini ile ilgili düzenlemeler yapılabilir.



DİKKAT

Tabloları tasarım görünümünde açmanın kullanıcı açısından ne gibi yararları vardır?



SIRA SİZDE

Access veritabanına tablo eklemenin çeşitli yolları vardır. Yeni bir veri kaynağı varsa yeni tablo oluşturularak veritabanına eklenebilir ve alanlar tanımlanarak veri girişi yapılabılır. Excel çalışma kitabı, Word belgesi, metin dosyası, web hizmeti ya da başka bir veritabanındaki tablo veritabanında kullanılmak istenirse bu kaynaklar veritabanına aktarılabilir ya da bağlanabilir.

Access'te tabloların yapısı dinamiktir ve üzerinde işlem yapılabilir. Veritabanında daha önce oluşturulmuş bir tablonun alan veri türü değiştirilebileceği gibi ihtiyaç duyulursa yeni alan da eklenebilir. Veritabanında oluşturulan her tabloya içeriğine göre bir ad verilmelidir. Tabloların içeriğe göre adlandırılması hangi bilgilerin hangi tabloda olabileceği kolay bulmayı sağlar. Access'te tablo isimleri en fazla 64 karakter uzunluğu ile sınırlıdır.

Tablo İlişkileri

Veritabanındaki her tablonun belirli bir konu hakkında verileri depoladığını daha önce belirtmişük. İlişkisel veritabanlarında farklı tablolarda yer alan veriler başka bir tabloyu ilgilendirebilir. Bu durumda farklı tablolarda depolanan verileri bağlamak için tablolar arasında ilişkiler oluşturulur. İki tablo arasında ilişki oluşturabilmek için tabloların ortak bir alana sahip olması gereklidir. Örneğin, öğrenci bilgilerinin olduğu *Öğrenciler* tablosu ile sınav notlarının yer aldığı *Notlar* tablosu arasında ilişki kurabilmek için iki tabloda da öğrenci numarası gibi bir ortak alanın yer olması gereklidir.

Ortak alanların olduğu iki tablo arasında kurulan mantıksal bağlantıya **İşki** adı verilir.

Tablolarda ortak alanların olması ilişkilerin tanımlanabilmesini sağlar ve aynı anda birden çok tablodan bilgi görüntüleyen sorgular, formlar ve raporlar kolayca oluşturulabilir. Access'te bire bir ilişkisi, bire çok ilişkisi ve çoka çok ilişkisi olmak üzere üç çeşit tablo ilişkisi vardır.

Bire Bir İlişkisi

İki tablo arasındaki bire bir ilişkisinde birincil tablodaki her kaydın birincil anahtar değeri ilişkili tabloda bir ve yalnızca bir kayda ait eşleşen alan veya alanlardaki değere karşılık gelir. Yani bir tablonun bir kaydı ile diğer tablonun bir kaydı eşleştirilebilir. Örneğin bir otelde konaklayan bir müşterinin bir yatağa yerleştirilmesi bire bir ilişkisidir. Böylece Otel müşterilerinin bilgilerinin yer aldığı tablonun bir kaydı ile otelin yatak bilgilerinin bulunduğu tablonun bir kaydı ilişkilendirilmiş olur.

Bire Çok İlişkisi

İki tablo arasındaki bire çok ilişkisinde birincil tablodaki her kaydın birincil anahtar değeri ilişkili tabloda birden çok kaydın eşleşen alanı veya alanlarındaki değere karşılık gelir. Bir başka deyişle bir tablonun bir kaydı diğer tablonun birden fazla kaydı ile eşleştirilebilir. Otel örneğini düşünecek olursak bir odaya birden fazla müşteri yerleştirilebilir ancak bir müşterinin birden fazla odaya yerleştirilmesi mümkün değildir.

Çoka Çok İlişkisi

İki tablo arasındaki çoka çok ilişkisinde, tablolardan birindeki bir kayıt diğer tabloda birden çok kayıtlı ilişkilendirilebilir. Otel örneğine devam edecek olursak bir otel çalışanı birden fazla müşteriye hizmet verebilirken, bir otel müsterisi de birden fazla otel çalışanından hizmet alabilir.

Anahtarlar

Anahtarlar, tablo ilişkisinin parçası olan alanlardır ve ilişkisel veritabanlarında tablolar anahtarlarla birbirine bağlanır. Anahtarlar genellikle tek alandan oluşsa da bazen birden fazla alandan da oluşabilir. Access'te birincil anahtar ve yabancı anahtar olmak üzere iki çeşit anahtardan sözedilebilir.

Birincil Anahtar

Birincil anahtar, değeri veya değerleri bir tablodaki her kaydı benzersiz olarak tanımlayan bir veya daha çok alan (sütun) olarak tanımlanabilir. Birincil anahtarda her zaman bir değer olmalıdır. Birincil anahtar Null (geçersiz, boş) değerlere izin vermez ve her zaman benzersiz bir dizine sahip olmalıdır. Birincil anahtar bir tabloyu diğer tablolardaki yabancı anahtarlarla ilişkilendirmek için kullanılır.

Tablolarda birincil anahtar tanımlaması zorunlu değildir.

SIRA SİZDE

3

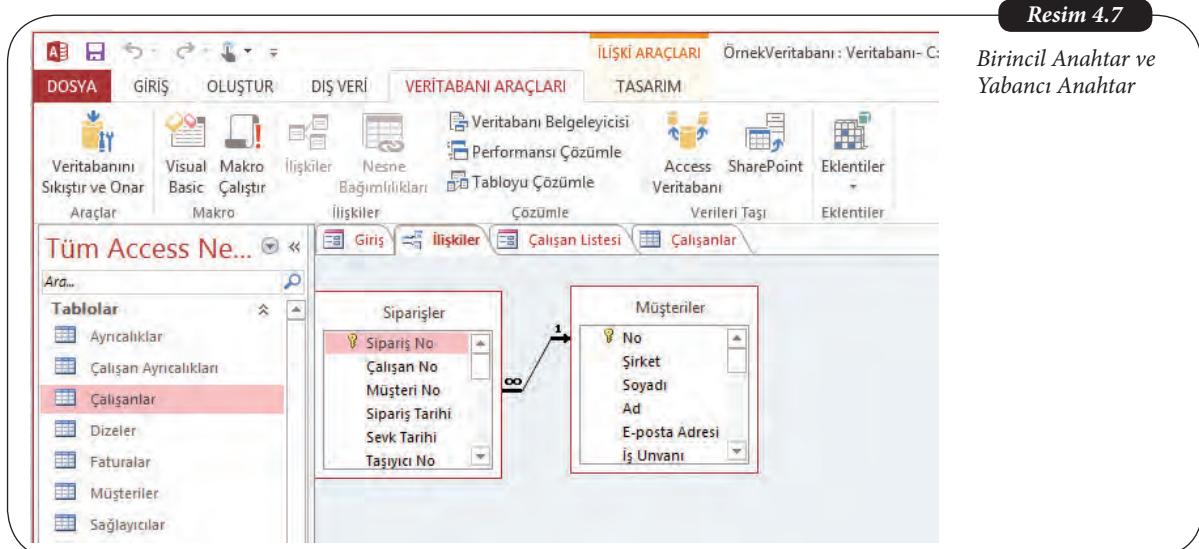
İki tablo birbiri ile ilişkilendirilirken, birinci tabloda birincil anahtar olmayan ve benzersiz dizine sahip olmayan bir alan, ikinci tabloda birincil anahtar olmayan ve benzersiz dizine sahip olmayan bir alana sürüklendirirse nasıl bir ilişki oluşur?

Yabancı Anahtar

İki tabloyu ilişkilendirebilmek için ortak alana ihtiyaç duyduğunu belirtmişlik. Ortak alan bir tabloda birincil anahtar ise diğer tabloda yabancı anahtar olarak isimlendirilir. Başka bir deyişle yabancı anahtar, bir tablonun birincil anahtar değerlerinin başka bir tabloda karşılık gelen değerlerini içerir. Bir tabloda bir veya daha fazla yabancı anahtar bulunabilir.

Örneğin, öğrenci bilgilerinin olduğu *Öğrenciler* tablosu ile sınav sonuçlarının yer aldığı *Notlar* tablosunu ele alalım. Öğrenciler tablosunda yer alan öğrenci numarası Öğrenciler tablosunun birincil anahtarı olsun. Notlar tablosunda ise birden fazla dersten alınan sınav sonuçları ve öğrenci numarası yer alınsın. Notlar tablosunda her sınav sonucu, Öğrenciler tablosundaki bir kayda karşı gelen öğrenci numarası ile ilişkilenecektir. Bu durumda Notlar tablosundaki öğrenci numarası, Öğrenci tablosunun birincil anahtarı iken Notlar tablosunun yabancı anahtarıdır.

Örnek veritabanında birbirile ilişkili iki tablo olan Siparişler ve Müşteriler tablosundaki ilişkiler Resim 4.7'de gösterilmiştir. Buna göre Müşteriler tablosundaki No alanı, Müşteriler tablosunun birincil anahtarı iken Siparişler tablosunun *Müşteri No* alanı Müşteriler tablosunun yabancı anahtarıdır. Birincil anahtar olarak tanımlanan alan adının yanında anahtar simgesi görünürken yabancı anahtar böyle bir simge içermez.



Resim 4.7

Birincil Anahtar ve
Yabancı Anahtar

Bilgi Tutarlılığı

Bilgi tutarlılığı, bir tabloya kayıtlar girildiğinde ya da silinmek istendiğinde tablolar arasında tanımlanan ilişkileri korumak için izlenen kurallardır. Bilgi tutarlılığının zorunlu kılınması, bilgi tutarlılığını zedeleyen tüm işlemlerin Access tarafından reddedilmesini sağlar. Buna göre Access, bir başvurunun hedefini değiştiren güncelleştirmeleri ve bir başvurunun hedefini kaldırın silme işlemlerini yapmaz.

İlişkili tablolarda verilerin ayrı tutulması tutarlılığı, verimliliği ve anlaşılırlığı artırır.

Formlar

Access'te veriler tablolarda saklanır, ancak tüm bu verilerle çalışmanın en kolay ve en yaygın yol bir form kullanmaktadır. Formlar, Access veritabanında yer alan nesnelerden biridir. Veritabanında depolanan tablolara veri eklemek, düzenlemek ya da görüntülemek amacıyla kullanılırlar. Formlar, verilerle çalışmaya yönelik kullanımını kolay bir biçim sunan arayüzlerdir. Formlara komut düğmeleri gibi işlevsel öğeler eklenebilir. Bu düğmeler yardımıyla formda hangi verilerin görüntüleneceği belirlenebilir, diğer formları veya raporları açmak ve çeşitli görevleri gerçekleştirmek üzere programlanabilir.

Veritabanı oluşturulurken veriler, form kullanılmadan tablo veri sayfalarında düzenlenebilir ancak çoğu veritabanı kullanıcı tablodaki verileri görüntülemek, düzenlemek ve tablolara veri girmek için formları kullanmayı tercih etmektedir.

Diğer kullanıcıların veritabanındaki verilerle etkileşimi denetlemek için Formlar kullanılır. Access masaüstü veritabanı birden fazla kullanıcı tarafından kullanılacaksa, verimlilik ve veri girişi doğruluğu açısından formların iyi tasarlanmış olması çok önemlidir. Örneğin, sadece belirli alanları gösteren ve belli işlemlerin gerçekleştirilemesine izin veren bir form oluşturulabilir. Böylece veriler korunabilir ve gerekli şekilde girilmesi sağlanabilir.

Boş Form Oluşturma

Denetim veya önceden biçimlendirilmiş öğe bulunmayan bir form oluşturmak için, *Oluştur* sekmesinde *Boş Form* seçilmelidir. Bu işlem yapıldığında Access, Düzen görünümünde boş bir form açar ve Alan Listesi bölmescini göründür. *Alan Listesi* bölgesinde, formda görmek istenilen alanları içeren tablo ya da tabloların yanındaki artı işaretü (+) seçilir. Forma alan eklemek için, alana çift tıklanmalıdır ya da forma sürüklenmelidir. Forma aynı anda birden fazla alan eklemek için CTRL tuşu basılı tutularak alanları seçmek ve forma sürüklemek gerekir.

DİKKAT



Forma logo, başlık, sayfa numarası, tarih ve saat bilgisini eklemek için *Form Düzen Araçları* sekmesinin *Denetimler* grubundaki araçlar kullanılır. Forma farklı denetimler eklenmek istenirse, *Tasarım* seçilerek *Denetimler* grubundaki araçlar kullanılır.

Bölünmüş Form Oluşturma

Bölünmüş form, form görünümü ve Veri Sayfası görünümü olmak üzere aynı anda verilerin iki görünümünü birden sunar. Bölünmüş formlarla çalışırken tek bir formda iki form türü aynı anda kullanılabilir. Örneğin, formun veri sayfası kısmı kayıtları hızla bulurken, kaydı görüntülemek veya düzenlemek için form kısmı kullanılabilir. İki görünüm de aynı veri kaynağına bağlıdır ve her zaman birbirile eşitlenmiş durumdadır (Resim 4.8).

Resim 4.8

Bölünmüş Form Örneği

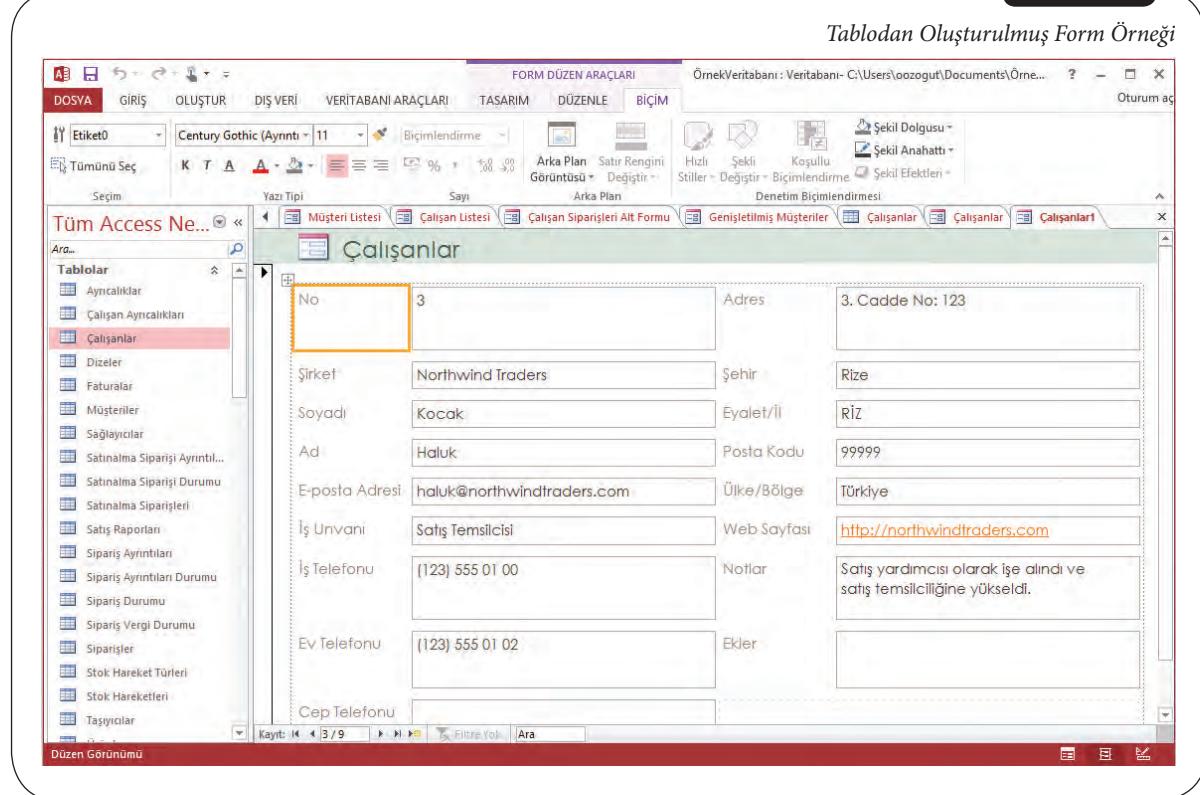
No	Şirket	Soyadı	Ad	E-posta Adresi	İş Unvanı	İş Tipi
1	Northwind Tr Göktepe	Göktepe	SafİYE	safие@northwindtrac	Satış Temsilcisi	(12)
2	Northwind Tr Biber	Biber	Attila	attila@northwindtrac	Genel Müdür Yardımcı	(12)
3	Northwind Tr Kocak	Kocak	Haluk	haluk@northwindtrac	Satış Temsilcisi	(12)
4	Northwind Tr Salah	Salah	Meryem	meryem@northwindtrac	Satış Temsilcisi	(12)
5	Northwind Tr Tarkan	Tarkan	Serap	serap@northwindtrac	Satış MüdürÜ	(12)
6	Northwind Tr Nefer	Nefer	Mehmet	mehmet@northwindtrac	Satış Temsilcisi	(12)
7	Northwind Tr Zare	Zare	Ali	ali@northwindtraders	Satış Temsilcisi	(12)
8	Northwind Tr Güzel	Güzel	Lale	lale@northwindtrade	Satış KoordinatörÜ	(12)
9	Northwind Tr Türkoglu	Türkoglu	Aliye	aliye@northwindtrac	Satış Temsilcisi	(12)

Bölünmüş Form aracını kullanarak yeni bir bölünmüş form oluşturmak için, Gezinti Bölmesi’nde verileri içeren tabloyu veya sorguyu seçerek *Oluştur* sekmesinde *Diğer Formlar*’ı ve ardından da *Bölünmüş Form*’u işaretleyin. Access formu oluşturur ve bu formda tasarım değişiklikleri yapılmasına izin verir. Örneğin, gereklisse, metin kutularının boyutunu verilerin sağa doğru şekilde ayarlayabilirsiniz.

Tablo ya da Sorgudan Form Oluşturma

Veritabanındaki bir tablodan ya da sorgudan form oluşturmak için, Gezinti Bölmesi'nden oluşturulacak forma ilişkin verileri içeren tablo ya da sorgu seçilerek *Oluştur* sekmesinde *Form* işaretlenmelidir. Bu işlemin sonucunda Access bir form oluşturarak *Düzen* görünümünde açar. İhtiyaç duyulan tasarım değişiklikleri bu ortamda yapılabilir (Resim 4.9).

Resim 4.9



Birden Çok Kayıt Görüntüleyen Form Oluşturma

Sürekli form olarak da bilinen çok öğe formudur. Birden çok kayıt görüntüleyen ama veri sayfasına göre daha fazla özelleştirilebilir olan bir form istiyorsanız, Çoklu Öğe aracını kullanabilirsiniz. Gezinti Bölmesi'nden oluşturulacak forma ilişkin verileri içeren tablo ya da sorgu seçilerek *Oluştur* sekmesinde *Diğer Formlar* ve *Birden Çok Öğe* işaretlenmelidir. Düzen görünümünde açılan form üzerinde gerekli tasarım değişiklikleri yapılabilir.

Alt Form İçeren Form Oluşturma

Aynı ayrı tablolarda depolanan ilişkili verilerle çalışırken, birden fazla tablo ya da sorgudan alınan verileri aynı form üzerinde görüntülemek için alt formlar kullanılır. Alt form, başka bir formun içine eklenen formdur. Birincil forma ana form adı verilir.

Alt formlar, özellikle bire çok ilişkisi bulunan tablolardan veya sorgulardan alınan veriler görüntülenmek istendiğinde etkilidir. Bire çok ilişkisi, birincil tablodaki her bir kaydın birincil anahtar değerinin, ilişkili tabloda birçok kaydın eşleşen alanındaki veya alanlarındaki değere karşılık geldiği iki tablo arasındaki ilişki olarak tanımlanır. Örneğin öğrenci verilerini görüntüleyen ve her öğrencinin kayıt yaptırdığı derslerden aldığı sınav sonuçlarını gösteren alt formu bulunan bir form oluşturulabilir. Öğrenciler tablosundaki veriler ilişkinin "bir" tarafını oluşturur. Notlar tablosundaki verilerse ilişkinin "çok" tarafıdır (her öğrenci birden fazla derse kayıt yaptırarak sınav olabilir).

Resim 4.10

Alt Form İçeren Form Örneği

Kaynak: <https://support.office.com/tr-TR/article/Alt-form-ekleme-c59387f9-73aa-4082-82cb-5d29ef04299f>

Resim 4.10'da bir şirkette Çalışanlar ile Siparişler tablosu kullanılarak oluşturulmuş form görüntülenmektedir. Ana formda (1) çalışan bilgisi yer alırken, alt formda (2) ilgili çalışanın aldığı siparişler görüntülenmektedir.

Daha önce oluşturulmuş bir forma alt form eklemek için;

1. Gezinti Bölmesi'nde varolan formu seçerek farenin sağ tuşuna tıklayın ve *Tasarım Görünümü*'nü seçin.
2. Tasarım sekmesinin *Denetimler* grubunda aşağı açılan oka tıklayarak *Denetimler* galerisini görüntüleyin ve *Denetim Sihirbazları Kullan* öğesini seçin. Tasarım sekmesinin *Denetimler* galerasında *Alt Form/Alt Rapor* düğmesini seçin.
3. Formda, alt formu yerleştirmek istediğiniz yeri seçin ve sihirbazdaki önerileri takip edin.

DİKKAT



Alt form, yalnızca ana formdaki geçerli kayıtları ilişkili veriler görüntülenecek şekilde bağlanmalıdır. Aksi takdirde alt formda geçerli kayıtları ilişkili olmayan veriler de görüntülenir.

Gezinti Formu Oluşturma

Access'te veritabanındaki çeşitli formlar ve raporlar arasında geçiş yapmayı kolaylaştıran Gezinti Denetimi bulunur. Gezinti formu en basit tanımlıla Gezinti Denetimi içeren formdur. Gezinti formu oluşturmak için, gezinti formu eklemek istediğiniz veritabanını açarak **Oluştur** sekmesinin **Formlar** grubunda **Gezinti**'yi ve istediğiniz gezinti formu stilini seçin.

DİKKAT



Gezinti formları, webde yayınlanması planlanan veritabanına mutlaka eklenmelidir. Çünkü Access Gezinti Bölmesi, tarayıcıda görüntülenmez.

Raporlar

Raporlar, veritabanında yer alan verilerin, yazdırılabilen bir görünümde düzenlenmiş hâlidir. Bir rapor genellikle belli bir soruya yanıt verecek şekilde düzenlenir. **Raporlar** verileri tablolardan alabilir ancak hazırlanacak rapor tek bir tablodaki verilerin düzenlenmesi ile elde edilemiyorsa sorgular yolu ile verilerin diğer tablolardan alınması gereklidir.

Raporlar üzerinde bulunan her nesnenin boyutu ve görünümü üzerinde düzenleme yapılarak bilgiler istenilen biçimde görüntülenebilir. Access veritabanında oluşturulan bir raporda aşağıdaki düzenlemeler yapılabilir.

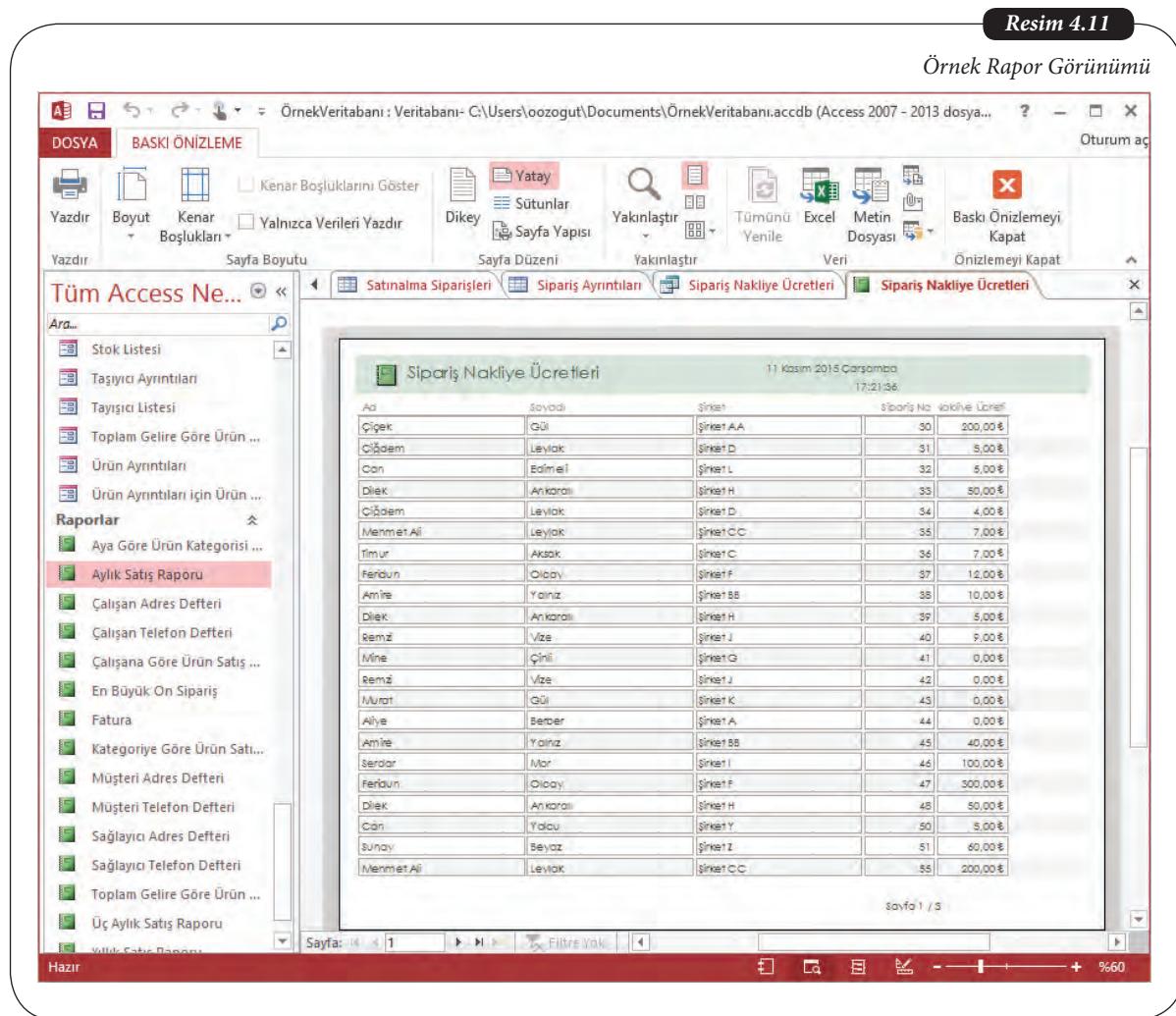
- Gruplandırma, sıralama veya toplamlar ekleme
- Koşullu biçimlendirmeyle verileri vurgulama
- Renk ve yazı tiplerini özelleştirme
- Logo veya arka plan görüntüsü ekleme

Resim 4.11'de Siparişler ve Müşteriler tablosu kullanılarak oluşturulmuş *Sipariş Nakliye Ücretleri* raporu görüntülenmektedir.

Raporlar genellikle çıktı olarak alınabilecek şekilde biçimlendirilir, Ekran'da görüntülenebilir, başka bir programa verilebilir ya da e-posta iletişi olarak gönderilebilir.

Resim 4.11

Örnek Rapor Görünümü



GRAFİK ARAYÜZ İLE VERİ SORGULAMA

Sorgular, veritabanındaki verileri farklı şekillerde görüntülemek, yalnızca belirli alanlarda çalışma yapmak ve verileri çözümlemek için kullanılır. **Sorgular**, veritabanının en önemli Özelliğidir ve sorgularla pek çok farklı işlev gerçekleştirilebilir. Sorguların en sık kullanılan işlevi, tablolardan belirli verilerin bulunup, alınmasıdır. Görüntülenmek istenen veriler çeşitli tablolara dağılmış durumdaysa, sorgular sayesinde bu veriler tek bir veri sayfasında görüntülenebilir. Bir başka deyişle sorgular, bir veya birden fazla tablodaki belirli verileri alarak bu verileri istenilen şekilde sıralamamıza olanak sağlar.

Sorgular, belirtlen koşullara uyan verileri bulmaya ve almaya yardımcı olduğu gibi birden çok kaydı aynı anda güncelleştirmek, silmek ya da özel hesaplamalar yapmak için de kullanılır. Access'te seçme sorguları ve eylem sorguları olmak üzere iki temel **sorgu türü** vardır. Seçme sorgusu, veriyi bulup alır ve kullanımına hazır hâle getirir. Sorgunun sonuçları ekranada görüntülenebilir, yazdırılabilir ya da kopyalanabilir.

Eylem sorgusu, verilerle ilgili bir görev gerçekleştirir. Eylem sorguları yeni tablolar oluşturmak, var olan tablolara veri eklemek, verileri güncelleştirmek veya silmek amacıyla kullanılabilir. **Eylem sorguları** ile ilgili ayrıntılı bilgiye kitabınızın 7. Ünitesinde yer verilmiştir.

Sorgular, tablolarda depollanmış verileri özel bir görünümle bize sunan nesnelerdir. Sorgu sonuçları genellikle formlara ve raporlara yönelik kayıt kaynağı olarak hizmet verir.

Sorgu Türleri;

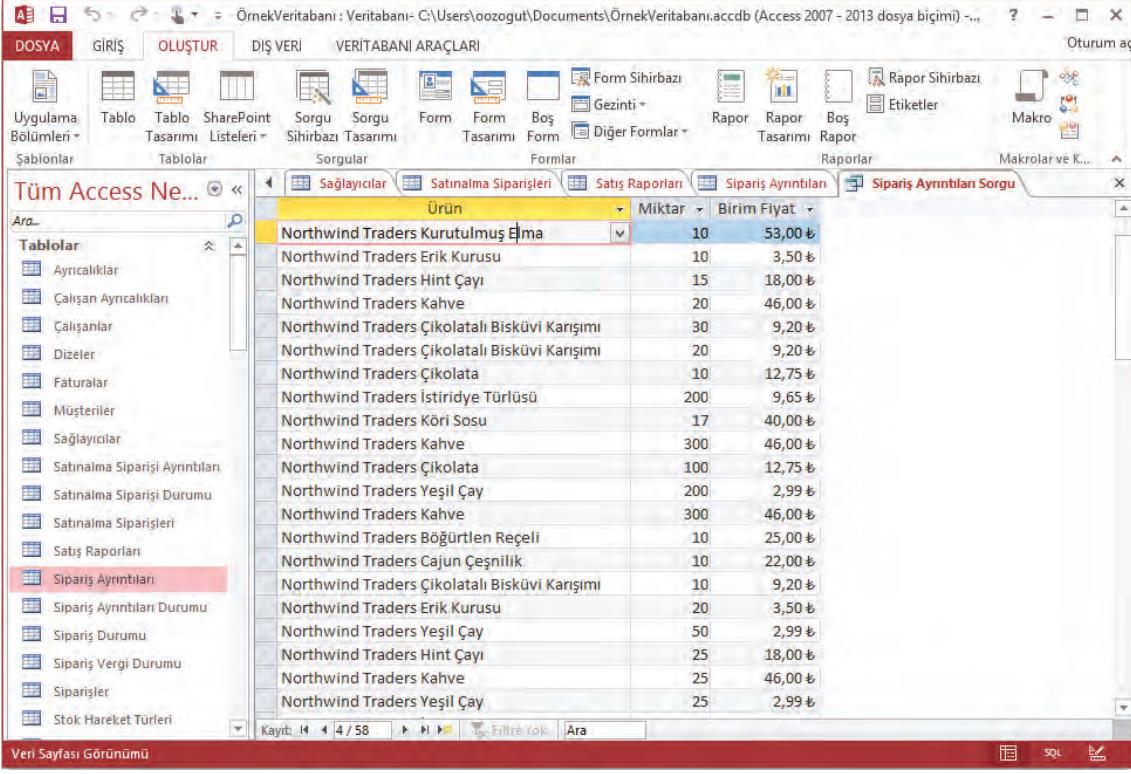
- Seçme Sorgusu,
- Eylem Sorgusu,
- Çapraz Sorgu,
- Parametre Sorgusu,
- SQL Sorgu.

Eylem Sorguları;

- Tablo oluşturma Sorgusu,
- Ekleme Sorgusu,
- Güncelleştirme Sorgusu,
- Silme Sorgusu.

Seçme sorgusu yalnızca gereksinim duyulan verileri Veri Sayfası görünümünde almaya yardımcı olur. Seçme sorguları *Sorgu Sihirbazı* ya da *Sorgu Tasarımı* kullanılarak oluşturulabilir. İlk olarak örnek veritabanı dosyasından *Sipariş Ayrıntıları* tablosunu kullanarak yalnızca ürün adlarını, ürün miktarlarını ve ürün fiyatlarını görüntülemek için *Sorgu Sihirbazı*'nı kullanarak bir seçme sorgusu oluşturalım. *Oluştur* sekmesinde, *Sorgu Sihirbazı*'nı seçin ve sihirbazın adımlarını takip edin. Sihirbazı tamamladığınızda oluşturduğunuz sorgu Veri Sayfası görünümünde Resim 4.12'de olduğu gibi görünecektir.

Resim 4.12

Sipariş Ayrıntıları Sorgusu


The screenshot shows the Microsoft Access 2013 ribbon with the 'OLUŞTUR' (Create) tab selected. The main area displays a table titled 'Sipariş Ayrıntıları' with columns: Ürün (Product), Miktar (Quantity), and Birim Fiyat (Unit Price). The table contains 28 rows of data from the Northwind Traders database. The 'Satış Raporları' tab is also visible in the ribbon.

Ürün	Miktar	Birim Fiyat
Northwind Traders Kurutılmış Elma	10	53,00 ₺
Northwind Traders Erik Kuruşu	10	3,50 ₺
Northwind Traders Hint Çayı	15	18,00 ₺
Northwind Traders Kahve	20	46,00 ₺
Northwind Traders Çikolatalı Bisküvi Karışımlı	30	9,20 ₺
Northwind Traders Çikolatalı Bisküvi Karışımlı	20	9,20 ₺
Northwind Traders Çikolata	10	12,75 ₺
Northwind Traders İstiridye Türlüsü	200	9,65 ₺
Northwind Traders Köri Sosu	17	40,00 ₺
Northwind Traders Kahve	300	46,00 ₺
Northwind Traders Çikolata	100	12,75 ₺
Northwind Traders Yeşil Çay	200	2,99 ₺
Northwind Traders Kahve	300	46,00 ₺
Northwind Traders Bögürten Reçeli	10	25,00 ₺
Northwind Traders Cajun Çeşnilik	10	22,00 ₺
Northwind Traders Çikolatalı Bisküvi Karışımlı	10	9,20 ₺
Northwind Traders Erik Kuruşu	20	3,50 ₺
Northwind Traders Yeşil Çay	50	2,99 ₺
Northwind Traders Hint Çayı	25	18,00 ₺
Northwind Traders Kahve	25	46,00 ₺
Northwind Traders Yeşil Çay	25	2,99 ₺

SIRA SİZDE



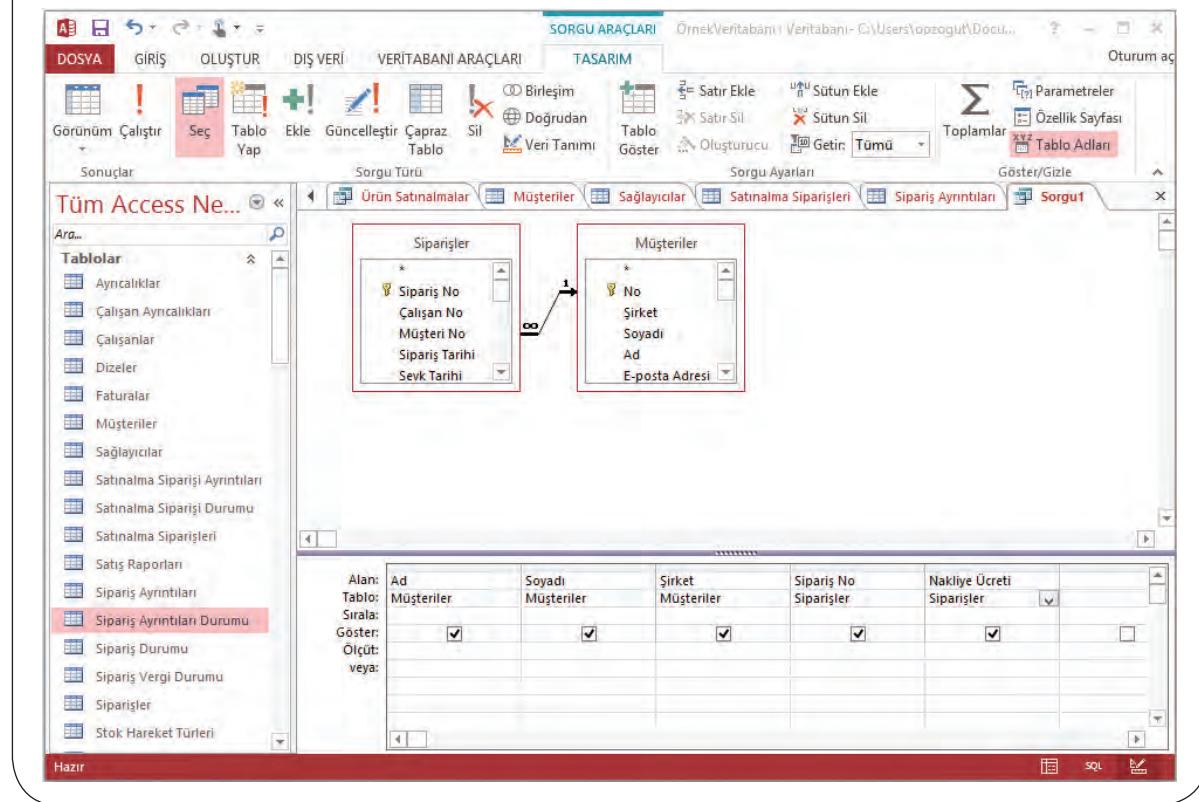
4

Resim 4.12'de görüntülenen *Sipariş Ayrıntıları* sorgusunda kaç tane alan vardır ve alan isimleri nelerdir?

Şimdi de Sorgu Tasarımı düğmesini kullanarak örnek veritabanında başka bir seçme sorgusu oluşturalım. Oluşturacağımız sorgu Müşteriler ve Siparişler tablolarını kullanarak alınan siparişlerin nakliye ücretlerini listelesin. *Oluştur* sekmesinde, *Sorgu Tasarımı* düğmesini seçin. *Tabloyu Göster* iletişim kutusundan *Müşteriler* ve *Siparişler* tablolarını ekleyerek iletişim kutusunu kapatın. Oluşturulan sorguda alanlar hangi sıra ile gösterileceğse Resim 4.13'te gösterildiği gibi Tasarım Kılavuzunda tablo ve alanları belirleyein.

Resim 4.13

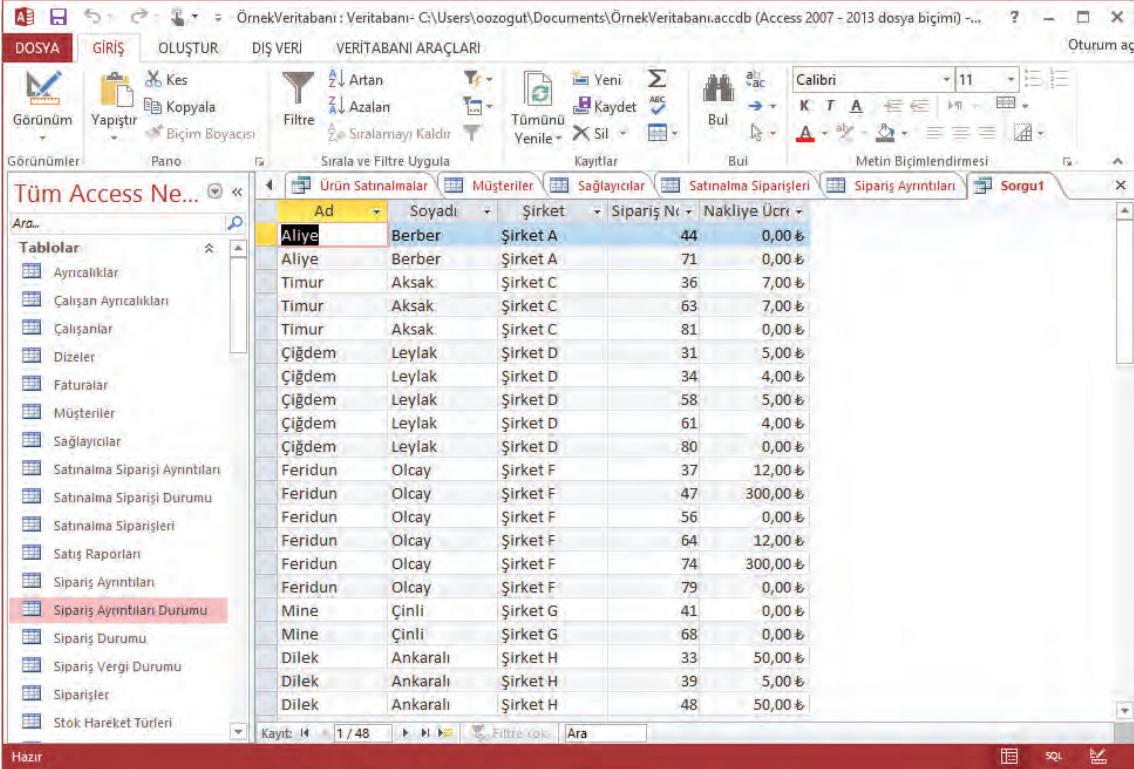
Tasarım Görünümünde Seçme Sorgusu Oluşturma



Tasarım sekmesindeki *Çalıştır* düğmesine tıkladığınızda veri sayfası görünümünde sorgulama sonuçları Resim 4.14'de verildiği gibi görüntülenir. Sorgu tamamlandıktan sonra kapatmak isterseniz sorguya isim vermenizi isteyen bir iletişim kutusu açılır. Sorguya *Sipariş Nakliye Ücretleri* ismini vererek sorguyu adlandırdığınızda, *Sipariş Nakliye Ücretleri* sorgusun Access'in Sorgular nesnesinin altındaki sorgu listesine eklenir.

Resim 4.14

İki Tablodan Alanlar Alınarak Oluşturulan Seçme Sorgusu



The screenshot shows the Microsoft Access interface with a query design grid. The title bar indicates the database is 'ÖrnekVeritabanı.accdb'. The ribbon tabs include 'DOSYA', 'GİRİŞ' (selected), 'OLUŞTUR', 'DİS VERİ', and 'VERİTABANI ARAÇLARI'. The left pane shows the 'Tablolar' (Tables) list with various tables listed, and 'Sipariş Ayrıntıları Durumu' is highlighted with a red box. The main area displays a query grid with the following data:

Ad	Soyadı	Şirket	Sipariş No	Nakliye Ücreti
Aliye	Berber	Şirket A	44	0,00 ₺
Aliye	Berber	Şirket A	71	0,00 ₺
Timur	Aksak	Şirket C	36	7,00 ₺
Timur	Aksak	Şirket C	63	7,00 ₺
Timur	Aksak	Şirket C	81	0,00 ₺
Çiğdem	Leylak	Şirket D	31	5,00 ₺
Çiğdem	Leylak	Şirket D	34	4,00 ₺
Çiğdem	Leylak	Şirket D	58	5,00 ₺
Çiğdem	Leylak	Şirket D	61	4,00 ₺
Çiğdem	Leylak	Şirket D	80	0,00 ₺
Feridun	Olcay	Şirket F	37	12,00 ₺
Feridun	Olcay	Şirket F	47	300,00 ₺
Feridun	Olcay	Şirket F	56	0,00 ₺
Feridun	Olcay	Şirket F	64	12,00 ₺
Feridun	Olcay	Şirket F	74	300,00 ₺
Feridun	Olcay	Şirket F	79	0,00 ₺
Mine	Çinli	Şirket G	41	0,00 ₺
Mine	Çinli	Şirket G	68	0,00 ₺
Dilek	Ankaralı	Şirket H	33	50,00 ₺
Dilek	Ankaralı	Şirket H	39	5,00 ₺
Dilek	Ankaralı	Şirket H	48	50,00 ₺

DİKKAT



Birden fazla tablo kullanarak seçme sorgusu oluşturulacaksa, tabloların birbiri ile ilişkili olması gereklidir.

Sipariş Nakliye Ücretleri sorgusunu kullanarak belli bir nakliye ücretinin üzerinde olan siparişleri listeleyebiliriz. Bunun için sorguyu tasarım görünümünde açtıktan sonra, tasarım kılavuzunda *Nakliye Ücretleri* alanının altında bulunan Ölüt satırına ≥ 50 ölçüyü eklenirse ve sorgu çalıştırılırsa, nakliye ücreti 50 TL ve daha yüksek olan siparişler listelenir.

Özet



Masaüstü veritabanı kullanım alanlarını tanımlamak Veritabanı, birbiri ile ilişkili veri dosyaları ve veri tabloları kümesi veya sisteminin çok amaçlı kullanıma olanak sağlayacak şekilde depolanmasıdır. İlişkisel veritabanı yönetim sistemlerinde, veritabanındaki her kayıt bir tek konu hakkındadır ve yalnızca o konuya ilişkili bilgileri içerir. Sistem tüm verileri tablolar içerisinde yönetir. İlişkisel veritabanı yönetim sistemlerinin Veri Tanimlama, Veri Yönetimi ve Veri Denetimi olmak üzere 3 ana becerisi vardır.

Masaüstü veritabanı yönetim sistemleri, basit ihtiyaçların karşılanması için kullanılan, kişisel bilgisayarlarla çalışan ve elde edilmesi kolay olan ucuz sistemlerdir. Masaüstü veritabanı yönetim sistemi yazılımlarını kullanmak için üst düzey bilgi ve beceriye ihtiyaç yoktur. MS Access, ilk masaüstü ilişkisel veritabanı yönetim sistemidir ve küçük ölçekli veritabanları için çok kullanışlı bir yazılımdır. MS Access'te bulunan nesnelerin çoğu şablonlar ve sihirbazlar yardımıyla hazırlanabilmektedir. Veritabanında yer alan tabloların birbiri ile doğru ilişkilendirilmesi çok önemlidir. Tasarımı iyi yapılarak MS Access'te oluşturulan bir veritabanı dosyası kolayca yönetilebilir. Microsoft Access'te veriler tablolar hâlinde saklanır. Tablo verileri çevrimiçi formlar kullanılarak görüntülenebilir. Tablolara yeni veri eklenebilir ve veriler güncellenebilir. Veritabanında belirli bir kriterde uyan veriler aranırsa sorgular kullanılarak bu veriler sızulebilir. Raporlar kullanılarak veriler çözümlenebilir ve belirli bir düzende yazdırılabilir. Veri erişim sayfaları oluşturularak kullanıcılar veritabanlarındaki verileri internetten veya bir intranetten görüntüleyebilir, güncelleştirebilir ya da çözümleyebilir.



Veritabanında bulunan tablo, form, rapor nesnelerinin işlevlerini açıklamak

Tablolar, belirli konular hakkındaki verilerin toplandığı bütündür. Her konu için ayrı bir tablo oluşturmak hata yapma oranını azaltır. Bir veritabanı tablosu görünüm olarak elektronik tabloya benzer. Veriler satır ve sütunlar içerisinde saklanır. Tablodaki her satır bir kayıt her sütün ise bir alan olarak kabul edilir. Tablolarda ortak alanların olması ilişkilerin tanımlanabilmesini sağlar ve aynı anda birden çok tablodan bilgi görüntüleyen sorgular, formlar ve raporlar kolayca oluşturulabilir. Access'te bire bir ilişkisi, bire çok ilişkisi ve çokça çok ilişkisi olmak üzere üç çeşit tablo ilişkisi vardır. İlişkisel veritabanlarında tablolar, anahtarlarla birbirine bağlanır. Anahtarlar genellikle tek alandan oluşsa da bazen birden fazla alandan da oluşabilir. Access'te birincil anahtar ve yabancı anahtar olmak üzere iki çeşit anahtardan sözedilebilir. Birincil anahtar, tablodaki her kaydı bengersiz olarak tanımlayan bir veya daha çok alandır. Birincil anahtarda her zaman bir değer olmalıdır. Birincil anahtar geçersiz ya da boş değer alamaz. Birincil anahtar bir tabloyu diğer tablolardaki yabancı

anahtarlarla ilişkilendirmek için kullanılır. Yabancı anahtar, bir tablonun birincil anahtar değerlerinin başka bir tabloda karşılık gelen değerlerini içerir. Bir tablo da bir veya daha fazla yabancı anahtar bulunabilir.

Formlar, verilerle çalışmaya yönelik kullanımı kolay bir şekilde sunan arayüzlerdir. Veritabanında depolanan tablolara veri eklemek, düzenlemek ya da görüntülemek amacı ile kullanılır. Formlara komut düğmeleri gibi işlevsel öğeler eklenebilir. Bu düğmeler yardımcıyla formda hangi verilerin görüntüleneceği belirlenebilir, diğer formları veya raporları açmak ve çeşitli görevleri gerçekleştirmek üzere programlanabilir. Diğer kullanıcıların veritabanındaki verilerle etkileşimi denetlemek için de Formlar kullanılır. Access'te aşağıda listelenen form türleri kullanılabilir.

- Boş Form Oluşturma
- Bölünmüş Form Oluşturma
- Tablo ya da Sorgudan Form Oluşturma
- Birden Çok Kayıt Görüntüleyen Form Oluşturma
- Alt Form İçeren Form Oluşturma
- Gezinti Formu Oluşturma

Raporlar, veritabanında yer alan verilerin, yazdırılabilceğin görünümde düzenlenmiş hâlidir. Ekranda görüntülenebilir, başka bir programa verilebilir ya da e-posta iletisi olarak gönderilebilir. Bir rapor genellikle beli bir soruya yanıt verecek şekilde düzenlenir. Raporlar verileri tablolardan alabilir ancak hazırlanacak rapor tek bir tablodaki verilerin düzenlenmesi ile elde edilemeyeceğini sorgular yolu ile verilerin diğer tablolardan alınması gereklidir.



Grafik arayüz ile basit sorgu oluşturma işlemlerini uygulamak

Sorgular, tablolarda depollanmış verileri özel bir görsel arayüz sunan nesnelerdir. Sorgu sonuçları genellikle formlara ve raporlara yönelik kayıt kaynağı olarak hizmet verir. Sorguların en sık kullanılan işlevi, tablolardan belirli verilerin bulunup, alınmasıdır. Sorgular, belirtlen koşullara uyan verileri bulmaya ve almayı yardımcı olduğu gibi birden çok kaydını aynı anda güncelleştirmek, silmek ya da özel hesaplamalar yapmak için de kullanılır. Access'te; Seçme sorgusu Eylem sorgusu, Çapraz Sorgu, Parametre Sorgusu, SQL sorgu olmak üzere beş farklı sorgu türü vardır. Eylem sorguları; Tablo oluşturma sorgusu, Ekleme sorgusu, Güncelleştirme sorgusu ve Silme sorgusundan oluşur. En yaygın kullanılan sorgu türü seçme sorgularıdır. Seçme sorgusu, veriyi bulup alır ve kullanıma hazır hale getirir. Sorgunun sonuçları ekranda görüntülenebilir, yazdırılabilir ya da kopyalanabilir. Seçme sorguları basit sorgulardır, Sorgu Sihirbazı ya da Sorgu Tasarımı kullanılarak oluşturulabilir. Birden fazla tablo kullanılarak seçme sorgusu oluşturulacaksa, tablolardan birbiri ile ilişkili olması gereklidir.

Kendimizi Sınayalım

1. Birbiri ile ilişkili veri dosyaları ve veri tabloları kümesinin çok amaçlı kullanıma olanak sağlayacak şekilde depolanmasına ne ad verilir?
 - a. Veri sayfası
 - b. Veritabanı
 - c. Veri tablosu
 - d. Veri alanı
 - e. Veri nesnesi
2. Aşağıdakilerden hangisi masaüstü veritabanı sistemlerinin özelliklerinden biridir?
 - a. Karmaşık sistemlerdir
 - b. Üst düzey kullanıcı bilgisi gerektirir
 - c. Elde edilmesi kolay sistemlerdir
 - d. Pahalı sistemlerdir
 - e. Kodlama bilgisi gerektirir
3. Veritabanı tasarım sürecinin **ilk** aşaması aşağıdakilerden hangisidir?
 - a. Alan veri türlerini belirlemek
 - b. Tablolar arasındaki ilişkileri belirlemek
 - c. Veritabanının amacını belirlemek
 - d. Kullanılacak Formları belirlemek
 - e. Birincil anahtarları belirlemek
4. Aşağıdakilerden hangisi masaüstü veritabanı sistemleri yazılımlarından biridir?
 - a. Oracle
 - b. MySQL
 - c. Informix
 - d. DB2
 - e. MS Access
5. İlişkisel veritabanlarında sistem, tüm verileri aşağıdaki nesnelerden hangisi ile yönetir?
 - a. Tablolar
 - b. Formlar
 - c. Sorgular
 - d. Raporlar
 - e. Modüller
6. Resim 4.5'de verilen Çalışanlar tablosuna kayıt eklemek için aşağıdaki eylemlerden hangisi gerçekleşmelidir?
 - a. Şirkete yeni personel alımı
 - b. Şirket çalışanlarından birinin ünvan değişikliği
 - c. Şirket çalışanlarından birinin sipariş alması
 - d. Şirket çalışanlarından birinin işten ayrılması
 - e. Şirket çalışanlarından birinin izinden dönmesi
7. Oluşturulan bir tablo alanında e-posta adresleri yer alırsa, alan veri türü aşağıdakilerden hangisidir?
 - a. Ek
 - b. Not
 - c. Evet/Hayır
 - d. OLE Nesnesi
 - e. Köprü
8. Veritabanında depolanan tablolara veri eklemek, düzenlemek ya da görüntülemek için aşağıdaki veritabanı nesnele-rinden hangisi kullanılır?
 - a. Tablolar
 - b. Formlar
 - c. Sorgular
 - d. Makrolar
 - e. Modüller
9. Aşağıdaki veritabanı nesnelerinin hangisinde veriler, yazdırılabilen bir görünümde düzenlenir?
 - a. Tablolar
 - b. Formlar
 - c. Sorgular
 - d. Raporlar
 - e. Modüller
10. Birden fazla tablo kullanarak seçme sorgusu oluşturulmuşsa aşağıdaki ifadelerden hangisi doğrudur?
 - a. Sorgu oluşturulurken mutlaka kod kullanılmıştır
 - b. Tabloların alan veri türleri aynıdır
 - c. Tablolar birbiri ile ilişkilidir
 - d. Tabloların alan sayıları birbirine eşittir
 - e. En az bir alanda ölçüt kullanılmıştır

Kendimizi Sınayalım Yanıt Anahtarları

- | | |
|-------|--|
| 1. b | Yanıtınız yanlış ise “Masaüstü Veritabanları” konusunu yeniden gözden geçiriniz. |
| 2. c | Yanıtınız yanlış ise “Masaüstü Veritabanları” konusunu yeniden gözden geçiriniz. |
| 3. c | Yanıtınız yanlış ise “Masaüstü Veritabanları” konusunu yeniden gözden geçiriniz. |
| 4. e | Yanıtınız yanlış ise “Masaüstü Veritabanları” konusunu yeniden gözden geçiriniz. |
| 5. a | Yanıtınız yanlış ise “Tablolar, Raporlar, Formlar” konusunu yeniden gözden geçiriniz. |
| 6. a | Yanıtınız yanlış ise “Tablolar, Raporlar, Formlar” konusunu yeniden gözden geçiriniz. |
| 7. e | Yanıtınız yanlış ise “Tablolar, Raporlar, Formlar” konusunu yeniden gözden geçiriniz. |
| 8. b | Yanıtınız yanlış ise “Tablolar, Raporlar, Formlar” konusunu yeniden gözden geçiriniz. |
| 9. d | Yanıtınız yanlış ise “Tablolar, Raporlar, Formlar” konusunu yeniden gözden geçiriniz. |
| 10. c | Yanıtınız yanlış ise “Grafik Arayüz ile Veri Sorgulama” konusunu yeniden gözden geçiriniz. |

Sıra Sizde Yanıt Anahtarları

Sıra Sizde 1

MS Access, kişisel ihtiyaçlar için kullanılabileceği gibi küçük ölçekli işletmelerde de her türlü ihtiyaca cevap verecek niyetindedir. Örneğin küçük bir işletmede ürünler, müşteriler, müşteri siparişleri, ürün sevkiyatı gibi bir dizi bilgiyi Access veritabanında kayıt altına almak, veri sorgulamak ve raporlamak kısacası sistemi yönetmek oldukça kolaydır. Access nesnelerinin çoğu şablonlar ve sihirbazlar yardımıyla oluşturulabileceği için kullanıcıların bilgi düzeyinin yüksek olması gerekmek. Masaüstü bilgisayarlarında yazılım çalışabilir ve lisanslı yazılım elde edilmesi oldukça ucuzdur.

Sıra Sizde 2

Tablonun Tasarım görünümünde açılması, tablonun yapısına ilişkin ayrıntılı bir görüntü elde edilmesini sağlar. Örneğin, her alanın alan adı ve alan veri türü ayarı tasarım görünümünde görüntülenir. Deneyimli kullanıcılar tabloları doğrudan tasarım görünümünde tasarlarken, veri sayfası görünümünde ya da formlar yardımıyla veri girişi yaparlar.

Sıra Sizde 3

Birinci tabloda birincil anahtar olmayan ve benzersiz dizine sahip olmayan bir alan, ikinci tabloda birincil anahtar olmayan ve benzersiz dizine sahip olmayan bir alana süreklenirse iki tablo arasında belirsiz bir ilişki oluşur. Belirsiz ilişkilere sahip tabloları içeren sorgularda tablolara arasında varsayılan birleştirme satırı görüntülenir ancak bilgi tutarlılığına zorlanmaz. Bu nedenle söz konusu tablolara kayıtlarının benzersizliği konusunda bir garanti yoktur.

Sıra Sizde 4

Resim 4.12'de görüntülenen Sipariş ayrıntıları sorgusunda; Ürün, Miktar ve Fiyat olmak üzere üç tane alan vardır.

Yararlanılan ve Başvurulabilecek Kaynaklar

Taşçı, C.N. (2004). **Veri Tabanları**. Cengiz Hakan Aydin, Yaşar HOŞCAN ve Ali Ekrem ÖZKUL (Edt.), Temel Bilgi Teknolojileri. Eskişehir: Anadolu Üniversitesi.

Cox, J., Lambert, J. and Frye, C. (2011). **Adım Adım Microsoft Office Professional 2010**, Ankara.

Yararlanılan ve Başvurulabilecek Internet Adresleri

Microsoft Office Access 2016, <https://products.office.com/tr-tr/access> (Erişim Tarihi: 1 Kasım 2015)

Wikipedia (Microsoft Access), https://tr.wikipedia.org/wiki/Microsoft_Access (Erişim Tarihi: 2 Kasım 2015)

Microsoft Çevrimiçi Office Desteği (Access 2016), <https://support.office.com/tr-TR/article/Access-veritaban%C4%B1n%C4%B1-yap%C4%B1s%C4%B1n%C4%B1-%C4%B1-%C3%B6%C4%9Frenme-001A5C05-3FEA-48F1-90A0-CCCAA57BA4AF> (Erişim Tarihi: 6 Kasım 2015)

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 adlandırılmasında olup, 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** komutlardır. 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
CREATE SCHEMA	Bu komut ile mevcut veritabanında bir şema içinde tablolar ve görünümler oluşturulabilir. Bu şema üzerinde farklı kullanıcılara ait yetkiler tanımlanabilir.
CREATE CERTIFICATE	SQL sunucudaki bir veritabanına sertifika ekler.
CREATE SEQUENCE	Dizi üretimi için bir nesne ve özelliklerini oluşturur. Tanımlanan değerlere bağlı olarak farklı değerlerde nümerik diziler üretir.
CREATE ASYMMETRIC KEY	Veritabanında asimetrik anahtar oluşturur.
CREATE INDEX	Tablo veya görünüm üzerinde ilişkisel indeks oluşturur. Özellikle sorgu performansının artırılması için farklı yapıarda oluşturulabilir.
CREATE TABLE	Veritabanında yeni bir tablo oluşturur.
CREATE TRIGGER	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.
CREATE VIEW	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 kullanılanıdır. 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:

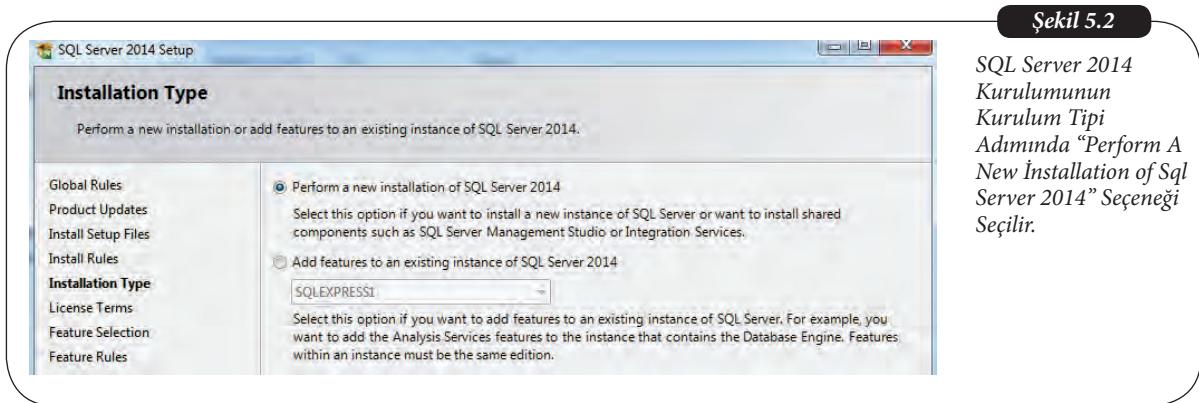
- Kurulum (Installation):** Bu adımda, Şekil 5.1'deki “New SQL Server stand-alone installation or add features to an existing installation” seçeneğinin seçilmesi gereklidir.

Şekil 5.1

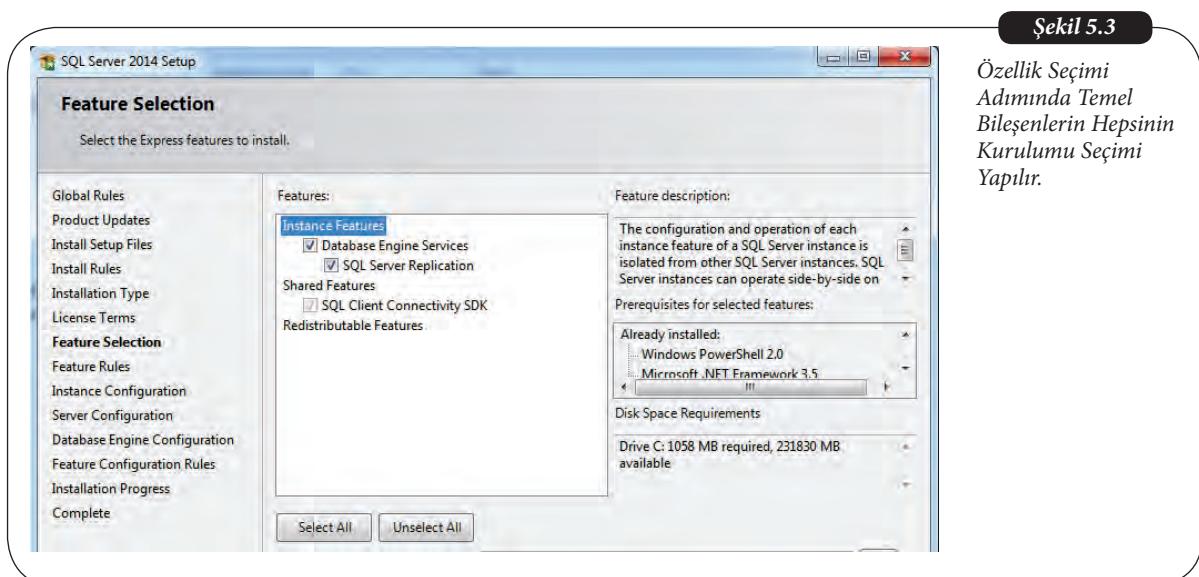
MS SQL Server 2014 Kurulumunda “New SQL Server Stand-Alone Installation or Add Features to An Existing Installation” Seçeneginin Seçilmesi Gerekir.



- Evrensel Kurallar (Global Rules):** Bu adımda, kurulum yardımcı 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.



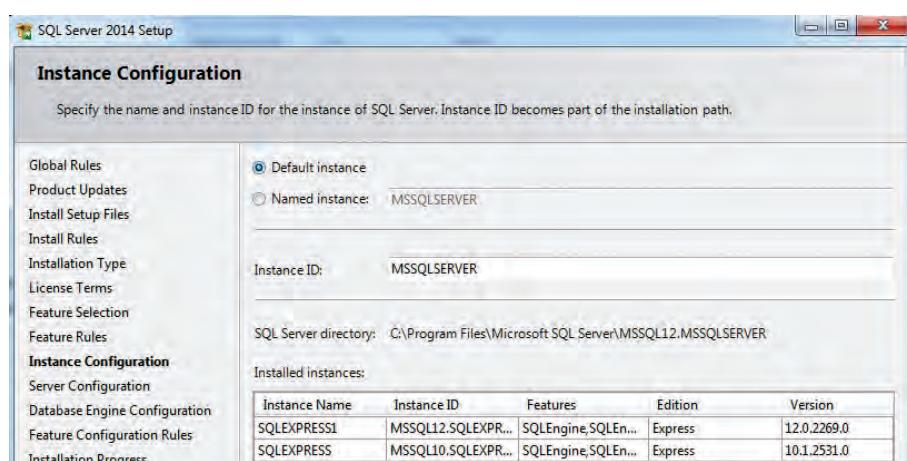
- iv) **Lisanslama (License Terms):** Bu adımda “I accept the license terms” ile lisans anlaşmasının kabul edildiğini işaretlenmelidir.
- v) **Özellik Seçimi (Feature Selection):** Express kurulum adımında, Şekil 5.3'teki temel bileşenlerin hepsinin kurulumu önerilir. Bunlar, MS SQL Server üzerinde hangi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

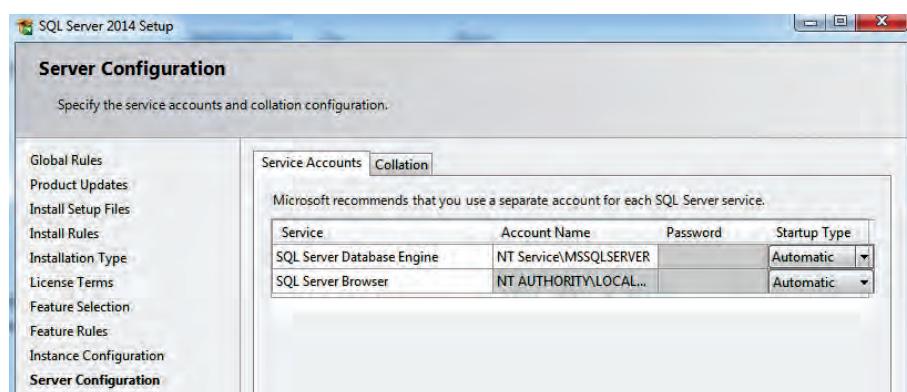
SQL Server 2014 kurulumunun Oluşum Yapılandırması Adımında “Default Instance” Seçimi Yapılır.



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

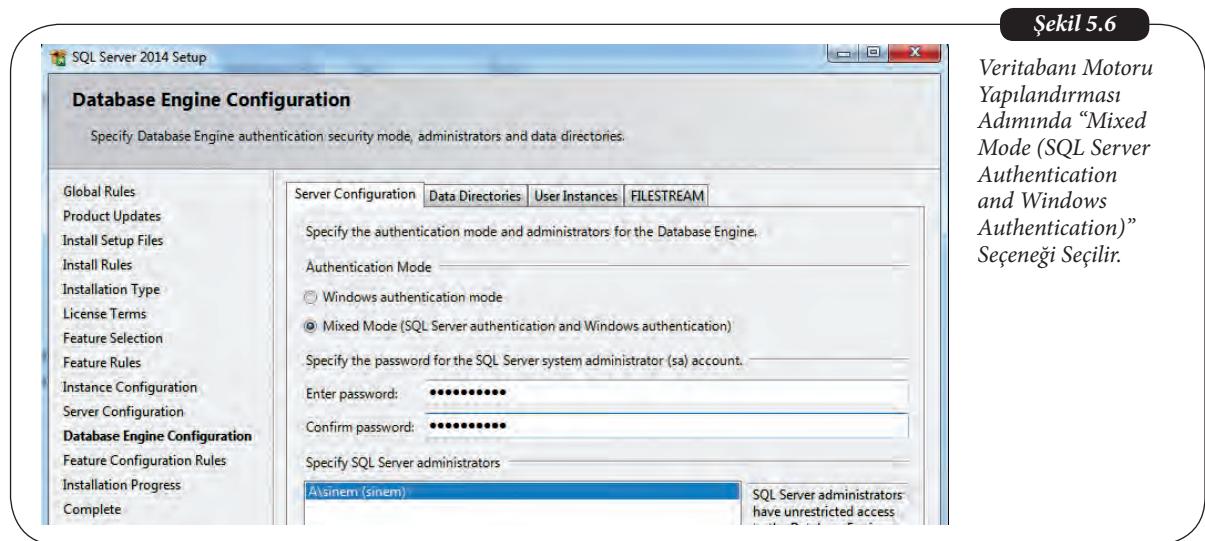
Şekil 5.5

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.

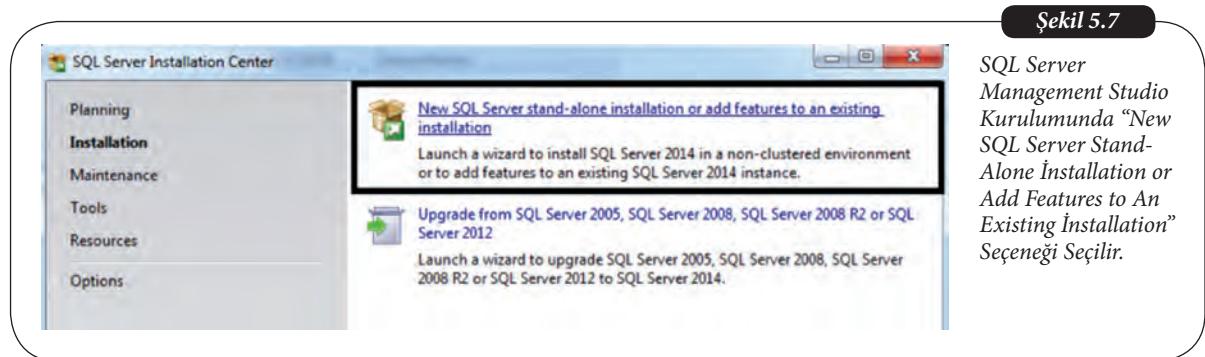


- ix) **Kurulum İşlemi ve Tamamlanma (Installation Progress and Complete):** Bu adımda kurulum işlemi sürdürülür ve tamamlanır. Kurulum yapılan özellikler de bu adımda listelenir.

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.

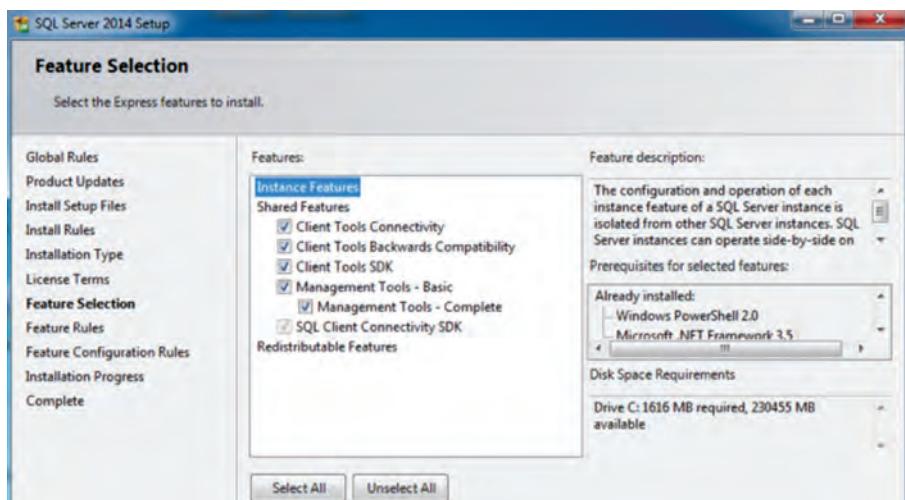
- i) **Kurulum (Installation):** Bu adımda Şekil 5.7'deki “New SQL Server stand-alone installation or add features to an existing installation” seçeneği seçilir.



- 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 bilgisayarlarınızdaki "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 VTY'Sde 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

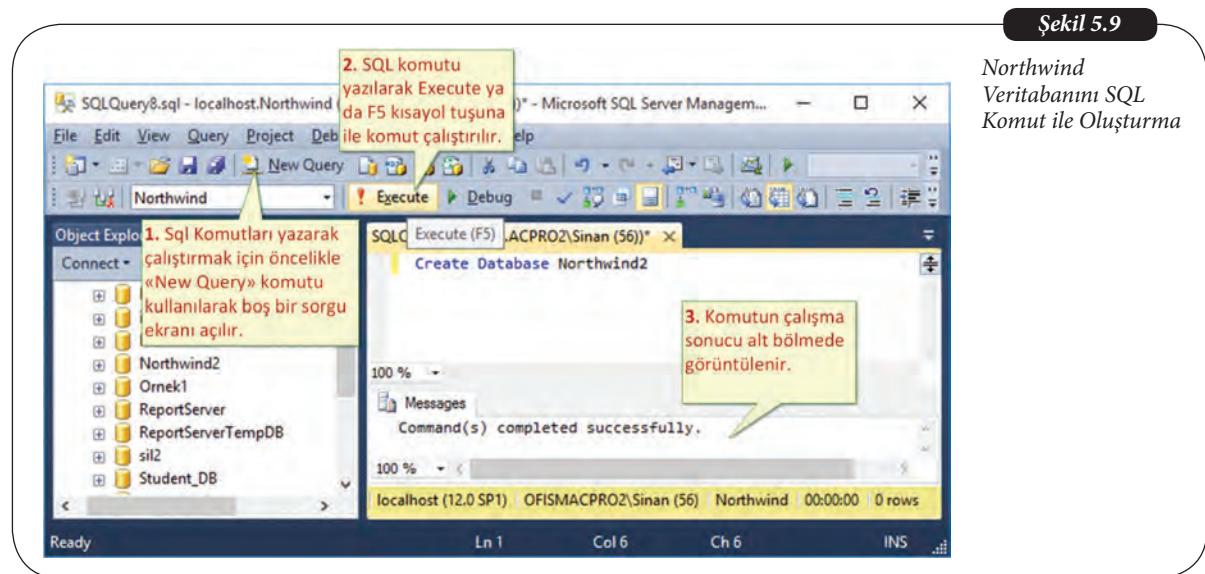
"Northwind" veritabanını komut ile oluşturmak için; "SQL Server Management Studio" programı açılarak ana menüde "New Query" e tıklayarak **SQL Sorgu penceresi** oluşturulur. Şekil 5.9'deki SQL Sorgu penceresine

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



Şekil 5.9

Northwind
Veritabanını SQL
Komut ile Oluşturma

"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

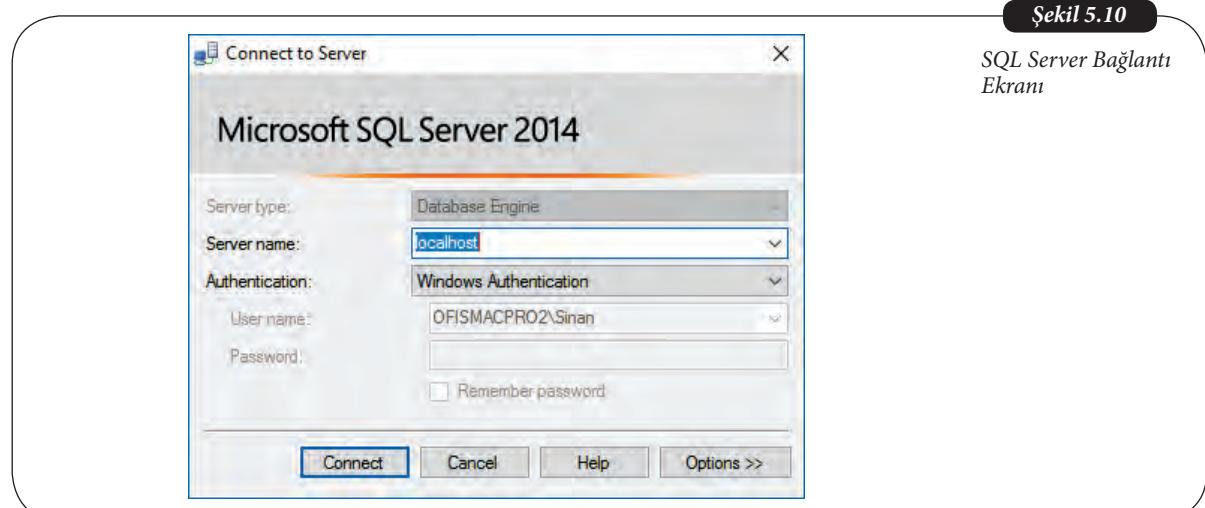
"NWYeniİsim" veritabanını silmek için; "SQL Server Management Studio" programı açılarak ana menüde "New Query" e tıklayarak SQL Sorgu penceresi oluşturulur. İlgili SQL Sorgu penceresine

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.



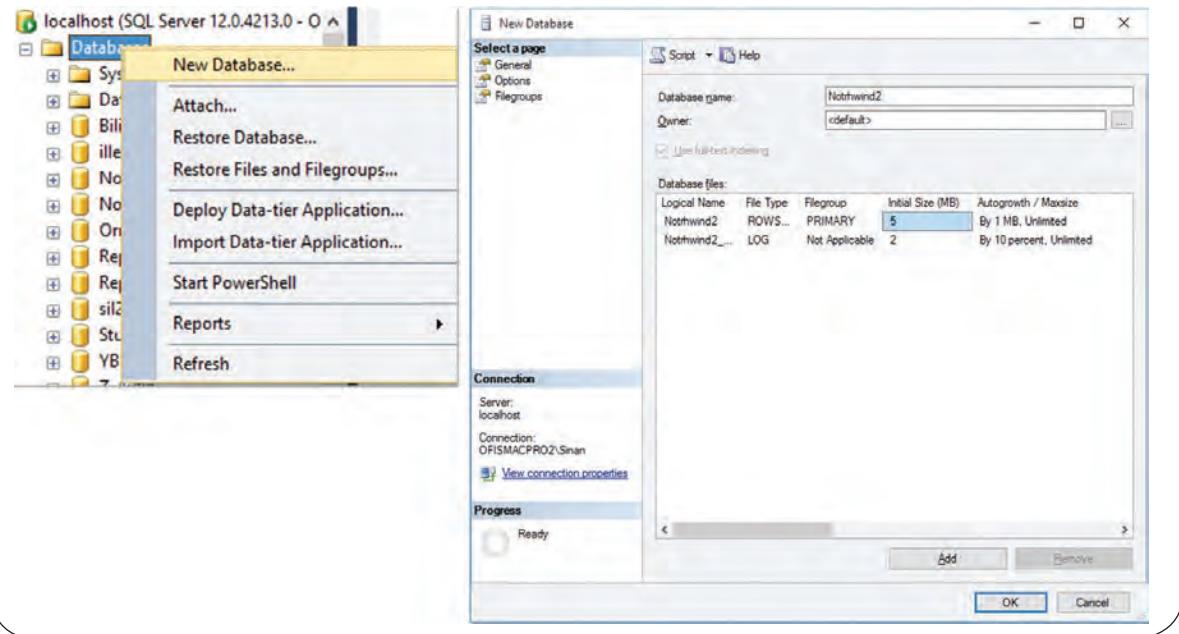
Şekil 5.10

SQL Server Bağlantı
Ekranı

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

Yeni Veritabanı Oluşturma



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. üniteye 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] özniteligiine 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 sırasındaki bu kısıtlara uyumluluğu kontrol eder. Burada, veritabanı tasarımcısı alternatif davranış seçenekleri olan **SET NULL**, **CASCADE** ve **SET DEFAULT**'u da tasarlatabilir. Bu davranışın tetiklenmesi için veritabanında yapılacak olan olası silme ve güncelleme içi **ON DELETE** veya **ON UPDATE** durumlarına bağlı tanımlanabilir.

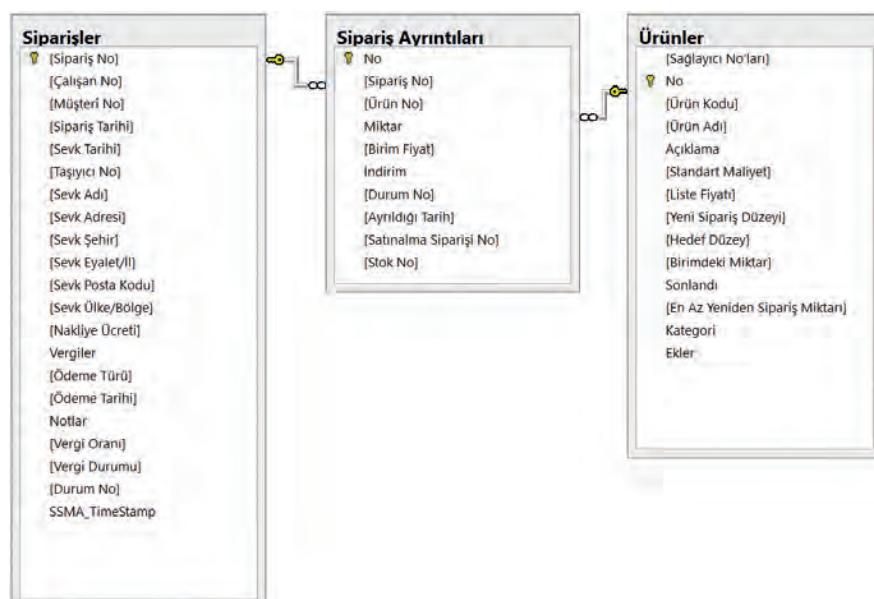
Birincil Anahtar (Primary Key) olarak tanımlanan alan ya da alanlar, ilgili tabloda benzersiz değer alırlar. Diğer bir deyişle aynı değerin faklı satırda yer almamasını garanti altına alırlar.

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

2



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

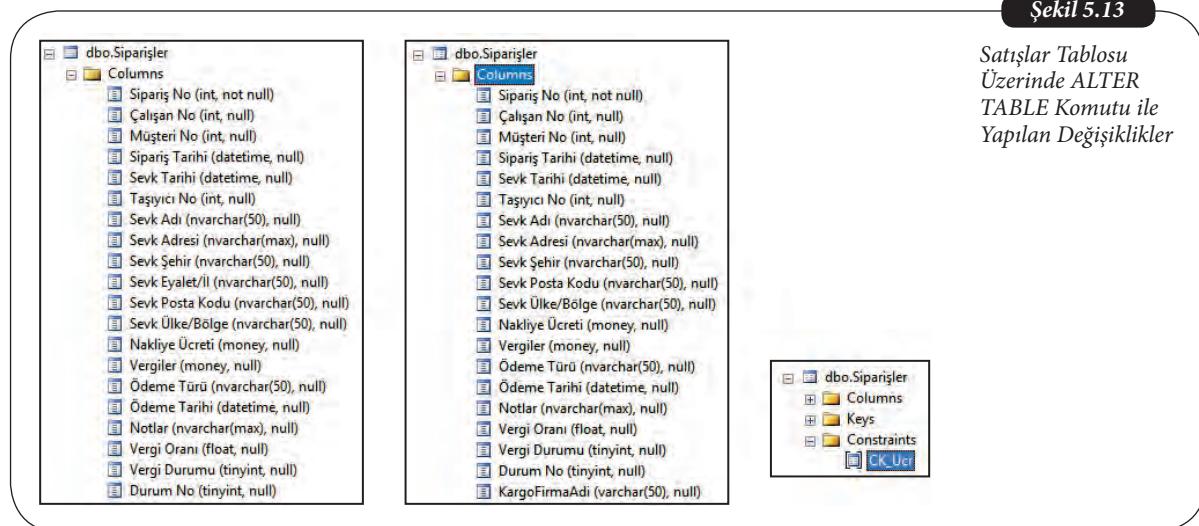
Örnek 5.1 ile oluşturulan *Siparişler* tablosunda aşağıdaki işlemleri gerçekleştiren SQL Komutlarını oluşturunuz.

- [*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.



SIRA SİZDE

3

VERİ TANIMLAMA DİLİ İNDEKS İŞLEMLERİ

İndeksler veritabanı yönetim sistemlerinde mevcut verilerin dizin hâline getirilerek istenilen tablo alanlarına daha hızlı ulaşılmasını 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.

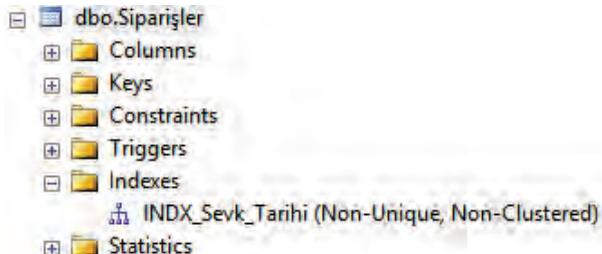
İstenilen indeksin oluşturulması için Create Index komutu kullanılmalıdır. Create Unique Index komutu ise ilgili alanda değerleri tekrarlanmadığı durumlarda kullanılır ve dolayısıyla aranan değerleri bulmak için daha az süre gerektirir. Siperişlere ilişkin aynı tarihte sevk olabileceği için [Sevk Tarihi] 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

Create Index Komutu ile Oluşturulan Indeks in Management Studio'da İzlenmesi



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.

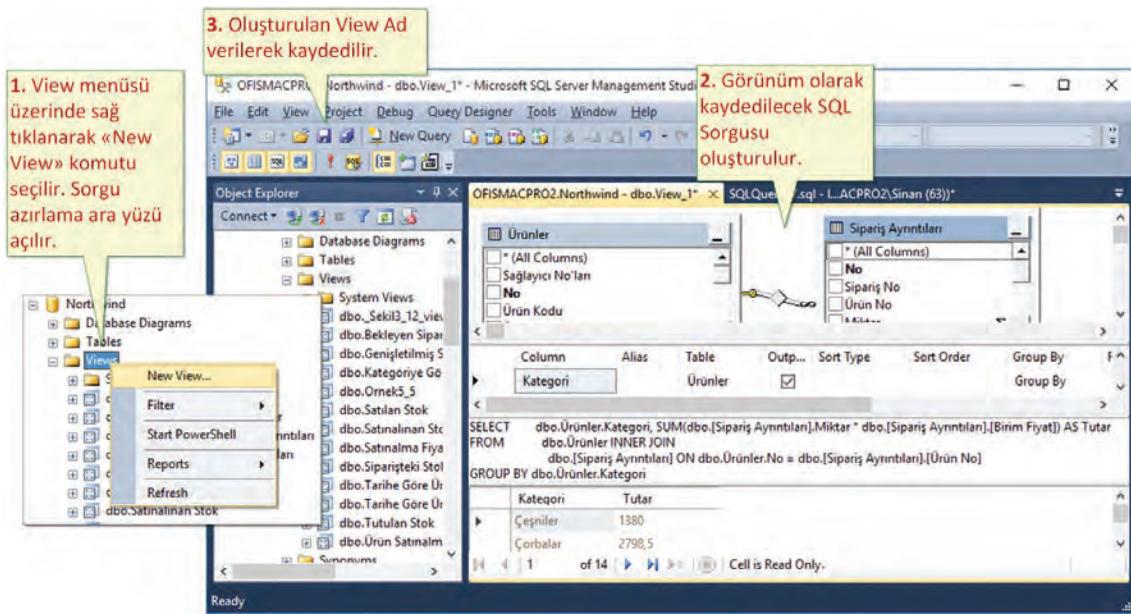


DİKKAT

Görünümler SQL komutu yazarak oluşturulabildiği gibi Management Studio Grafik arayüzünde 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
  
```

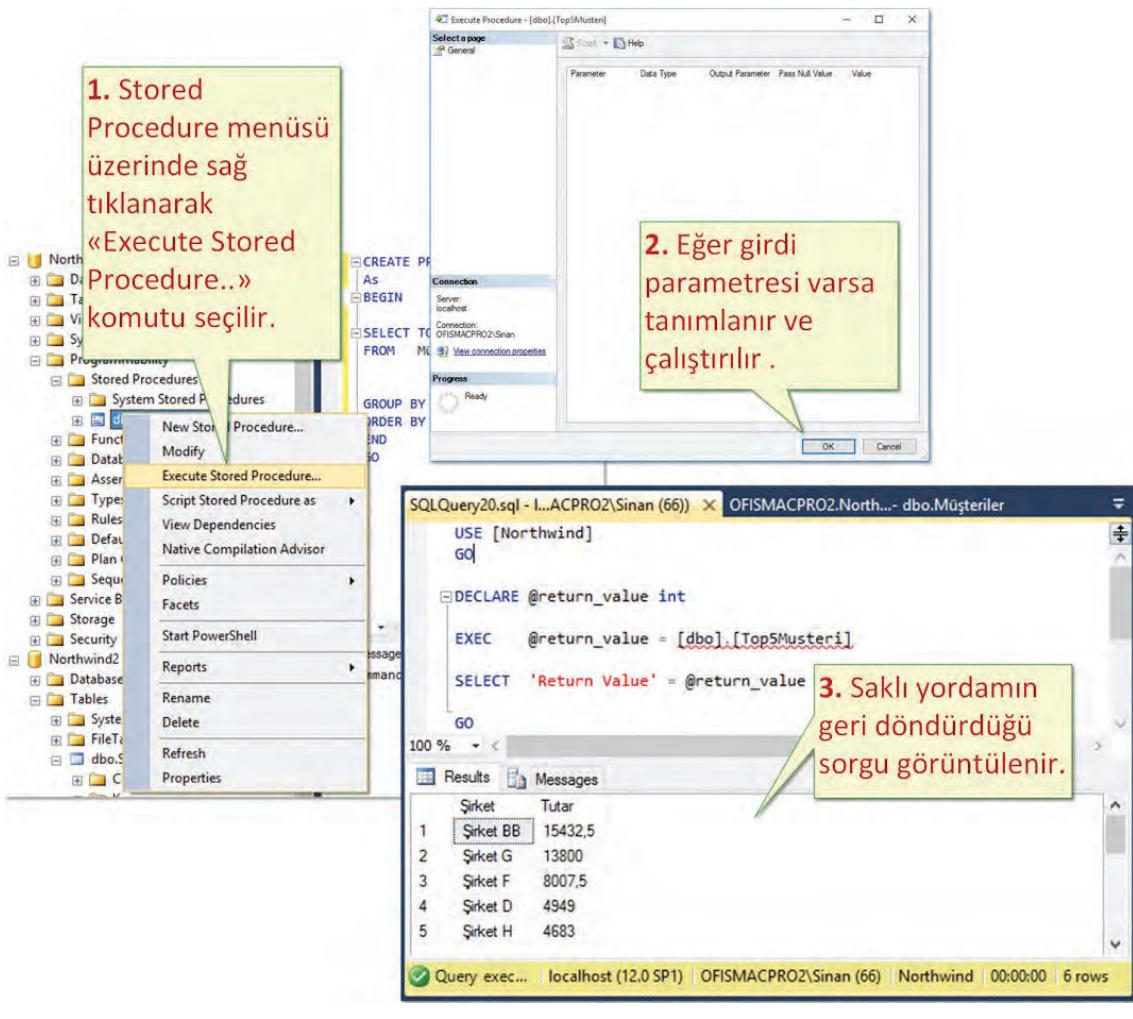
Northwind örnek veritabanında tutar olarak en fazla satış yapılan 5 müşteriyi listeleyen saklı yordamı oluşturun.

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.

Sekil 5.16*Saklı Yordamın Çalıştırılması*

Oluşturulan herhangi bir saklı yordamı silmek için **DROP PROCEDURE** SaklıYordamIsmini 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 kümlesi ya da komutların yazılmasıdır. Görüntüler sadece bir SQL sorgusunu çalıştırırken saklı yordamlarda Transactional SQL programlama dili kullanılabilir. Bu programlama dilinde değişkenler oluşturulabilir, döngüler kurulabilir, kayıt kayıt veri tabloları işlenebilir 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 oluşturabilme 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 sınırlamları alanların tanımlanmasında ve veri tutarlığının sağlanması için kullanılabilir. 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

Görünümleri(view), kullanıcıların veritabanındaki nesneleri sorgulayarak istedikleri veri kümelerini elde etmek için kullanılan yapılardır. Bu yapılar veriyi değil veriyi elde edecek sorgu komutlarını saklayarak çağrılarındakilerin 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 yanlışdır?
 - 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 alfabeteki 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 kümeleme indekse sahip olabilir. Kümeleme 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

6

Amaçlarımız

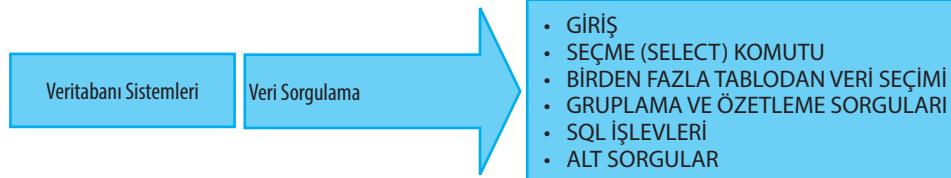
Bu üniteyi tamamladıktan sonra;

- 🕒 Seçme sorgularının yazım kuralını tanımlayabilecek,
- 🕒 Tabloların birbirine bağlanma türlerini sıralayabilecek,
- 🕒 Gruplama ve özetleme işlemlerini tanımlayabilecek,
- 🕒 SQL işlevlerini sıralayabilecek,
- 🕒 Alt Sorgu yapılarını tanımlayabilecek bilgi ve becerilere sahip olacaksınız.

Anahtar Kavramlar

- Seçme Sorguları
- Gruplama ve Öztleme
- Tablo Bağlama
- Tablo Birleştirme
- Select, From, Where, Group By, Having
- Alt Sorgu

İçindekiler



Veri Sorgulama

GİRİŞ

SQL dili veritabanı sistemlerinin yönetilmesi için tasarlanmış bir **bildirim** dilidir. Bu dil kullanıcıların ya da istemci yazılımlarının veri ile ilgili isteklerini alarak, depoladığı veriler üzerinde uygulayan bir yapıdadır. Diğer programlama dillerinden farklı olarak işlemleri değil sonuçları tarif eden bir yapısı vardır. Dolayısıyla veritabanı yönetim sistemi tarafından SQL komutları çözümlenerek istenilen işlem yerine getirilir. Bu yapı verinin eklenmesi, işlenmesi ve sorgulanması gibi birçok işlemin istemci ya da kullanıcılar tarafından kolayca yapılmasını sağlamaktadır.

Veritabanı yönetim sistemlerini kullanmamızın temel nedenlerinden biri de verinin farklı görünümelerini elde edebilmektir. Organizasyonlarda tüm sistem verilerini barındıran merkezi veritabanları organizasyonda farklı kademe ve görevlerdeki çalışanların veri ihtiyacını karşılamaktadır. Örneğin bir bankada banko çalışanı mudisinin hesap bilgilerine ulaşmakta ve işlem yapmak için bir hesap kaydına ulaşırken bankanın müdüri aylık mevduat ortalaması bilgisine erişmek isteyebilir. Bu ve benzer veri ihtiyaçlarını karşılamak için veritabanı yönetim sistemleri yazılımı ihtiyaça göre veri kümesinin elde etmek üzere tasarlanmıştır. İlişkisel veritabanı yönetim sistemlerinde tablolar hâlinde depolanan verilere ulaşmak için SQL dilinin Veri Sorgulama Dili (DQL- Data Query Language) komutları kullanılmaktadır. Bu ünitede SQL'de veri sorgulama için kullanılan **SELECT** komutu ile ilgilidir. Verinin sadece okunması ile ilgili olan bu komutun çalışma prensibinin öğrenilmesi ilişkisel veritabanı sistemlerinin anlaşılmasıında en önemli adımdır.

SELECT komutunun temel yazım şekli verildikten sonra verinin sınırlanması, farklı tablolardan verilerin birleştirilmesi, veri kümelerinin birbiri arası birleştirilmesi, iç-içe sorgular ünite içerisinde aktarılmıştır. SQL sorgu dili komutlarının öğrenilmesinde kullanıcıların bu komutları deneyimlemeleri dilin öğrenilmesi için oldukça önemlidir. Okurların önceki ünitelerde SQL Server Express yazılımını kendi bilgisayarlarına kurarak örnek veritabanı yüklemeleri ve ünite içerisindeki örnek ve sıra sizde sorularını deneyimlemesi konunun öğrenilmesinde fayda sağlayacaktır.

SEÇME (SELECT) KOMUTU

SELECT ifadesi ilişkisel veritabanı yönetim sistemlerinde veri kayıtlarını getirmek için kullanılan komuttur. Bu komut ile bir veri tablosunun tüm satırlarının getirilebileceği gibi bir veya daha çok sayıda tabloda bulunan sütün veya sütunlarının istenilen şartlara uygun kayıtlarınında getirmek mümkündür. **SELECT**, kullanıcıların istediği veri kümelerini tarif edebilmeleri için oldukça esnek bir yapıda tasarlanmıştır. Genel yazım kuralı aşağıda verilen komutun köşeli parantez içerisinde belirtilen bölümleri isteğe bağlı seçeneklerdir.

SQL, Yapılandırılmış Sorgu Dili veri ile ilgili yapılacak işlemleri tarif etmek için İngilizce dil yapısında geliştirilmiş bir **bildirim** dilidir (Declarative language).

SELECT komutu VTYS'nin en fazla kullanılan komutudur. Farklı tablolarda depolanan verinin kullanıcıların ihtiyacına cevap verecek şekilde elde edilmesini sağlar.

```
SELECT alan_listesi [INTO yeni_tablo]
[FROM tablo_kaynağı]
[WHERE seçme_koşulları]
[GROUP BY gruplama ifadeleri]
[HAVING Gruplanan_alan_kısıtlamaları]
[ORDER BY sıralama düzeni ASC DESC]
```

"SELECT * FROM Tablo_Adi" seçme sorgusunu Tablo_adi tablosundaki tüm alan ve satırları listeler.

Seçme komutu İngilizce dil yapısına göre bir emir cümlesidir ve seç fiili ile başlamaktadır. Yukarıdaki Seçme sorgusunun yazım kuralında “Select alan_listesi” dışında yer alan kısımlar zorunlu değildir. Çalışabilecek en basit seçme sorgusuna örnek olarak **“SELECT 'Hello World”** verilebilir. Komutun diğer kısımlarını satırlar hâlinde kısaca aşağıdaki gibi açıklanabilir.

FROM ile başlayan satır seçme işleminin hangi veri kümelerinden yapılacağını belirtir. Bu kısımda bir tablo, bir görünüm ya da bir alt sorgu yer alabilir.

WHERE satırı seçme işleminde görüntülenecek ya da hesaba katılacak satırların sınırlanması sağlanır. Filtreleme işlemi gibi düşünülebilir.

GROUP BY ifadesi verileri özetleme ya da gruplama işlemleri için kullanılabilen bir seçme sorgusu bütümüdür.

HAVING Gruplanan ya da hesaplanan alanların sınırlanması için kullanılan bir kısımdır. Where kısmı ile karıştırılmaması gereklidir.

ORDER BY sorgunun en sonunda yer alan sıralama işlemidir. Eylem gereği de tüm seçme işlemi tamamlandıktan sonra verinin kullanıcıya hangi sırada gönderileceği bu satırda yazılan komutlarla belirlenir.

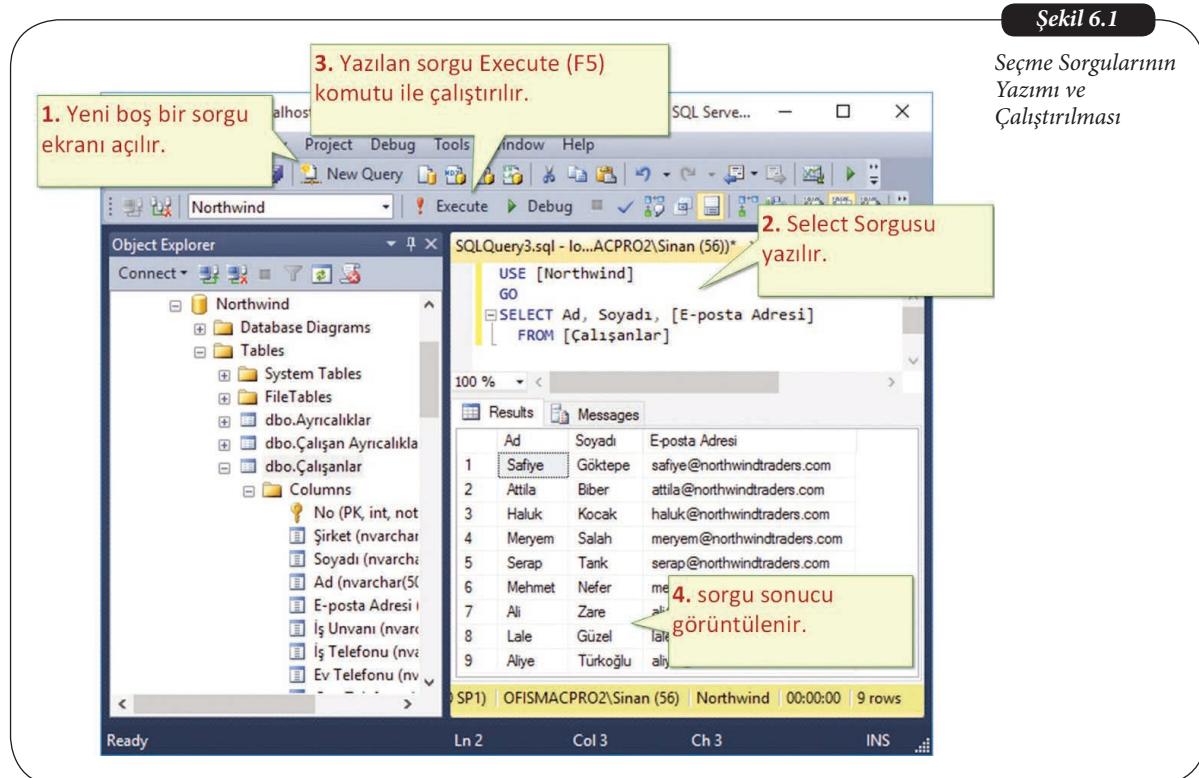
Select komutu ile hedeflenen veri kümelerini seçmek için komutun yapısında olduğu gibi yukarıdan aşağıya bir tasarım yapılması oldukça önemlidir. Aksi takdirde istenilmeyen sonuçlara ulaşılabilirmektedir. Seçme sorgusunun evreni **FROM** sonrasında belirtilen veri kaynakları tarafından oluşturulur. Daha sonraki **WHERE**, **GROUP BY**, **HAVING** ve **ORDER BY** ifadeleri sırasıyla takip eder. Önce **WHERE** ile seçme ve gruplama işlemlerinde hesaba katılmak istenmeyen satırlar çıkarılır sonrasında istenilen kümeye gruplama ve özetleme işlemi gerçekleştirilir. Gruplama sonrasında yine görüntülenmek istenmeyen hesaplanmış satırlar **HAVING** komutu ile devre dışı bırakılır. Sürecin en son işlemi ise sıralama olmaktadır. Hedeflenen veri kümelerini elde edilecek sorguların oluşturulmasında bu sıralamanın unutulmaması tasarımanın doğru yapılması için önemlidir. **SELECT** ifadesinin mantıksal işleme sırasında aşağıdaki gibidir (<https://msdn.microsoft.com/en-us/library/ms189499.aspx>). Bu sıralamada yer alan aşamalar 6. adım dışında izleyen bölümlerde yer alacaktır.

Seçme ifadesinin mantıksal işleme sırasının bilinmesi kullanıcıların sorguları daha iyi tasarlamasında yardımcı olmaktadır.

1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH CUBE or WITH ROLLUP
7. HAVING
8. SELECT
9. DISTINCT
10. ORDER BY
11. TOP

Bu kitap için kullanılan örnek Northwind veritabanı için örnek bir sorgu ve çalıştırılma sonucu Şekil 6.1'de verilmiştir. Şekil 6.1'de kullanılan SQL Server Management Studio kullanıcıların veritabanı yönetim sistemini yönetmeleri ve tasarlamaları için oluşturulmuş grafik bir ara yüzdür. Arka planda sunucularda çalışan veritabanı motoru bu arayüze ve bir çok arka plandaki veri istek talebine yanıt vermektedir. Örneğin bir web sunucusunun

isteği yazılıma farklı ara yazılımlar sayesinde ulaşılır. Burada SQL dilinin kullanımının öğrenilmesi için bu ara yüzde örnekler gösterilecektir. Şekilde veritabanı yönetim sisteme “**SELECT Ad, Soyadı, [E-posta Adresi] FROM [Çalışanlar]**” sorgusunun sonucu istenmiş ve çalışanlar tablosundaki [Ad], [Soyadı], [E-posta Adresi] alanlarının listelenmesi sağlanmıştır.



Select sorgusuyla elde edilen bir veri kümesinin yeni bir tablo olarak kaydedilmesi için **FROM** ifadesinden önce **INTO tablo_Adı** ifadesi yazılmalıdır. Aşağıdaki sorgunun sonucunda kullanıcıya bir tablo listelenmez, bunun yerine “**(9 row(s) affected)**” iletisi ile yeni bir veritabanı tablosu oluşturulduğu bildirilir.

**SELECT Ad, Soyadı, [E-posta Adresi] INTO Çalışanlar_ePosta
FROM [Çalışanlar]**

Bu üitede kullanılacak Northwind veritabanı dosyalarına ulaşmak için <https://bit.ly/30UFiLpK839el> adresinde yer alan Northwind.rar dosyasına ulaşabilir ve SQL Server veritabanına ekleyebilirsiniz. Ekleme işlemi bu kitabın 3. Ünitesinde anlatılmıştır. Eğer daha önce kurulmuş ise tekrar kurulması gerekmeyecektir.



Benzersiz Değerlerin Elde Edilmesi (DISTINCT)

Seçme sorgularında sıkça kullanılan ifadelerden birisi de bir alandaki verilerin tekrarsız olarak görüntülenmesini sağlayan **DISTINCT** ifadesidir. 9 satır verinin yer aldığı [Çalışanlar] tablosu için yazılan “**SELECT DISTINCT [İş Unvanı] FROM [Çalışanlar]**” sorgusu [İş Unvanı] alanındaki tüm verileri tekrarsız olarak “Genel Müdür Yardımcısı, Satış Koordinatörü, Satış Müdürü, Satış Temsilcisi” şeklinde dört satır ile görüntüler. **DISTINCT** ifadesi kayıtların sayılması ile ilgili işlevlerde de benzersiz satır sayısının sayılması ile ilgili kullanılmaktadır.

Sorgu Sonuçlarının Sıralanması (ORDER BY)

Seçme komutunun mantıksal işleme sırasının son iki adımı olan **ORDER BY** ve **TOP** ifadeleri elde edilen veri kümesinin kullanıcı ya da istemci yazılıma iletilmeden önce sıralanmasını sağlar. **SELECT** komutunun son kısmında yer alan **ORDER BY** ifadesinden sonra veri kümesinin istenilen alanlara göre sıralanması sağlanır. Aşağıda [*Çalışanlar*] tablosunun [*Şehir*], [*Soyadı*] ve [*Ad*] alanlarını sorgulayarak elde edilen veri kümesinin önce [*Şehir*]’e göre artan, [*Ad*]’a göre azalan ve [*Soyadı*] alanına göre artan sıralayan bir seçme sorgusu yer almaktadır.

```
SELECT Şehir, Soyadı, Ad FROM Çalışanlar
ORDER BY Şehir, Ad DESC, Soyadı
```

Yukarıdaki örnek sorguya dikkat edilirse artan sıralama yapılan alan adlarının yanında herhangi bir ifade yok iken azalan sıralanmak istenen [*Ad*] alanının yanında **DESC** (DESCending: azalan) kelimesi yer almaktadır. Sorgu ifadelerinde sıralama yapılacak alanların sonuç kümesinde bulunma zorunluluğu yoktur.

Bazı durumlarda da **SELECT** komutu ile elde edilen verinin en üst satırlarının görüntülenmesi gerekebilir. Örneğin en fazla satış yapılan on müşteri ya da en fazla ciro yapılan ürün listesi elde edilmek istenebilir. VTYS tüm veri hazırlama sürecini yerine getirdikten sonra en son **TOP** kısmını işleyerek en üstte yer alan satırlar görüntülenir ya da iletılır. **TOP** ifadesinin kullanımında doğal olarak verinin sıralanması da söz konusudur. Aksi durumda en üstte yer alacak verilerin çok bir anlamı olmayacağından, Northwind veritabanında liste fiyatı en fazla olan üç ürünü bulmak için aşağıdaki sorgu yeterli olacaktır.

```
SELECT TOP (3) [Ürün Adı], [Liste Fiyatı]
FROM Ürünler ORDER BY [Liste Fiyatı] DESC
```

Sorguda **TOP** ifadesinden sonra parantez içinde yazılan ‘3’ ifadesi anlaşılacağı üzere en üstte yer alan üç satırın görüntülenmesini sağlar. Diğer taraftan sorgunun son kısmında yer alan sıralama komutunun yapısına dikkat edilmelidir.

Seçimlerin Sınırlanması (WHERE)

Veri sorgulamasında kayıtların belirli ölçütlerle göre sınırlanması ya da diğer bir ifadeyle tablolar içinde aranması için **WHERE** ifadesi kullanılır. Basit bir örnek ile başlanarak **WHERE** komutu ile gerçekleştirilebilecek koşul ifadelerine bu kısımda yer verecektir. Birçok istemci yazılımı, işlem yapılmak istenen nesnenin (müşteri, hesap, ürün, öğrenci vb.) girişi yapılarak veritabanında sorgular. Bu gibi işlemlerde Select sorgusunun **WHERE** komutu sonrası aranacak bilginin eşleştirilmesi yapılır. [*Ürünler*] tablosunda “Ceviz” adlı ürünün [*Liste Fiyatı*] ve [*Ürün Kodu*] bilgisi sorgulanmak için oluşturulmuş aşağıdaki sorguda **WHERE**’den sonra parantez içinde “[*Ürün Adı*] = ‘*Ceviz*’” şeklinde yazılan ifade ilgili ürünün bilgisinin veri tablosu içerisinde bulunarak görüntülenmesini sağlar. Bu ifade de parantez gerekli değildir ancak koşul ifadelerinin sayısının fazla olması parantez kullanımını oldukça önemli hâle getirmektedir.

```
SELECT [Ürün Kodu], [Ürün Adı], [Liste Fiyatı] FROM Ürünler
WHERE ([Ürün Adı] = 'Ceviz')
```

MS SQL Server VTYS’i yazılımında sıkça kullanılan koşul ifadeleri Tablo 6.1’de yer almaktadır. Bazı ifadelerin yanında köşeli parantez içerisindeki [*NOT*] ifadesi ilgili koşulun değilini ifade etmek için de kullanılabilir.

Operatör	Tanımı	Where sonrası yazım şekli
=	Eşitir	[Ürün Adı] = 'Ceviz'
<>	Eşit Değildir (!=)	[Kategori] <> 'Çerezler'
>	Büyük	[Liste Fiyatı]>50
<	Küçük	[Liste Fiyatı]<50
>=	Büyük veya eşit	[Ürün Adı] >= 'H'
<=	Küçük veya eşit	[Liste Fiyatı]<=10
[NOT] BETWEEN	Belirtilen değerler arasında. (Sınır değerleri listeye dahil edilir)	[Liste Fiyatı] BETWEEN 10 AND 20
[NOT] LIKE	Metin içerisindeki desene göre kısıtlama	[Ürün Adı] LIKE 'C%' (C ile başlayan ürünler)
[NOT] IN	Bir alanın birden fazla değer ile sınırlanması	[Ürün Adı] IN ('Ceviz', 'Hint Çayı')
EXISTS	Alt sorgular ile mevcut sorgunun sınırlanması sağlanır	EXISTS (Select * From Where)
IS [NOT] NULL	Boş satırların bulunması	[Liste FİYATI] IS NOT NULL
CONTAINS	Metin içerisinde karakter bazlı aranacak ifadeler	CONTAINS ('ŞikayetMetni', 'Son Kullanım')
FREE TEXT	Metin içerisinde doğal dil özelliklerine göre ifade arama	FREETEXT (Belge, 'go fast')

Tablo 6.1
SQL Koşul Operatörleri Listesi

Listede yer alan **CONTAINS** ve **FREETEXT** ifadeleri uzun metin verisi içeren alanlarda ilgili metin ifadelerinin bulunması için kullanılır. Arama yapılacak alanların daha önce Full-Text Indeks olarak tanımlanması gerekmektedir.

Ürünler Tablosunda [Liste fiyatı] 40 ile 60 arasında olan ve [Ürün Adı] içerisinde "A" harfi olan ürünleri bulmak istersek aşağıdaki koşul ifadelerinin yazılması gerekmektedir.

```
SELECT [Ürün Kodu], [Ürün Adı], [Liste Fiyatı] FROM Ürünler
WHERE ([Liste Fiyatı] BETWEEN 20 AND 40) AND ([Ürün Adı]
LIKE '%A%')
```

Sorgu cümlesi içinde **WHERE** ifadesinden sonra **AND** ile birbirine bağlanmış iki koşulun olduğu görülmektedir. İlk kısmı liste fiyatı koşulu ve diğer koşul ise ürün adı ile ilgili koşuldur. Koşullar kayıt bazlı değerlendirildiği için her bir kaydın bu koşulu sağlayıp sağlanmadığı kontrol edilecek ve her ikisini birlikte sağlayanlar kullanıcıya görüntülenecektir. İkinci koşuldaki **LIKE**'dan sonra gelen '**%A%**' ifadesi, metin içerisindeki % karakteri herhangi bir metin katarını temsil eder. Joker karakter ismi de verilen bu semboller ile metin içerisindeki desenler aranabilmektedir. **LIKE** ile kullanılabilen ifadeler aşağıda listelenmiştir

- %: Herhangi bir karakter dizisi ya da hiçbir metin için kullanılır. 'A%' ifadesi "A" ile başlayan ya da sadece 'A' içeren alanları sınırlayabilir.
- _ (Alt çizgi): Herhangi tek bir karakter. '_eri' ifadesi ilk harfi farklı olabilecek "Geri", "Seri" gibi ifadeleri bulabilmektedir.
- []: Köşeli parantez aralığında tanımlanan harfler ile sınırlanmaktadır. ' [a-f]' veya '[abcdef]' şeklinde yazılabilir. '[C-T]apa' ifadesi ile "Çapa", "Sapa" gibi metinler sınırlanabilir.
- [^]: Belirtilen aralıkta karakter içermeyen metinler bulunur. 'File[^1]%' ifadesi içerisinde 1 olamayan ancak 'File' ile başlayan metinleri bulabilecektir.

Koşul ifadeleri arasında **AND** ve **OR** ifadeleri kullanılabilmektedir. Mantıksal sınama ifadeleri olan bu ifadelerden **AND** birbirine bağlanan ifadelerin her ikisinin de gerçekleşmesini zorunlu kılar. **OR** ifadesi ise koşullardan en az birinin yerine getirilmesi durumunda ilgili satırı veri kümesine dahil eder.

SIRA SİZDE

1

Northwind veritabanında

```
SELECT [Ürün Kodu], [Ürün Adı], [Liste Fiyatı] FROM Ürünler
WHERE ([Ürün Adı] = 'Ceviz') AND ([Ürün Adı] = 'Zeytin Yağı')
sorgusunun nasıl bir çıktı vereceğini bulunuz.
```

BİRDEN FAZLA TABLODAN VERİ SEÇİMİ

İlişkisel veritabanı yönetim sistemlerinde gerçek dünya varlıklarının tablolar şekilde gösterimi söz konusudur. Bu yapılarda sistem, verileri birbirine bağlı çok sayıda tablolar hâlinde organize edilir. Daha önceki konularda da bahsedildiği üzere tablolar arasında farklı ilişki türleri kurulabilmektedir. Veritabanı tasarıımı esnasında tablolar ve aralarındaki ilişkilere karar verilmektedir. Ancak istenen veri kümelerini elde etme de tablo yapılarını ve aralarındaki ilişki özelliklerini bilmek gerekmektedir.

Basit olarak iki tabloyu birbirine bağlamak için **WHERE** komutu kullanılabilir. Örnek veritabanında hangi müşterinin hangi siparişleri verdığını listelemek için [*Müşteriler*] ve [*Siparişler*] tablosunu **WHERE** komutu ile bağlayarak bir sorgu yazılmak istenirse aşağıdaki sorgu komutu yazılabilir.

```
SELECT Top (3) Müşteriler.Şirket, Müşteriler.Soyadı,
Müşteriler.Ad, Müşteriler. [No], Siparişler.[Müşteri No]
FROM Siparişler, Müşteriler
WHERE Siparişler.[Müşteri No] = Müşteriler.[No]
```

İki tabloyu birbiri ile ilişkilendirmek ya da diğer bir ifadeyle bağlamak için **FROM** cümlesiinden sonra [*Siparişler*], [*Müşteriler*] tablolarını yazmak ve **WHERE** ile iki tablo arasındaki bağlantının belirtilmesi gereklidir. “*Siparişler.[Müşteri No]* = *Müşteriler.[No]*” eşitliği her iki tablodaki bu değerleri eşlemek anlamına gelir. İki tabloyu bağlamak [*Siparişler*] tablosunda yer alan [*Müşteri No*] değerini [*Müşteriler*] tablosundaki [*No*] alanında karşı gelen satırdaki değerleri birleştirmektedir. Tablo 6.2'de yukarıdaki sorgunun çalıştırılması sonucu elde edilen sonuç tablosu yer almaktadır. Bu tabloda [*Müşteri No*] ve [*No*] alanlarındaki değerlerin aynı olduğuna dikkat ediniz.

Tablo 6.2
Where ile
İlişkilendirilmiş İki
Tablo Sorgusunun
Sonucu

Şirket	Soyadı	Ad	No	Müşteri No
Şirket A	Berber	Aliye	1	1
Şirket A	Berber	Aliye	1	1
Şirket C	Aksak	Timur	3	3

JOIN Komutu ile Tabloların Bağlanması

Tabloları bir sorgu içerisinde birbiri ile bağlamanın en kolay ve yaygın yolu **JOIN** komutudur. **JOIN** komutu ile sadece iki değil çok sayıda tablo birbirine bağlanabilmektedir. Ayrıca farklı bağlanma türleri ile kullanıcıların istenilen veri kümelerini elde etmelerine olanak sağlar. Farklı **JOIN** komutları olmasına karşı yazım kuralı genelde benzerdir. Aşağıdaki yazım kuralı **INNER JOIN** komutu için yazılmıştır.

```
SELECT Alan_Adı FROM Tablo1 INNER JOIN Tablo2 ON Tablo1
Alan1=Tablo2.Alan2.
```

FROM ifadesinden sonra bağlanacak tablo adlarının arasında **INNER JOIN** ifadesi yer almaktır. **ON** ifadesinden sonra da bağlantı eşitliği tanımlanmaktadır. Bu kısımda farklı mantıksal ilişkilerle de tablolar bağlanabilir. Birden fazla tablonun bulunduğu ya da iç-içe sorguların kullanılmasında tablo isimlerini uzun olarak yazmaktansa bunun yerine kısa takma adlar kullanılabilir. Bu durumda sorgular daha kısa ve anlaşılır hâle gelebilir. Örneğin aşağıdaki sorguda [*Müşteri*] tablosu **FROM** ifadesinden sonra “**Müşteriler AS M**” şeklinde yazılarak bu sorgu içerisinde M takma adı ile kullanılmıştır. Benzer şekilde [*Siparişler*] tablosu da “**Siparişler AS S**” ifadesi ile artık “S” harfi ile kullanılmıştır.

```
SELECT M.Şirket, M.Soyadı, M.Ad, S.[Müşteri No], M.No
FROM Siparişler AS S INNER JOIN Müşteriler AS M ON S.
[Müşteri No] = M.No
```

JOIN ifadesi ile tabloları bağlama işlemlerini basit iki tablo kullanarak açıklamak anlaşılırlığını kolaylaştıracaktır. Aşağıda A ve B adında iki adet tablo olduğunu varsayıyalım. Bu iki tablonun birbirine **JOIN** komutu ile bağlanmasıında **INNER**, **OUTER** ve **CROSS** olmak üzere üç farklı seçenek bulunmaktadır.

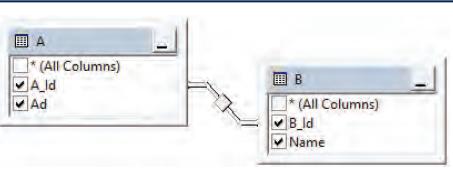
Sorgu içerisinde tablo adlarından sonra **AS** ifadesi kullanılarak takma bir adla ifade edilebilir. Alan adları için de aynı durum söz konusudur.

A Tablosu	
A_Id	Ad
1	Bir
2	İki
4	Dört
8	Sekiz

B Tablosu	
B_Id	Name
1	One
2	Two
3	Three
7	Seven

Tablo 6.3
A ve B Tablolarının ve
Tablo İçerikleri

Inner Bağlantı: Her iki tablonun da eşleşen alanlarının seçilmesi sağlanır. Diğer bir deyişle kesişim kümesi seçilir. Aşağıda [A] ve [B] tablosunun **INNER JOIN** komutu ile birleştirilen sorgu ve sonucu yer almaktadır. Seçme işleminin sonucunda her iki kümenin bağlantılı olduğu alandaki ortak değerlerin sayısı kadar satır görüntülenir.

Inner Join		Sonuç														
	<table border="1"> <thead> <tr> <th>A_Id</th> <th>Ad</th> <th>B_Id</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Bir</td> <td>1</td> <td>One</td> </tr> <tr> <td>2</td> <td>İki</td> <td>2</td> <td>Two</td> </tr> </tbody> </table>				A_Id	Ad	B_Id	Name	1	Bir	1	One	2	İki	2	Two
A_Id	Ad	B_Id	Name													
1	Bir	1	One													
2	İki	2	Two													
<pre>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A INNER JOIN B ON A.A_Id=B.B_Id</pre>																

Outer Bağlantıları: Outer bağlantıları soldan, sağdan ve tüm (Full) şeklinde üç farklı bağlantı türünden oluşmaktadır.

LEFT OUTER JOIN bağlantısı bu ifadenin solunda ve sağında kalan tabloların konumu baz alınarak soldaki tablonun tüm içeriğinin görüntülenmesi sağlar. Aşağıdaki tabloda **LEFT OUTER JOIN** solundaki [A] tablosunun tüm verisinin sonuçta yer alması sağlanmıştır. [B] tablosunda 4 ve 8 değerlerinin karşılığı olmadığı için **NULL** (boş) olduğu belirtilmiştir. Sonuçta soldaki tablonun eleman sayısı kadar satır görüntülenir.

Left Outer Join			
		Sonuç	
		A_Id	Ad
		1	Bir
		2	İki
		4	Dört
		8	Sekiz
<code>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A LEFT OUTER JOIN B ON A.A_Id= B.B_Id</code>			

LEFT OUTER JOIN bağlantısının tam tersi olan RIGHT OUTER JOIN ise ifade-nin sağında kalan tablodaki tüm değerlerin seçilmesini sağlar. Soru sonucundaki sağdaki tablonun satır sayısı kadar veri kümesi elde edilir.

Right Outer Join			
		Sonuç	
		A_Id	Ad
		1	Bir
		2	İki
		NULL	NULL
		NULL	NULL
<code>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A RIGHT OUTER JOIN B ON A.A_Id= B.B_Id</code>			

FULL OUTER JOIN ise her iki tabloda yer alan tüm satırların görüntülenmesini sağlar. Aşağıdaki tabloda ortak olan satırların bir kez yer aldığına dikkat ediniz. Bileşim kümesini elde eden bu bağlantıda kesim değerleri 1 kez yer aldığından eleman sayısı [A] tablosunun [B] den farklı eleman sayısı, [B] tablosunun [A] tablosundan farklı eleman sayısı ve Her iki kümenin ortak elemanlarının sayısının toplamından oluşur.

Full Outer Join			
		Sonuç	
		A_Id	Ad
		1	Bir
		2	İki
		4	Dört
		8	Sekiz
		NULL	NULL
		NULL	NULL
<code>SELECT A.A_Id, A.Ad, B.B_Id, B.Name FROM A FULL OUTER JOIN B ON A.A_Id= B.B_Id</code>			

Cross Bağlantı: Herhangi iki alanın eşleştirilmemiği bağlantı türü çapraz (cross) bağlantıdır. Bu bağlantıda [A] daki her bir satır için [B] deki tüm satırlar tekrarlanır. Böylece her iki tablonun eleman sayılarının çarpımı çapraz bağlantı sonucu oluşan listenin satır sayısını verir. Çapraz tablo bağlantısı şablon veri tablolarının üretilmesinde kullanılabilir.

Cross Join				
	A_Id	Ad	B_Id	Name
	1	Bir	1	One
	2	İki	1	One
	4	Dört	1	One
	8	Sekiz	1	One
	1	Bir	2	Two
	2	İki	2	Two
	4	Dört	2	Two
	8	Sekiz	2	Two
	1	Bir	3	Three
	2	İki	3	Three
	4	Dört	3	Three
	8	Sekiz	3	Three
	1	Bir	7	Seven
	2	İki	7	Seven
	4	Dört	7	Seven
	8	Sekiz	7	Seven

**SELECT A.A_Id, A.Ad, B.B_Id, B.Name
FROM A CROSS JOIN B**

Yukarıda örneklerle açıklanan bağlantı türlerinin yanı sıra bağlantı eşitliğinde düzenleme yapılarak farklı kümelere erişilebilir. Aşağıdaki örnekte [A] tablosundaki A_Id değerlerini [B] tablosundaki B_Id değerlerinden küçük olacak şekilde bağlanmıştır.

Inner Join				
	A_Id	Ad	B_Id	Name
	1	Bir	2	Two
	1	Bir	3	Three
	1	Bir	7	Seven
	2	İki	7	Seven
	2	İki	3	Three
	4	Dört	7	Seven

**SELECT A.A_Id, A.Ad, B.B_Id, B.Name
FROM A INNER JOIN B ON A.A_Id < B.B_Id**

Tablo 6.3'de yer alan A ve B tabloları ile hazırlanan bir sorguda A_Id ve B_Id sütunlarının birbirine eşit olmaması istendiğinde nasıl bir sorgu yazılabilir ve sonucu ne olurdu araştırınız?



SIRA SİZDE

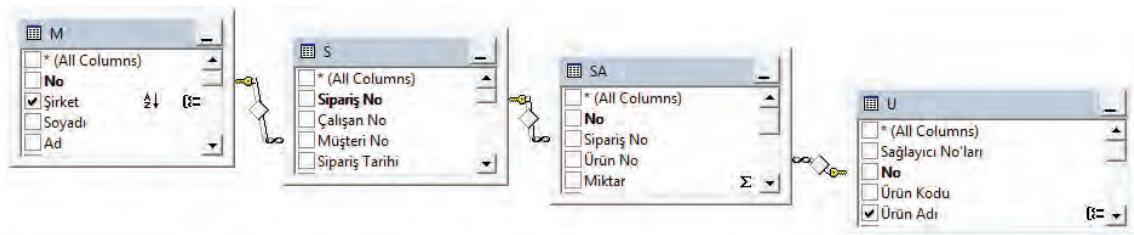
Çok Sayıda Tablonun Bağlanması

Çok sayıda tablonun birbirine bağlanması için farklı bir bağlantı türü yoktur. **JOIN** komutu aynı yapıyla kullanılır ancak **ON** kısmında birbiri ile ilişkili alanların doğru tariflenmesi gereklidir. Northwind örnek tablosunda hangi müşterinin hangi üründen kaç adet ürün sipariş ettiğini sorgulandığını düşünelim. Northwind tablosunun yapısı in-

celendiğinde [*Siparişler*] tablosunda müşteri ve sipariş numarasının bulunduğu, [*Sipariş Ayrıntıları*] tablosunda hangi siparişte hangi üründen ne miktarda sipariş edildiği bilgilerinin yer aldığığini görmektedir. Müşteri ve Ürün bilgileri de [*Müşteriler*] ve [*Ürünler*] tablosunda yer aldığına göre bu dört tablonun birbirine bağlanarak istenilen veri kümese ulaşılabilcegi açıklarıdır. Dört tabloyu birbirine bağlamak için aşağıdaki bağlantı ifadesinin yazılması gereklidir.

```
FROM Siparişler AS S
INNER JOIN [Sipariş Ayrıntıları] AS SA ON S.[Sipariş No] =
SA.[Sipariş No]
INNER JOIN Ürünler AS U ON SA.[Ürün No] = U.No
INNER JOIN Müşteriler AS M ON S.[Müşteri No] = M.No
```

Tabloların bağlantılarının oluşturulmasında **INNER JOIN** yapısı kullanıldığından sıralamanın önemi yoktur. Önemli olan **ON** ifadelerinden sonra hangi alanın hangi alanla bağlanacağıının belirtildiği eşitleme kısımdır. Bu alanda bir hatanın olması elde edilen veri kümescinin hatalı olmasına neden olur. Bu bağlantı türü grafik sorgu tasarım ekranında aşağıdaki gibi görüntülenir.



Veri Kümelerinin Birleştirilmesi

İki veya daha fazla veri kümescinin birleştirilmesi için veri kümelerinin aynı sayıda ve türde alanlarının olması gereklidir.

Tabloların birbirine bağlanması ilişkili alanların eşlenerek bağlanan tabloda karşılığının aranması için kullanılır. Veri birleştirme ise birbiri ile aynı sayıda ve türde alanları olan veri kümelerinin alt alta birleştirilmesi, kesişim kümelerinin ve farklarının bulunması işlemidir. Örneğin depolardaki stok miktarının hesaplanması depoya giren ve çıkan tablolarının birleştirilerek stok miktarının hesaplanması birleştirme sorguları kullanılır. Veri kümeleri **INTERSECT**, **EXCEPT**, **UNION**, **UNION ALL** komutları ile birleştirilebilir. Aşağıda tablo 6.2'deki A ve B tabloları için bu komutların kullanımı örneklenmiştir.

Intersect: İki veya daha çok veri kümescindeki satırların kesiminin bulunmasında kullanılır. Aşağıdaki sorgu ifadesi dikkatli incelediğinde iki farklı sorgunun arasına **INTERSECT** ifadesi kullanılarak birleştirildiği görülebilir. Sonuç veri kümescinin alan adının ilk sorgudaki alan adı olduğu görülmektedir.

<pre><code>SELECT A_Id FROM A INTERSECT SELECT B_Id FROM B</code></pre>	<table border="1"> <thead> <tr> <th>A_Id</th> </tr> </thead> <tbody> <tr> <td>1</td> </tr> <tr> <td>2</td> </tr> </tbody> </table>	A_Id	1	2	
A_Id					
1					
2					

Except: Bu ifade ile birleştirilen sorgular veri kümeleri arasındaki farkı listelemek için kullanılır. İlk sorgunun aşağıda verilen kümeden farkı belirlenerek görüntülenir.

<code>SELECT B_Id FROM B EXCEPT SELECT A_Id FROM A</code>	<table border="1"> <thead> <tr> <th>B_Id</th> </tr> </thead> <tbody> <tr> <td>3</td> </tr> <tr> <td>7</td> </tr> </tbody> </table>	B_Id	3	7	
B_Id					
3					
7					
<code>SELECT A_Id FROM A EXCEPT SELECT B_Id FROM B</code>	<table border="1"> <thead> <tr> <th>A_Id</th> </tr> </thead> <tbody> <tr> <td>4</td> </tr> <tr> <td>8</td> </tr> </tbody> </table>	A_Id	4	8	
A_Id					
4					
8					

Union ve Union All: `UNION` ve `UNION ALL` komutları kümelerin birleşim kümelemini elde etmek için kullanılır. `UNION` ifadesi ile birleştirilmiş veri kümelerinde kesişim kümesi bir kez yer alır. `UNION ALL` ile birleştirilmiş kümelerde ise herhangi bir işlem yapılmadan iki liste alt alta eklenir.

<code>SELECT A_Id FROM A UNION SELECT B_Id FROM B</code>	<table border="1"> <thead> <tr> <th>A_Id</th> </tr> </thead> <tbody> <tr> <td>1</td> </tr> <tr> <td>2</td> </tr> <tr> <td>3</td> </tr> <tr> <td>4</td> </tr> <tr> <td>7</td> </tr> <tr> <td>8</td> </tr> </tbody> </table>	A_Id	1	2	3	4	7	8			
A_Id											
1											
2											
3											
4											
7											
8											
<code>SELECT A_Id FROM A UNION ALL SELECT B_Id FROM B</code>	<table border="1"> <thead> <tr> <th>A_Id</th> </tr> </thead> <tbody> <tr> <td>1</td> </tr> <tr> <td>2</td> </tr> <tr> <td>4</td> </tr> <tr> <td>8</td> </tr> <tr> <td>1</td> </tr> <tr> <td>2</td> </tr> <tr> <td>3</td> </tr> <tr> <td>7</td> </tr> </tbody> </table>	A_Id	1	2	4	8	1	2	3	7	
A_Id											
1											
2											
4											
8											
1											
2											
3											
7											

Tabloların birleştirilmesinde alan sayıları eşleşmediğinde SQL Server “**All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.**” iletisi ile kullanıcıyı bilgilendirir. Yine benzer şekilde veri tipi uyusmaması durumunda ise “**Conversion failed when converting the nvarchar value ‘One’ to data type tinyint.**” benzeri bir hata üretilir.

GRUPLAMA VE ÖZETLEME SORGULARI

Seçme sorguları ile ilgili buraya kadar uygulanan örnek ve komutlarda tablodaki değerlerin sıralanması, listelenmesi ve birleştirilmesi gibi işlemler gerçekleştirildi. Veritabanı Yönetim Sistemlerinin önemli bir görevi ise kullanıcı isteklerine göre bu verilerin özetlenmesi, gruplanması işlemleridir. Bu sayede veri kümesi işlenerek mevcut sistem hakkında bilgi elde

Veritabanı tablolarında özetleme ve gruplama işlemlerinde **GROUP BY** komutu kullanılır

edilir. Aylık toplam satış tutarları, satışı yapılan ürün sayıları, en çok satış yapan personel listeleri ve benzeri bilgilerin elde edilmesi için kullanılan SQL komutu **GROUP BY**'dır.

GROUP BY komutu bir veri kümesinde belirlenen alanların içерdiği verinin tekrarsız hâle getirerek özetlenmesidir. Örnek veritabanında yer alan [*Siparişler*] tablosunda

“SELECT [Çalışan No] FROM Siparişler”

sorgusu çalıştırıldığında siparişi alan çalışan numaraları 47 satır olarak listeleneciktir. Diğer taraftan biz sadece hangi çalışanların satış yaptığı sorusunun yanıtını ariyorsak bu sorgu yerine

“SELECT [Çalışan No] FROM Siparişler GROUP BY [Çalışan No]”

sorgusunu çalıştırmalıyız. Bu sorguda **GROUP BY** kısmından sonra [*Çalışan No*] alanının yer aldığına dikkat edilmelidir. [*Çalışan No*] alanının tekrarsız olarak elde edilmesi sağlanarak 8 adet veri görüntülenecektir.

Gruplama işlemlerinde hangi alanın gruplanacağı ve hangi alanının görüntüleneceğini belirlemek çok önemlidir. Bu süreçte yapılan hatalar sorgunun çalışmamasına neden olacaktır. Örneğin kullanıcının aşağıdaki sorguyu yazması durumunda neler olabileceğini bir düşünelim.

“SELECT [Müşteri No] FROM Siparişler GROUP BY [Çalışan No]

Kullanıcı [*Siparişler*] tablosunda [*Çalışan No*] alanına göre gruplama yaparak [*Müşteri No*] alanını görüntülemek istemiştir. Sorgunun yapısı irdelenirse tablodaki bir alana göre gruplama yapılırken diğer bir alandaki değerlerinin görüntülenmesi mümkün değildir. Yukarıdaki bu hatalı istek veritabanı sistemi tarafından **“Column ‘Siparişler. Müşteri No’ is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.”** hata mesajı ile geri dönecektir. Çünkü [*Müşteri No*] alanı ile ilgili ya bir özetleme işlemi (sayma gibi) ya da gruplama yapılmalıdır. Şekil 6.2'de yukarıdaki hatalı sorgunun nasıl doğru ve anlamlı bir sorgu hâline getirildiği görülebilir. Şekildeki ilk sorguda Her iki alana göre gruplama yapılmış ancak tek bir alan görüntülenmiştir. Bu bir hata oluşturur çünkü gruplanan bir alanın görüntülenmesi zorunlu değildir. Şekildeki ikinciörnekte ise [*Müşteri No*] alanı **Count** fonksiyonu ile sayılmak suretiyle gruplanmadan yer almıştır. Diğer bir deyişle her bir çalışanın kaç satırda yer aldığı hesaplanmıştır.

Şekil 6.2

Group By Örnek Sorgular ve Sonuçları

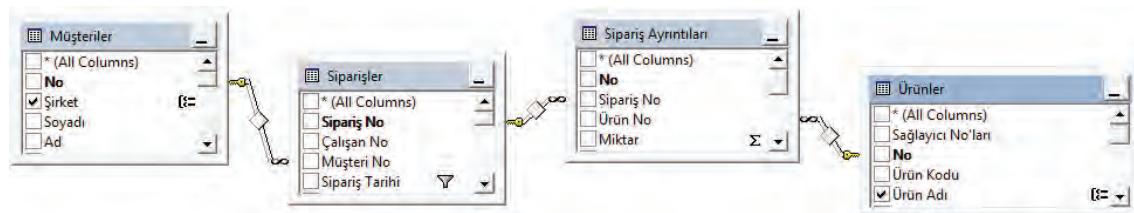
SQL Statement (Left Window)	Result (Left Window)	SQL Statement (Right Window)	Result (Right Window)																				
<pre>SELECT [Müşteri No] FROM Siparişler Group BY [Çalışan No],[Müşteri No]</pre>	<p>Müşteri No</p> <table border="1"> <tr><td>1</td><td>1</td></tr> <tr><td>2</td><td>7</td></tr> <tr><td>3</td><td>10</td></tr> <tr><td>4</td><td>11</td></tr> <tr><td>5</td><td>10</td></tr> </table>	1	1	2	7	3	10	4	11	5	10	<pre>SELECT [Çalışan No], Count([Müşteri No]) As MUSS_AY FROM Siparişler Group BY [Çalışan No]</pre>	<p>Çalışan No MUSS_AY</p> <table border="1"> <tr><td>1</td><td>12</td></tr> <tr><td>2</td><td>3</td></tr> <tr><td>3</td><td>6</td></tr> <tr><td>4</td><td>8</td></tr> <tr><td>5</td><td>1</td></tr> </table>	1	12	2	3	3	6	4	8	5	1
1	1																						
2	7																						
3	10																						
4	11																						
5	10																						
1	12																						
2	3																						
3	6																						
4	8																						
5	1																						

Gruplama işlemleri birden fazla tablo üzerinde gerçekleştirilen sorgulamalarda da uygulanabilir. Ünenin başında verilen seçme sorgularının mantıksal işleme sırasına göre **GROUP BY** işleminden önce tablolardan seçme işlemi ve varsa **WHERE** ile sınırlama işlemi uygulanır.

Northwind veritabanında 2006 Nisan ayında hangi müşterinin hangi ürününden kaçar adet sipariş verdığını elde eden sorguyu yazınız.

ÖRNEK 6.1

İstenilen verinin hazırlanması için öncelikle bu bilginin hangi tablolardan elde edileceğinin tespit edilmesi ve bu tabloların doğru şekilde bağlanması gereklidir. Daha sonrasında 2006 Nisan ayı için bir sınırlama ve daha sonra da ilgili alanları seçme ve gruplama işlemi gerçekleştirilebilir. Müşteri adı [*Müşteriler*] tablosundan, müşterilerin verdiği siparişler [*Siparişler*] tablosundan, siparişlerdeki ürünlerden kaçar adet verildiği [*Sipariş Ayrıntıları*] tablosundan ve Ürün isimleri ise [*Ürünler*] tablosundan elde edilebilir. Bu tablolar ilgili alanlardan birbirine bağlandığı takdirde hedef veri kümese ulaşılabilir. Bu dört tablo aşağıda grafik olarak gösterildiği gibi **FROM** komutu sonrası birbirine bağlanmıştır. Ancak tablolara kısa takma ad tanımlanarak sorgunun daha kolay yazılması sağlanmıştır.



Tablo bağlanması ardından kısıtlama komutu yazılmalıdır. Sipariş tarihi bilgisi [*Siparişler*] tablosunda [*Sipariş Tarihi*] adıyla yer almaktadır. Bu alanın kısıtlanması için tarih türündeki bir alanın ay ve yıl bilgisini elde edebilecek **YEAR()** ve **MONTH()** işlevlerinin kullanılması gereklidir. Satışların ürün ve müşteri bazında elde edilmesi istenildiğinden **GROUP BY** alanında [*Şirket*] ve [*Ürün Adı*] alanına göre gruplanması gereklidir. En son aşamada ise hangi alanlar sorgu sonunda görüntülenecek ya da hesaplanacak bu alanların **SELECT** kısmı sonrasında yazılması tamamlanır. Gruplanan Şirket ve ürün adı alanlarının seçmede kullanılması, oluşan verinin anlaşılması açısından önemlidir. Bir diğer önemli bilgi ise toplam kaçar adet siparişin alındığıdır. Bunun için de [*Sipariş Ayrıntıları*] tablosundaki [*Miktar*] bilgisinin gruplanan alanlara göre toplanması gereklidir. **SUM()** komutu sayısal değerlerin toplanması için kullanılan bir özetteme işlevdir. Bu işlevin **SELECT** kısmına yazılması ile sorgu aşağıdaki gibi tamamlanmış olacaktır.

```

    SELECT M.Şirket, U.[Ürün Adı], SUM (SA.Miktar) AS Toplam Miktar
    FROM Siparişler AS S
    INNER JOIN [Sipariş Ayrıntıları] AS SA ON S.[Sipariş No]
        = SA.[Sipariş No]
    INNER JOIN Müşteriler AS M ON S.[Müşteri No] = M.No
    INNER JOIN Ürünler AS U ON SA.[Ürün No] = U.No
    WHERE (YEAR(S.[Sipariş Tarihi]) = 2006) AND (MONTH(S.
        [Sipariş Tarihi]) = 4)
    GROUP BY M.Şirket, U.[Ürün Adı]
  
```

Sorgunun çalıştırılması sonucu Tablo 6.4'de verilen tablo elde edilecektir. Birçok tablonun bulunduğu ve karmaşık kısıtlamaların yer aldığı sorguları oluşturmak için kullanıcıların hatasız bir sorgu yazması sistematik bir yaklaşımı gerektirir.

Tablo 6.4
Örnek Uygulama İçin
Yazılan Sorgu Sonucu

Şirket	Ürün Adı	Toplam Miktar
Şirket F	Çikolata	10
Şirket H	Çikolatalı Bisküvi Karışımı	25
.....
Şirket BB	Yengeç Eti	50
Şirket Z	Yengeç Eti	30
Şirket Z	Zeytin Yağı	25

SIRA SIZDE



Örnek Uygulama 6.1'de 2006 Nisan ve 2007 Ocak ayındaki siparişlere ilişkin bilgi istenseydi WHERE ifadesi nasıl yazılmalıydı?

Özetleme İşlevleri

Özetleme İşlevleri graplama işlemlerinde sayma, toplama, ortalama ve benzeri işlemleri yerine getirirler.

Graplama sorgularında sorgulamak istenen veri kümesi **GROUP BY** ifadesinden sonraki alanlara göre gruplanırken diğer alanlar üzerinde de özetleme işlemleri gerçekleştirilir. Sayısal alanlar için sayma, toplama, ortalama gibi matematiksel işlemler, metin türündeki alanlar için ise sayma, en büyük ve en küçük değerlerin alınması gibi işlemler özetleme işlevleri tarafından gerçekleştirilir. SQL Server'da sık olarak kullanılan özetleme işlevleri ve açıklamaları Tablo 6.5'de verilmiştir.

Tablo 6.5
Özetleme İşlevleri ve
Anlamı

İşlev	Açıklaması
COUNT , COUNT_BIG	Bir gruptaki kayıt sayısını hesaplarlar. COUNT Int (tamsayı) veri tipi ile veri döndürürken, COUNT_BIG BigInt(büyük tamsayı) veri tipinde sonuç döndürür.
SUM	Sayısal veriler için kullanılır. NULL olan verileri dikkate almaz. Distinct komutu ile benzersiz değerlerin toplanması sağlanabilir.
AVG	Sayısal değerli alanların ortalamasını hesaplar.
MIN	Belirtilen alanın en küçük değerini getirir. NULL değerleri dikkate alınmaz. Metin değerler için de alfabetik sıraya göre en küçüğünü getirir.
MAX	Belirtilen alanın en büyük değerini getirir. NULL değerleri dikkate alınmaz. Metin değerler için de alfabetik sıraya göre en küçüğünü getirir.
STDEV	Belirtilen alan için standart sapma hesaplar.
STDEVP	Belirtilen alanın tüm değerleri için ana kütle standart sapmasını hesaplar.
VAR	Belirtilen alan için varyans hesaplar.
VARP	Belirtilen alanın tüm değerleri için ana kütle varyansını hesaplar.

Özetleme işlevleri işlem yapılan alandaki NULL olmayan, dolu olan alanlar için işlem yapmaktadır. İşlev adından sonra açılacak parantez içinde alan adından önce **DISTINCT** ifadesi konursa tekrarsız satırlar için işlem yapılmaktadır. Örneğin [*Sipariş Ayrıntıları*] tablosunda aşağıdaki iki sorgu çalıştırıldığında ilk sorgu 56 sayısını hesaplar. Bu ilgili tabloda [*Ürün No*] alanı boş olmayan 56 satır olduğu anlamına gelir. İkinci satır ise 24 sonucunu veriri ki bu da ilgili tabloda 24 farklı [*Ürün No*] yer aldığı anlamına gelir. Tekrar eden verilerin bir kere sayılacağı anlamına gelir.

```
SELECT COUNT([Ürün No]) FROM [Sipariş Ayrıntıları]
SELECT COUNT(DISTINCT [Ürün No]) FROM [Sipariş Ayrıntıları]
```

DISTINCT ifadesi diğer özetleme işlevleri ile de kullanılabilir.

ÖRNEK 6.2

Northwind veritabanında sipariş edilen ürünler aylık temelde raporlanmak istenmektedir. Buna göre her ay gerçekleşen siparişlerin toplam sayısı ve ortalama satış miktarları, yine aylık temelde satışlardan elde edilen toplam ciro tutarı ve aylara göre ortalamalarını hesaplayan SQL komutlarını oluşturunuz.

Northwind örnek veritabanında sipariş bilgileri 2006 yılına ait verilerde oluşmuş olsa da sorgunun tasarımindan yıl bilgisinin de graplama alanı olarak belirlenmesi daha ileride sorgunun doğru çalışmasını sağlayacaktır. Sorgunun hazırlanması için [Siparişler] tablosundaki [Sipariş Tarihi] ve [Sipariş Ayrıntıları] tablosundaki [Miktar] ve [Birim Fiyat] alanlarından oluşan bir veri kümesini oluşturmalı ve daha sonrasında yıl ve ay bilgisine göre graplama yapılmalıdır. Aşağıda bu amaçla oluşturulan sorgu incelendiğinde, sorgunun graplama satırında yıl ve ay bilgilerini elde eden fonksiyonlar kullanılarak yapıldığı görülmektedir. Sadece tarih bilgisine göre graplama yapılması durumunda, gün bilgilerini de içeren tarih veri türü veri kümesini günler bazında gruplayacak ve istenilen veri kümesi elde edilemeyecektir.

```
SELECT YEAR(S.[Sipariş Tarihi]) AS YIL,
MONTH(S.[Sipariş Tarihi]) AS AY,
SUM(SA.Miktar) AS Top_Miktat,
AVG(SA.Miktar) AS Ort_Miktar,
SUM(SA.[Birim Fiyat] * SA.Miktar) AS Top_Ciro,
AVG(SA.[Birim Fiyat] * SA.Miktar) AS Ort_Ciro
FROM Siparişler AS S INNER JOIN [Sipariş Ayrıntıları] AS SA
ON S.[Sipariş No] = SA.[Sipariş No]
GROUP BY YEAR(S.[Sipariş Tarihi]), MONTH(S.[Sipariş Tarihi])
```

Tarih alanından elde edilen Ay ve Yıl bilgisine göre graplama yapılan sorgunun seçme kısmında hesap yapılması istenen alanlara göre özetleme işlevleri kullanılmıştır. Toplama için **SUM** ve ortalama için **AVG** işlevi kullanılarak oluşan bu yeni alanlar için **AS** ifadesi ile birer takma ad verilmiştir. Ciro miktarının hesaplanmasında [Sipariş Ayrıntıları] tablosundaki [Birim Fiyat] ve [Miktar] çarpılarak ürünün satışı ile elde edilen tutarlar hesaplanmış ve toplanarak ve ortalaması hesaplanarak özetlenmiştir.

YIL	AY	Top_Miktar	Ort_Miktar	Top_Ciro	Ort_Ciro
2006	1	225	28	3836,00	479,50
2006	2	230	76	2241,50	747,1666
2006	3	1092	84	32609,25	2508,4038
2006	4	985	46	19355,25	921,6785
2006	5	95	23	1788,50	447,125
2006	6	315	35	8306,50	922,9444

Tablo 6.6
Örnek Uygulama 6.2'i
İçin Yazılan Sorgunun
Sonuç Tablosu

Tablo 6.5'de "**AVG(SA.Miktar) AS Ort_Miktar**" ifadesi ile elde edilen [*Ort_Miktar*] sutunu verisi incelendiğinde, verinin tamsayı değerlerinden oluşan olduğu görülmektedir. Toplam değer eleman sayısına bölünderek hesaplanan ortalamanın aslında tam sayı değil kesirli sayı olması beklenir. Bunun nedeni SQL Server'daki özetleme işlevlerinin sonuç olarak üzerinde hesaplama yapılan alanın türünde veri geri döndürmelidirler. "**Tinyint**", "**smallint**" ve **int** veri türleri **int**, **bigint** veri türü **bigint**, **float** ve **real** veri türleri **float** türünde veri geri döndürürler. Bu nedenle smallint veri türünde olan [Miktar] alanının orta-

Veri türünün dönüşümü için kullanılan **Cast** komutunun yazım kuralı **Cast(AlanAdı As Veritürü)** şeklidedir.

lama hesabı tamsayı (int) veri türünde geri dönmüştür. Ancak bu değerin kesirli sayı olarak hesaplanması istenirse veri türlerinin dönüştürülmesinde kullanılan **Cast** ve **Convert** komutları kullanılmalıdır. Ort_Miktar alanının kesirli sayı olarak hesaplanması için **AVG (Cast(SA.Miktar as real))** şeklinde yazmak yeterli olacaktır.

DİKKAT



Cast ve Convert işlevleri **Select** sorgularında görüntülenecek değerlerin veri türünü değiştirmek için kullanılır. Tablolarındaki alanların veri yapılarını kalıcı olarak değiştirmek için veri tanımlama dilindeki **ALTER** komutu kullanılır.

Özetlenen Değerlerin Sınırlanması

Gruplama sorgularında özetlenen değerlerin belirli koşullara göre sınırlaması için **HAVING** komutu kullanılır. **HAVING** komutunun mantıksal işleme sırası Select ifadesinden önce Gruplama işlevinden sonra yer almıştır. Dolayısıyla gruplama ve özetleme işlemi tamamlandıktan sonra koşullara uymayan satırlar seçilmez. **WHERE** komutu ise gruplama işleminden önce gerçekleştiği **WHERE** ile belirtilen koşullar Özetleme işlemlerine dahil edilmezler. Örnek uygulama 6.2'de tanımlanan gruplama ve özetleme ile elde edilen sorğunun aylık siparişin 50 nin üzerinde olan aylar için hesaplanması istensin. Bu durumda mevcut sorğunun sonuna aşağıda da görüleceği gibi **HAVING SUM(SA.Miktar)>50** satırının eklenmesi yeterli olacaktır.

```
SELECT YEAR(S.[Sipariş Tarihi]) AS YIL,
MONTH(S.[Sipariş Tarihi]) AS AY,
SUM(SA.Miktar) AS Top_Miktat,
AVG(Cast(SA.Miktar as real)) AS Ort_Miktar,
SUM(SA.[Birim Fiyat] * SA.Miktar) AS Top_Ciro,
AVG(SA.[Birim Fiyat] * SA.Miktar) AS Ort_Ciro
FROM Siparişler AS S INNER JOIN [Sipariş Ayırtıları] AS SA
ON S.[Sipariş No] = SA.[Sipariş No]
GROUP BY YEAR(S.[Sipariş Tarihi]), MONTH(S.[Sipariş Tarihi])
HAVING SUM(SA.Miktar)>50
```

SQL İŞLEVLERİ

SQL sorgularında çeşitli hesaplama ve mantıksal işlemleri gerçekleştirebilmek için işlevler bulunmaktadır. Microsoft SQL Server Transact-SQL dili farklı işlemler için farklı türde işlevler sunmaktadır. Bu üitede Transact-SQL dilinde hazır olarak sunulan ve seçme sorgularında kullanılabilen işlevlere yer verilecektir. Kullanıcıların ihtiyaçlarına yönelik işlevleri Transact-SQL programlama dili kullanarak oluşturabilmektedir. Ancak bu ünite ve bu kitapta programlamaya ilgili ayrıntıya yer verilmemiştir.

Birçok programlama dilinde farklı yazım şekilleriyle benzer görevleri yerine getiren işlevler mevcuttur. Bu işlevlerin çalışma prensibinin kavranması ve ihtiyaç olduğunda ilgili yardım dosyalarına ulaşarak bu işlevleri etkin kullanmak bir veritabanı kullanıcısı ya da programcısı için oldukça önemlidir.

Mantıksal İşlevler

Mantıksal işlevler **SELECT** sorgularında alanların değerlerini belirli koşullara göre farklı değerlere dönüştürürler. **CHOOSE**, **IIF**, **CASE** işlevlerinin yazım kuralları ve örneklerine aşağıda yer verilmiştir.

CHOOSE: İndeks değeri içeren bir değerin tanımlanan sıralı listede eşleştirerek karşı gelen elemanı görüntüler. **CHOOSE** (İndeks, 'Seç1', 'Seç2',,'SeçN') olarak yazılır. Northwind veritabanında [*Siparişler*] tablosundaki [*Durum No*] alanındaki değer için aşağıdaki sorgu yazılarak indekse karşı gelen metin listelenebilir. İndeks değeri sıfırdan başladığından **CHOOSE** komutunda 1 eklenerken yer almıştır.

```
SELECT [Durum No],
CHOOSE([Durum No]+1,'Yeni','Faturalandı','Sevk Edildi','Kapatıldı')
AS Durum
FROM Siparişler
```

IIF: Mantıksal bir karşılaştırma sonucu doğru ve yanlış durumları için iki ayrı değer görüntüler. Bu mantıksal işlev **IIF** (Mantıksal_ifade, Doğru_değer, Yanlış_Değer) şeklinde oluşturulur. Nortwind veritabanında [*Sipariş Ayrıntıları*] tablosunda [*Satınalma Siparişi No*] alanında sipariş alınan ürün için sağlayıcılardan gerçekleştirilen Satınalma numarası yer almaktadır. Bu alandaki değer NULL (boş) ise stoktan karşılandığı anlamına gelmektedir. Bu alanın kullanıcıya daha anlaşılır şekilde sunmak için IIF komutundan yararlanılabilir.

```
SELECT IIF([Satınalma Siparişi No] is NULL, 'Stoktan karşılandı',
cast([Satınalma Siparişi No] as nvarchar(10))+ 'nolu sipariş
ile temin edildi')
AS Tedarikdurumu
FROM [Sipariş Ayrıntıları]
```

Seçme sorgusu [*Satınalma Siparişi No*] alanı boş ise 'stoktan karşılandı', boş değilse ilgili sayısal değeri metine dönüştürerek siparişten temin edildiğini belirten bir metin oluşturmaktadır. Yukarıdaki sorgunun çalıştırılması durumunda aşağıdakine benzer bir sonuç listelenecaktır.

```
Tedarikdurumu
Stoktan karşılandı
Stoktan karşılandı
96 nolu sipariş ile temin edildi
97 nolu sipariş ile temin edildi
```

CASE: **CASE** komutu mantıksal işlevlerden farklı olarak birden çok koşulun tanımlanmasına olanak sağlar. **CASE** komutunun iki farklı yazım kuralı Tablo 6.7'de yazım kuralı ve örnek sorgularla açıklanmıştır.

Tablo 6.7
CASE Komutu Yazılım Kuralları ve Örnek Sorguları

Kullanım 1	Kullanım 2																				
<p>Yazım Kuralı: CASE ifade WHEN Değer1 THEN Değer1 WHEN Değer2 THEN Değer2 ELSE 'diğer' END</p> <p>Örnek Sorgu : SELECT [Durum No], CASE[Durum No] WHEN 0 Then 'Yeni' WHEN 1 Then 'Faturalandı' WHEN 3 Then 'Sevk Edildi' WHEN 9 Then 'Kapatıldı' Else 'Hata' END AS Durum FROM Siparişler</p>	<p>Yazım Kuralı: CASE WHEN Mantıksal_ifade THEN Değer1 WHEN Mantıksal_ifade THEN Değer2 ELSE 'diğer,' END</p> <p>Örnek Sorgu : SELECT Miktar*[Birim Fiyat] AS Ciro, CASE WHEN Miktar*[Birim Fiyat]<1000 THEN 'Az' WHEN Miktar*[Birim Fiyat]<10000 THEN 'Orta' WHEN Miktar*[Birim Fiyat]<100000 THEN 'Yüksek' ELSE 'Çok Yüksek' END AS CiroDurum FROM [Sipariş Ayrıntıları]</p>																				
<table border="1"> <thead> <tr> <th colspan="2">Sonuç:</th> </tr> <tr> <th>DurumNo</th> <th>Durum</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Yeni</td> </tr> <tr> <td>1</td> <td>Faturalandı</td> </tr> <tr> <td>9</td> <td>Kapatıldı</td> </tr> </tbody> </table>	Sonuç:		DurumNo	Durum	0	Yeni	1	Faturalandı	9	Kapatıldı	<table border="1"> <thead> <tr> <th colspan="2">Sonuç:</th> </tr> <tr> <th>Ciro</th> <th>CiroDurum</th> </tr> </thead> <tbody> <tr> <td>1400,00</td> <td>Orta</td> </tr> <tr> <td>105,00</td> <td>Az</td> </tr> <tr> <td>13800,00</td> <td>Yüksek</td> </tr> </tbody> </table>	Sonuç:		Ciro	CiroDurum	1400,00	Orta	105,00	Az	13800,00	Yüksek
Sonuç:																					
DurumNo	Durum																				
0	Yeni																				
1	Faturalandı																				
9	Kapatıldı																				
Sonuç:																					
Ciro	CiroDurum																				
1400,00	Orta																				
105,00	Az																				
13800,00	Yüksek																				

Tablo incelemesi birinci kullanımında tek bir değerin aldığı değere göre istenilen sonucun yazdırılması sağlanabilir. İkinci kullanım ise daha esnektr ve her koşul ifadesine farklı bir koşul ifadesi tanımlanabilemektedir. Nortwind veritabanında yer alan tablolarda uygulanan örnek sorgularda istenilen şartların sağlanması durumunda kullanıcıya metin görüntüleyen örnekler yer verilmiştir.

SIRA SİZDE



Case komutunun sıralama işlemlerinde nasıl kullanıldığı araştırınız.

Sayısal İşlevler

Veritabanı tabloları üzerinde gerçekleştirilecek sorgulamalarda gerekli olabilecek hazır matematiksel işlevler bulunmaktadır. Bu işlevler veri tablosunun bir alanı için kullanılabileceği gibi sabit bir değer verilerek de hesaplama yapılabilir. Ancak unutulmaması gereken bir konu seçme sorgusunun alan listesinde yar alan bir işlem ya da değer seçimini tüm satırları için tekrar ettiğidir. Tablo 6.8'de seçme sorgularında kullanılabilen işlevler yer almaktadır.

İşlev	Açıklaması	Örnek
ABS	Belirtilen değerin mutlak değerini veren matematik işlevidir.	ABS(-5) Sonuç: 5
SIGN	Sayısal değerin pozitif, negatif veya işaretsiz olduğunu döndürür	SIGN(-550) Sonuç: -1
ASIN, ACOS, ATAN, ATN2	Belirtilen trigonometrik değerlerin radyan cinsinden açı değerini hesaplar.	4*Atan(1) Sonuç: 3,14
SIN, COS, TAN, COT	Radyan cinsinden belirtilen değerlerin trigonometrik değerlerini hesaplar.	Tan(3.14/4) Sonuç: 1
RADIANS	Derece cinsinden verilen açı değerini radyan cinsine çevirir.	RADIANS(180.0) Sonuç: 3.14
DEGREES	Radyan cinsinden verilen açı değerini derece cinsine çevirir.	Degrees(PI()) Sonuç: 180
PI	Pi (π) katsayısını verir.	PI() Sonuç: 3,14159265358979
CEILING	Kesirli sayıları bir üst tamsayıya yuvarlar.	CEILING(-5.5) Sonuç: -5
FLOOR	Kesirli sayıları bir üst tamsayıya yuvarlar.	FLOOR(-5.5) Sonuç: -6
ROUND	Belirtilen sayı ya da alandaki değeri istenilen basamağa yuvarlar. İlk değer yuvarlanacak değer, ikinci değer yuvarlanacak basamak.	ROUND(56.55, -1) Sonuç: 60.00
EXP, LOG, LOG10, POWER	Üssel ve logaritmik işlem yapan işlevlerdir.	POWER(2,5) Sonuç: 32
RAND	0 ile 1 aralığında düzgün dağılmış rassal sayı üretir.	RAND() Sonuç: 0,5299
SQRT	Belirtilen değerin karekökünü hesaplar.	SQRT(25) Sonuç: 5
SQUARE	Belirtilen değerin karesini hesaplar.	SQRT(25) Sonuç: 625

Tablo 6.8
SQL Server
Matematisel İşlevleri

Metin İşlevleri

Metin türündeki alanların ya da değerlerin işlenmesine yönelik çok sayıda metin işlevi bulunmaktadır. Tablo 6.9'da sık kullanılan metin işlevleri, gerçekleştirdiği işlem ve örnek kullanımları verilmiştir. Örnekler bir sorgunun tamamını içermeyip sadece işlev ifadeleri için örnekleştirilmiştir. Bu ifadeler bir seçme sorgusunun farklı böümlerinde kullanılabilir.

SQL Server komut ve işlevlerine ulaşmak için internet kaynaklarına başvurulabilir. Internet tarayıcısında “String Functions SQL MSDN” arama kelimelerinin kullanılması ile tüm işlev listesine ulaşılabilmektedir.



Tablo 6.9
SQL Server
Matematiksel İşlevleri

İşlev	Açıklaması	Örnek
SUBSTRING	Metin türündeki bir değerin içerisindeki belirli bir bölümün alınmasını sağlayan işlevdir.	SUBSTRING ('YBS206U' ,4,3) Sonuç: 206
PATINDEX	Bir metin içerisinde belirli bir metin ifadesinin arayarak ilk bulunduğu konum bilgisini döndürür.	PATINDEX ('%YBS%', Belge) Sonuç: 50
LEN	Metindeki karakter sayısını döndürür	LEN ('YBS206U') Sonuç: 7
LEFT, RIGHT	Metin türündeki bir alan ya da değerin soldan(LEFT) veya sağdan(RIGHT) istenilen sayıda karakterin seçilmesini sağlar.	LEFT 'YBS206U',3) Sonuç: 'YBS'
REPLACE	Bir metin içerisinde belirli bir metni bularak o istenilen metin ile değiştirir.	REPLACE ('YBS206U' , '206' , '304') Sonuç: 'YBS304U'
CONCAT	Bir veya daha fazla metin değeri birleştirir.	CONCAT ('Veri' , 'tabanı' , 'I') Sonuç: 'Veritabanı I'
REPLICATE	Bir metinin değişkenin istenildiği kadar tekrarlanması için kullanılır.	REPLICATE ('0',5) Sonuç: '00000'
SPACE	İstenilen sayıda boşluk karakteri üretir.	'A'+ SPACE (3)+'B' Sonuç: 'A A'
NCHAR, CHAR	NCHAR Unicode karakter tablosunun, CHAR ise ASCII kod tablosunun belirtilen rakama karşı gelen karakterini getirir.	NCHAR (65) Sonuç: 'A'
UNICODE, ASCII,	UNICODE bir karakterin Unicode, ASCII ise ASCII kod tablosundaki kod karşılığını getirir.	ASCII ('A') Sonuç: 65
STR	Sayısal bir değeri istenilen hassasiyette metne çevirir.	STR (123.45, 6, 1); Sonuç:123.5

Tarih İşlevleri

Farklı kültürde aynı takvim kullanılsa da tarih bilgilerinin yazılması konusunda farklılıklar bulunmaktadır. Bu nedenle veritabanı yönetim sistemlerinde tarihin gösterilmesi, eklenmesi ile ilgili problemleri ortadan kaldırmak için çeşitli işlev ve formatlar bulunmaktadır. Bilgisayarlarda tarih ve saat bilgileri aslında bir sayı olarak saklanır. Bu sayının tamsayı kısmı gün, ondalıklı kısmı ise 24 saat 1 tamsayıya denk gelecek şekilde depolanmaktadır. SQL Server v veritabanı yönetim sistemi yazılımda Tarih ve saatle ilgili veri türleri Tablo 6.10'da verilmiştir. Tarih ve saat veri türleri ile ilgili temel işlevler aşağıda yer almaktadır.

Tablo 6.10
Transact-SQL Tarih
ve Saat Türleri
(Microsoft MSDN'den
Düzenlenmiştir.)

Veri türü	Birim	Aralık	Hassasiyet	Depolama (byte)
time	hh:mm:ss [nnnnnnnn]	00:00:00.000000 23:59:59.999999	100 nanosaniye	3 to 5
date	YYYY-MM-DD	0001-01-01 9999-12-31	1 gün	3
small datetime	YYYY-MM-DD hh:mm:ss	1900-01-01 2079-06-06	1 dakika	4
datetime	YYYY-MM-DD hh:mm:ss [.nnn]	1753-01-01 9999-12-31	0.00333 saniye	8
datetime2	YYYY-MM-DD hh:mm:ss [nnnnnnnn]	0001-01-01 00:00:00.000000 9999-12-31 23:59:59.999999	100 nanosaniye	6 - 8

DATENAME: Tarih veri türündeki bir değerin bir bölümünü isim olarak geri döndürür. “**DATENAME**(DatePart, Tarih_verisi)” şeklinde yazılan işlev verilen tarih verisinin ilgili kısmının karşılığını İngilizce olarak döndürür. **SELECT DATENAME**(month, '2016/06/19') ya da **SELECT DATENAME**(m, '2016/06/19') şeklinde yazılan komut “June” metin verisini geri döndürür.

DATEPART: Tarih veri türündeki bir değerin istenen bölümünün elde edilmesini sağlar. İşlev “**DATEPART**(DatePart, Tarih_verisi)” şeklinde yazılır. Örneğin bir tarih bilgisinin saniye değerini elde etmek için **SELECT DATENAME**(s, '2016/06/19 12:10:30.123') komutu yazılarak 30 sayısının geri dönmesi sağlanabilir.

DAY, MONTH, YEAR: Ünitede daha önceki örneklerde de kullanılan bu üç işlev sırasıyla bir tarih verisinin yıl, ay ve gün bilgisini sayısal olarak geri döndürür.

DATEFROMPARTS: Yıl, ay ve gün bilgilerinin sayısal olarak girildiği ve ilgili değerlerden tarih veri türünde değişken döndüren bir işlevdir. **SELECT DATEFROMPARTS**(2016, 6, 19) komutunun çalıştırılması ile “2016-06-19” şeklinde **Date** türünde veri döndürür. Bu işlev tarih verilerinin hatalı girilmesini engellemek için istemci yazılımlar tarafından kullanılabilmektedir.

DATEDIFF: İki tarih türündeki verinin arasındaki farkı istenilen tarih birimi olarak hesaplayan bir işlevdir. “**DATEDIFF**(DatePart, BaşlangıçTar,BitisTar)” olarak yazılır. Belirtilen iki tarih arasındaki farkı saat olarak hesaplanması istenen “**SELECT DATEDIFF**(hh, '2015/06/19 12:10', '2016/06/19 00:10')” bu komut 8772 sonucunu verecektir. İki tarih arasında 8772 saat fark olduğu hesaplanmıştır. Bu komut **Int** türünde veri döndüren bir işlevdir. Ancak hesaplamalarda eğer **Int** veri türünden daha büyük bir dönüş olacaksa **DATEDIFF_BIG** işlevi aynı parametrelerle kullanılabilir. Bu işlev **Bigint** veri türünde bir sayı döndürür.

DATEADD: Bir tarih türündeki veriye istenilen tarih birimi olarak ekleme işlemi gerçekleştiren işlevdir. “**DATEADD**(DatePart, Sayı, Tarihverisi)” yazım kuralı ile yazılan bu komutu bir önceki **DATEDIFF** işlevi için verilen örneğin sağlamaası için yazdığımızda (**SELECT DATEADD**(hh, 8772, '2015/06/19 12:10')) “2016-06-19 00:10:00.000” tarih bilgisini geri döndürdüğü görülecektir.

ISDATE: Bir metin türündeki verinin içeriğinin tarih bilgisi olup olmadığını kontrol ederek pozitif durumda 1, negatif durumda 0 değerini döndüren işlevdir. “**SELECT ISDATE**('2015/16/19 12:10')” işlevi 0 değeri döndürecektr.

ALT SORGULAR

Konusu seçme sorguları olan bu ünitenden birçok kısmında veri kümesi kavramı kullanıldı. Veritabanı yönetim sisteminde depolanan verilerin kullanıcıların taleplerine göre hazırlanmalarında oldukça karmaşık seçme sorgularını oluşturmak gerekmektedir. Bu nedenle oluşturulacak seçme sorguları birden fazla veri kümesi hâlinde hazırlanarak bir-birleriyle ilişkilendirmek, sınırlamak ya da birleştirmek suretiyle sistematik bir şekilde yapılandırılabilir. Her bir alt veri kümesine seçme sorgularında alt sorgu ismi verilmektedir. Alt sorgular karmaşık yapıdaki seçme sorgularının hazırlanmasında SQL komutlarının sağladığı esnekliklerden biridir.

Alt sorgular başka bir sorgunun herhangi bir kısmında (seçim, tablo, sınırlayıcı) parantez içerisinde eklenen seçme sorgularıdır. Alt sorgular bir üst sorguya bir değer ya da bir tablo çıktısı verebilirler. Alt sorgu alanları ile üst sorgu alanlarının birbiri ile ilişkilendirmek yoluyla da seçme sorguları hazırlanabilmektedir.

Bölümler	DatePart	Kısaltma
Yıl	year	yy, yyyy
Ceyrek	quarter	qq, q
Ay	month	mm, m
Yılın günü	dayofyear	dy, y
Gün	day	dd, d
Hafta	week	wk, ww
Saat	hour	hh
Dakika	minute	mi, n
Saniye	second	ss, s
Milisaniye	millisecond	ms
Milisaniye	microsecond	mcs
Nanosaniye	nanosecond	ns

SQL tarih işlevlerinde bir tarih verisinin ilgili bölümünü aşağıdaki tabloda yer alan Datepart sütunundaki ifadeler ya da kısaltma sütunundaki adlar ile ifade edilir.

Alt sorgu (subquery) bir seçme sorgusunun bir parçasını oluşturan ve parantez içinde yazılmış sql seçme sorgusuna denilir.

Sorgunun veri kaynağı olarak bir alt sorgunun kullanılması durumunda aşağıdaki yapıda sorgu oluşturulur. Altı çizili olarak gösterilmiş komutlar Alt sorgu olarak adlandırılır.

```
SELECT Ortalama, Alan1
FROM
(Select Alan1, AVG(Alan2) As Ortalama FROM Tablo5 GROUP BY Alan1)
AS Tablo2
```

Yukarıdaki seçme sorgusu bir tablodan değil parantez içerisinde yer alan başka bir veri kümelerinin (seçme sorgusunun) ürettiği listeyi kullanmaktadır.

Alt Sorguların Tablo Olarak Kullanılması

Ulaşılmak istenilen bazı veri kümelerinin SQL komutları ile hazırlanmasında birden fazla aşama gerekebilir. Alt sorguların tablo olarak kullanıldığı bir örnek uygulamayı adım adım gerçekleştirerek istenilen veri kümесini elde edelim.

ÖRNEK 6.3

Nortwind veritabanında işletmenin ürünlerine ait mevcut stok miktarlarını hesaplayan soruyu oluşturunuz.

Northwind veritabanında ürünlerde ait stok bilgileri [Stok Hareketleri] tablosunda yer almaktadır. Bu tablodaki [Hareket Türü] alanının değeri 1 ise ürün gelişi 2 ise ürün satışı anlamına gelmektedir. Stok miktarlarının hesaplanabilmesi için gelen ürünlerin pozitif gidenlerin ise negatif değerlerde gösterilmeleri ve daha sonra her ürün için bu değerlerin toplanması gerekmektedir. Bu işlem için iki adımlı bir yapı oluşturmak gerekektir. İlk adımda [Hareket Türü] değeri 1 olanlar için bir sorgu ve [Hareket Türü] değeri 2 olanlar için [Miktar] alanının değerini -1 ile çarpıldığı ikinci bir sorgu. Bu iki sorgunun alt alta birleştirilmesi için **UNION ALL** komutu kullanılarak bir veri kümesi hazırlanınsın. Bu veri kümescini hazırlayacak aşağıdaki sorgu komutları her bir stok hareketi için [*Ürün No*], [*Miktar*] verisini listeleyecek ancak ürün çıkışı için miktar değerlerini negatif olarak belirleyecektir.

```
SELECT [Ürün No], Miktar
FROM [Stok Hareketleri]
WHERE ([Hareket Türü] = 1)
UNION ALL
SELECT [Ürün No], Miktar *-1
FROM [Stok Hareketleri]
WHERE ([Hareket Türü] = 2)
```

Bu aşamadan sonra elde edilen veri kümescini [*Ürün No*] alanına göre grüplamak ve [*Miktar*] verisini toplamak her ürün için stok miktarını hesaplamak anlamına gelecektir. Bu amaçla yukarıda elde edilen veri kümescinin grüplama yapacak bir sorgunun tablosu olarak kullanmak yerinde olacaktır. Alt sorgu olarak yukarıdaki seçme sorgusunun bir başka sorguda parantez içinde kullanılması sağlanabilir. [*Stok*] adıyla yeni bir tablo gibi yer alan alt sorgunun alanları bir üst sorguda kullanılabilirmektedir. Aşağıdaki sorgu incelendiğinde [*Stok.Miktar*] alanının toplandığı, ürünlerde göre grüplama yapıldığı görülmektedir. Sorguya ürünler tablosunun eklenmesinin sebebi ise ürün adının da sonuçta görüntülenmemek istenmesidir. Aşağıdaki sorgunun çalıştırılması sonucu oluşan 28 kayıtlık listenin ilk 5 satırı Tablo 6.11'de verilmiştir.

```

SELECT Ürünler.[Ürün Adı], Stok.[Ürün No], SUM(Stok.Miktar) AS
Stok_Miktari
FROM (SELECT [Ürün No], Miktar
      FROM [Stok Hareketleri] WHERE ([Hareket Türü] = 1)
UNION ALL
      SELECT [Ürün No], Miktar * - 1 AS Expr1
      FROM [Stok Hareketleri] WHERE ([Hareket Türü] = 2)) AS Stok
INNER JOIN Ürünler ON Stok.[Ürün No] = Ürünler.No
GROUP BY Stok.[Ürün No], Ürünler.[Ürün Adı]
ORDER BY Stok_Miktari DESC

```

Ürün Adı	Ürün No	Stok_Miktari
Kahve	43	325
Yeşil Çay	81	125
Mantı	56	120
Ravyoli	57	80
Domates Sosu	66	80

Tablo 6.11
Örnek Uygulama
6.3 İçin Oluşturulan
Sorgunun Sonuç
Listesinin İlk Beş Satırı

Alt sorguların içinde de alt sorgu kullanılmaktadır. Hedef veri kümesine ulaşmak için çok sayıda iç içe sorgu kullanılmakta ancak daha sonra bu sorguların düzenlenmesinde zorluk ortaya çıkabilmektedir. Bu nedenle çok karmaşık sorgular, birbirini izleyen görünümler(view) olarak tasarılanabilmektedir.

Alt Sorguların Kısıtlayıcı Olarak Kullanılması

Alt sorgular tablo olarak kullanılabileceği gibi sınırlayıcı olarak da kullanılabilmektedir. **WHERE** komutu ile bir alandaki değerler başka bir sorgunun içeriği değerlerle sınırlanabilir.

Northwind veritabanında “İçecekler” kategorisindeki ürünlerden sipariş yapmamış müşteriler listesini oluşturunuz.

ÖRNEK 6.4

Muhtemelen bu kitap ile ilk kez seçme sorguları ile tanışan okurlar örnek uygulamada sorunun olumlu olması durumunda kolayca ilgili sorguyu yazabileceklerdi. Ancak dikkat edilirse bir ürün gurubundan alış veriş yapmayan müşteri listesi isteniyor. Tabloları birbirine bağlayarak ulaşabileceğimiz bir veri kümesi olmadığı açıkları. Bu durumlarda istenilen veri kümesinin değişilinden (tam tersinden) hareket edilmelidir. Öncelikle bu kategoride alış veriş yapan müşterileri elde eden bir sorgu oluşturulsun. İlgili üç tablo birbirine bağlanarak **WHERE (U.Kategori = 'İçecekler')** ifadesi ile ilgili müşterilerle sınırlanmış, zorunlu olmamakla beraber **[Müşteri No]** alanına göre gruplanmıştır. Gruplama gereksiz yere müşteri numaraların tekrarını engellemek için uygulanmıştır. Aşağıdaki sorgu çalıştırıldığında 9 adet müşteri numarasını listeleyecektir.

```

SELECT S.[Müşteri No] AS Müsteriler
FROM Siparişler AS S
INNER JOIN [Sipariş Ayırtıları] AS SA
ON S.[Sipariş No] = SA.[Sipariş No]
INNER JOIN Ürünler AS U ON SA.[Ürün No] = U.No
WHERE (U.Kategori = 'İçecekler')
GROUP BY S.[Müşteri No]

```

Müşreri listesinden çıkarılacak kümeyi bulduktan sonra **IN** komutunu **NOT** ifadesi ile uygulayarak hedeflenen veri kümese erişilebilir. Aşağıdaki sorguda dikkat edilmesi gereken **WHERE** komutu sonrasında [*Müşteriler*] tablosundaki [*No*] alanının nasıl sınırlandığıdır.

```
SELECT [No], Şirket
FROM Müşteriler
WHERE NOT ([No] IN
(SELECT S.[Müşteri No] AS Müsteriler
FROM Siparişler AS S INNER JOIN [Sipariş Ayrıntıları] AS SA
ON S.[Sipariş No] = SA.[Sipariş No]
INNER JOIN Ürünler AS U ON SA.[Ürün No] = U.No
WHERE (U.Kategori = 'İçecekler')
GROUP BY S.[Müşteri No])
```

Sorgunun çalıştırılması durumunda içecekler kategorisinde yer alan ürünlerden hiçbiri almamış 20 müşteri listelenenecektir.



DİKKAT

IN ile yapılan sınırlamalarda alt sorgunun sadece ilgili alanı kısıtlamak için veri türünde tek bir alan seçmelidir. Aksi durumda üst sorgu ilgili alanı hangi değerin içinde arayacağını bilmeyeceğinden hata mesajı oluşturacaktır. Böyle bir hatada kullanıcı “*Only one expression can be specified in the select list when the subquery is not introduced with EXISTS.*” ifadesi ile uyarılacaktır.

Alt sorgularla yapılan sınırlamalarda tek bir değer üreten alt sorgular da kullanılabilir mektedir.



SIRA SİZDE

5

Bir cümle ile “**SELECT [No] FROM [Sipariş Ayrıntıları] WHERE Miktar > (SELECT AVG (Miktar) AS Ort FROM [Sipariş Ayrıntıları])**” sorgunun elde ettiği veri kümесini tanımlayınız.

Alt Sorgu Üst Sorguların İlişkilendirilmesi

Alt sorgular bir üst sorgunun alanları ile sınırlanabilmektedir. Biraz karmaşık bir durum gibi görünse de veri kümeli mantığı ile değerlendirildiğinde bu yöntemle istenilen veri kümelerine etkili bir şekilde ulaşılabilenliği görülecektir.

ÖRNEK 6.5

Müşteriler tablosunda hiç sipariş vermemiş müşteri bulan sorguyu yazınız.

Bu sorgunun bir önceki konuda örnek uygulama 6.4'de uygulanan yöntemle istenilen veri kümeli bulunabilir. Diğer bir yöntem ise **EXISTS** komutudur. **EXISTS** komutu sorguya eklenen alt sorgu ile verilerin ilişkilendirilmesini sağlar. Üst sorguda [*Müşteriler*] tablosuna “M”, Alt sorguda ise [*Siparişler*] tablosuna [*AltSorgu*] adı verilmiştir. Alt sorgunun **WHERE** ifadesinden sonra yer alan “*AltSorgu.[Müşteri No]=M.[No]*” ifadesi incelenliğinde üst sorgu ile alt sorgunun nasıl bağlantıda olduğu anlaşılabilir. Alt sorguda yer alan müşterilerin üst listeden çıkarılması için **EXISTS** ifadesinden önce **NOT** operatörünün kullanılması önemlidir.

```
SELECT Şirket
FROM Müşteriler AS M
WHERE NOT EXISTS
(SELECT * FROM Siparişler AS AltSorgu
WHERE AltSorgu.[Müşteri No] = M. [No])
```

EXISTS ile yapılan sınırlamalarda alt sorgunun tüm alanlarının “*” simbolü ile seçildiğine dikkat edilmelidir. “*” ile tüm alanları seçilmeyen bir alt sorgu EXISTS ile kullanılamaz.



DİKKAT

Alt sorguların bir üst sorgunun tablosu, kısıtlayıcısı olabileceği gibi herhangi bir değer olarak da üst sorgu tarafından kullanılabilmektedir. Bu kullanıma örnek olması açısından aşağıdaki sorguyu dikkatle inceleyiniz.

```
SELECT [Ürün No], Miktar,  
Miktar-(Select AVG(Miktar) FROM [Sipariş Ayrıntıları]  
WHERE [Ürün No]=a.[Ürün No] ) AS Fark  
FROM [Northwind].[dbo].[Sipariş Ayrıntıları] as a
```

Komutlar incelendiğinde seçme alanında bir alt sorgunun yer aldığı görülmektedir. Bu sorgu [Sipariş Ayrıntıları] tablosunun her satırı için satışı yapılan ürünün, o ürün için yapılan tüm satışların ortalama miktardan ne kadar farklı olduğu hesaplanmaktadır. Sorguların satır bazlı oluşturulduğu düşünülürse her bir satıra gelindiğinde alt sorgu ilgili satırın ürün numarası ile bir ortalama hesabı yapıp geri dönecek ve aynı satırın miktar değerinden çıkaracaktır.

Bir sorgunun seçim kısmında yer alan alt sorgular ciddi performans kaybına neden olabilmektedir. Her satır için çalıştırılan sorgunun gerçekleştirdiği işlem, üst sorgu ile ilişkili olması ve sorgu sonucu oluşacak satır sayısı oluşan performans kaybı ile doğrudan ilişkilidir.

Özet



Seçme sorgularının yazım kuralını tanımlamak

Veri sorgulama dili veritabanı yönetim sistemlerinin en temel işlevlerinden biridir. Depolanan verilerin ihtiyacı olan kişiye ihtiyaç duyduğu ayrıntı ve şekilde ulaştırılması için kullanılan veri sorgulama dilinin veri seçimi ile ilgili komutu **SELECT** ifadesidir. Veri tablolarındaki tabloları birbirleriyle ilişkilendirilerek, birleştirilerek ya da sınırlandırılarak sorgulanması ya da verilerin özetlemesi ve gruplanması ile istenilen veri kümeleri hazırlanabilmektedir. **SELECT** komutunun yazım ifadesi aşağıdaki gibidir.

```
SELECT alan_listesi [INTO yeni_tablo]
[ FROM tablo_kaynağı ]
[ WHERE seçme_koşulları ]
[ GROUP BY gruplama_ifadeleri ]
[ HAVING Gruplanan_alan_kısıtlamaları ]
[ ORDER BY sıralama_düzeni ASC DESC ]
```

Select komutunu ile veri kümelerini hazırlamada seçme sorgusunun mantıksal işleme sırasının bilinmesi oldukça önemlidir. MS SQL Server yazılımında seçme sorgularında uygulanan işlemler sırasıyla FROM,ON,JOIN,WHERE, GROUP BY, WITH CUBE or WITH ROLLUP, HAVING, SELECT, DISTINCT, ORDER BY, TOP adımlarından oluşur.



Tablolardan birbirine bağlanma türlerini sıralamak

Tabloları bir sorgu içerisinde birbiri ile bağlamada **JOIN** komutu kullanılır. **JOIN** komutu ile sadece iki değil çok sayıda tablo birbirine bağlanabilmektedir. Ayrıca farklı bağlanma türleri ile kullanıcıların istenilen veri kümelerini elde etmelerine olanak sağlanır. Tabloları birbirine bağlarken her iki tablodaki ortak alanları birbirine esleyerek veri kümесini oluşturan **INNER JOIN**, bir tablonun ya da tüm tabloların değerlerinin hepsini kapsayan **OUTER JOIN** ve tablolara arası eşleştirme yapmadan çaprazlama yapan **CROSS JOIN** bağlantı türleri uygulanabilmektedir. Tabloları birbirine eklemek, kesişim ya da birleşim kümelerini elde etmek için **INTERSECT**, **EXCEPT**, **UNION**, **UNION ALL** komutları kullanılmaktadır.



Gruplama ve özetleme işlemlerini tanımlamak

Gruplama bir veri kümesinde belirlenen alanların içerdigi verinin tekrarsız hâle getirerek özetlenmesidir. Bu işlem sırasında gruplanmanın yapılacak alanlar sorguda **GRO-UP BY** ifadesinden sonra yazılır. Gruplama alanı dışındaki alanlar ise bir özetleme işlevi ile tek bir değer hâline getirilirler. Özetleme işlevleri olarak **COUNT**(Sayma), **SUM** (Toplama), **AVG**(Ortalama), **MIN**(En küçük değer), **MAX**(En büyük değer), **STDEV**(Standart sapma), **VAR**(Varyans) ifadeleri kullanılabilmektedir. Özetleme işlevleri ile elde edilen değerlerin sınırlanması için **HA-VING** ifadesi kullanılır. Bu komut örneğin Aylık satış ortalamasının belirli değer üstünde ya da altında olan ürünlerin listelenmesi sağlanabilmektedir.



SQL işlevlerini sıralamak

SQL komutlarında belirli işlemleri hızlı olarak yerine getirebilmek için kullanılmış çok sayıda işlev bulunmaktadır. Seçme sorgularında sıkça kullanılan mantıksal, sayısal, metin ve tarih işlevleri bu bölümde ele alınmıştır. Mantıksal işlevler seçme sorgularında bir alandaki değerin belirli şartlara uyması durumunda yerine açıklayıcı bir ifadenin görüntülemesini sağlayabilmektedir. **CHOOSE**, **IIF** ve **CASE** işlevleri mantıksal işlevler olarak kullanılmaktadır. Sayısal ve metin işlevleri genellikle diğer programlama dillerinde olduğu gibi matematiksel ve metin işleme işlevlerini içerirler. Tarih bilgilerine yönelik işlevler de bir veritabanı yönetim sisteminin sık kullanılan işlevleridir.



Alt Sorgu yapılarını tanımlamak

Veritabanı yönetim sisteminde depolanan verilerin kullanıcıların taleplerine göre hazırlanmalarında oldukça karmaşık seçme sorgularını oluşturmak gerekmektedir. Bu nedenle oluşturulacak seçme sorguları birden fazla veri kümlesi hâlinde hazırlanarak birbirleriyle ilişkilendirme, sınırlamak ya da birleştirmek suretiyle sistematik bir şekilde yapılandırılabilir. Her bir alt veri kümese seçme sorgularında alt sorgu ismi verilmektedir. Alt sorgular karmaşık yapıdaki seçme sorgularının hazırlanmasında SQL komutlarının sağladığı esnekliklerden birdir. Seçme sorgularında alt sorgular tablo, alan, ya da bir krit olarak yer alabilir. Krit olarak bir alt sorgunun kullanılmasında **IN** ve **EXISTS** sınırlama yapıları kullanılır. Ayrıca alt sorgu ve üst sorgu alanlarının ilişkilendirilerek veri kümelerinin sınırlanması mümkün olabilmektedir.

Kendimizi Sınayalım

1. Aşağıdaki SQL ifadelerinden hangisi sorgunun veri kaynağı ile ilgilidir?
 - a. Select
 - b. Having
 - c. From
 - d. Order by
 - e. Where
2. Aşağıdakilerden hangisi Select komutunun mantıksal işleme sırasının en sonunda yer alır?
 - a. Top
 - b. Join
 - c. Order By
 - d. Having
 - e. Where
3. Aşağıdaki komutlardan hangisi ile bir seçme sorgusunda tablolar birbirine bağlanabilir?
 - a. Order by
 - b. Top
 - c. Group By
 - d. Distinct
 - e. Where
4. INTERSECT ifadesi iki veya daha fazla veri kümesi arasında hangi işlemi gerçekleştirir?
 - a. Birleştirme
 - b. Fark
 - c. Ekleme
 - d. Güncellemeye
 - e. Kesişim
5. Distinct ifadesi aşağıdakilerden hangisi ile ilgilidir?
 - a. Ortalama hesaplama
 - b. Toplama işlemi
 - c. Gruplama
 - d. Benzersiz değerlerle işlem
 - e. Standart sapma
6. Aşağıdaki özetleme işlevlerinden hangisi metin türündeki veri ile kullanılabilir?
 - a. AVG
 - b. MAX
 - c. STDEV
 - d. SUM
 - e. VAR
7. Choose işlevi aşağıdakilerden hangi işlev kategorisinde yer alır?
 - a. Metin İşlevleri
 - b. Tarih İşlevleri
 - c. Sistem İşleleri
 - d. Sayısal İşlevleri
 - e. Mantıksal İşlevler
8. Tarih işlevlerinde aşağıdaki kısaltmalardan hangisi ay bilgisini ifade eder?
 - a. yy
 - b. m
 - c. mi
 - d. ms
 - e. mcs
9. DateADD komutunun işlevi nedir?
 - a. O anki tarih bilgisini getirir.
 - b. Metin bilgisini tarih bilgisine çevirir.
 - c. Tarih verisine istenilen birimde ekleme yapar.
 - d. İki tarih arasındaki süreyi hesaplar.
 - e. İki tarihi birleştirir.
10. Aşağıdakilerden hangisi ile alt sorgu ifadesi **kullanılamaz**?
 - a. Group By
 - b. From
 - c. IN
 - d. Exists
 - e. Select

Kendimizi Sınayalım Yanıt Anahtarları

1. c Yanıtınız yanlış ise “Seçme (Select) Komutu” konusunu yeniden gözden geçiriniz.
2. a Yanıtınız yanlış ise “Seçme (Select) Komutu” konusunu yeniden gözden geçiriniz.
3. e Yanıtınız yanlış ise “Birden Fazla Tablodan Veri Seçimi” konusunu yeniden gözden geçiriniz.
4. e Yanıtınız yanlış ise “Veri Kümelerinin Birleştirilmesi” konusunu yeniden gözden geçiriniz.
5. d Yanıtınız yanlış ise “Özetleme İşlevleri” konusunu yeniden gözden geçiriniz.
6. b Yanıtınız yanlış ise “Özetleme İşlevleri” konusunu yeniden gözden geçiriniz.
7. e Yanıtınız yanlış ise “Mantıksal İşlevler” konusunu yeniden gözden geçiriniz.
8. b Yanıtınız yanlış ise “Tarih İşlevleri” konusunu yeniden gözden geçiriniz.
9. c Yanıtınız yanlış ise “Tarih İşlevleri” konusunu yeniden gözden geçiriniz.
10. a Yanıtınız yanlış ise “Alt Sorgular” konusunu yeniden gözden geçiriniz.

Sıra Sizde Yanıt Anahtarları

Sıra Sizde 1

SQL sorgularındaki koşul satırlarının her bir satırda kontrol edileceği hatırlanırsa “**WHERE ([Ürün Adı] = 'Ceviz') AND ([Ürün Adı] = 'Zeytin Yağı')**” koşulunun bir kayıt içerisinde [Ürün Adı] alanının hem Ceviz hem de Zeytin Yağı metnine eşit olması durumu aranmaktadır. Bir alanın bir kattyta farklı iki değere eşit olması mümkün değildir. Dolayısıyla bu sorgu veritabanı verilerinin içeriğine bakılmadan herhangi bir sonuç döndürmeyeceğini açıklar. Eğer iki koşul arasında **OR** ifadesi olsaydı anlamlı bir sorgu ifadesi olurdu.

Sıra Sizde 2

A_Id ve B_Id değerlerinin birbiri ile aynı olmadığı bir veri kümesi elde edilmek istenildiğinde bağlantı kısmına A **INNER JOIN B ON A.A_Id <> B.B_Id** ifadesinin yazılması gerekiydi. Bu durumda her iki tabloda aynı olan değerler sonuç kümesinde yer almazdı. Aşağıda bağlantı şekli, sorgu ifadesi ve sonuç kümesini bulabilirsiniz.

**SELECT A.A_Id, A.Ad, B.B_Id, B.Name
FROM A INNER JOIN B ON A.A_Id <> B.B_Id**

A_Id	Ad	B_Id	Name
1	Bir	2	Two
1	Bir	3	Three
1	Bir	7	Seven
2	İksi	7	Seven
2	İksi	3	Three
2	İksi	1	One
4	Dört	1	One
4	Dört	2	Two
4	Dört	3	Three
4	Dört	7	Seven
8	Sekiz	7	Seven
8	Sekiz	3	Three
8	Sekiz	2	Two
8	Sekiz	1	One

Sıra Sizde 3

Kısıtlama işlemleri için yazılan ifadeler arasında kullanılan mantıksal operatörlerin doğru kullanımı çok önemlidir. Bu neden için SQL komutlarının taleplerinin satır bazlı işlentiği unutulmamalıdır. 2006 Nisan ve 2007 Ocak aylarındaki verinin görüntülenmesi için “**YEAR(S.[Sipariş Tarihi]) = 2006**”, (**MONTH(S.[Sipariş Tarihi]) = 4**, **YEAR(S.[Sipariş Tarihi]) = 2007**), (**MONTH(S.[Sipariş Tarihi]) = 1**) kısıtlarının nasıl birleştirileceğini düşünelim. Öncelikle bir kaydatta tarih bilgisinin yılı 2006 ve ayının 4 olması veya yılı 2007 ve ayının 1 olması gereklidir. İşte cümle hâline getirilmiş bu ifadenin kısıt olarak **WHERE** den sonra yazılmış hâli aşağıdadır. (**YEAR(S.[Sipariş Tarihi]) = 2006 AND MONTH(S.[Sipariş Tarihi]) = 4**) OR (**YEAR(S.[Sipariş Tarihi]) = 2007 AND MONTH(S.[Sipariş Tarihi]) = 1**)

Yararlanılan ve Başvurulabilecek Kaynaklar

Sıra Sizde 4

Sıralama işleminin belirlenen alanlardaki değerlere göre yapılmaması **CASE** komutu ile sağlanabilir. Ancak bu dinamik bir süreçtir. Seçme işlem mantık sırasının son adımı olan sıralama işlemi uygulanırken ilgili satırın değerlerine bakılarak sıralama ölçüyü değiştirilebilir. Örneğin Aşağıdaki soruda [*İş Ünvani*] değeri 'Sahibi' değerini alınca [*İş Ünvani*] alanına göre diğer durumlarda işe [*Şehir*] alanının değerine göre sıralama yapılır.

```
SELECT *
FROM [Northwind].[dbo].[Müşteriler]
Order By
CASE [İş Üvancı] when 'Sahibi' THEN [İş Üvancı]
ELSE Şehir END
```

Sıra Sizde 5

"Ortalama miktarın üzerinde sipariş edilen sipariş bilgileri" ifadesi sorgunun sözel karşılığıdır.

Elmasri, R. ve Navathe, S. B. (2015). **Fundamentals of Database Systems**, United Kingdom: Pearson.

Microsoft SQL Server MSDN [https://msdn.microsoft.com/en-us/library/mt590198\(v=sql.1\).aspx](https://msdn.microsoft.com/en-us/library/mt590198(v=sql.1).aspx)

7

Amaçlarımız

Bu üniteyi tamamladıktan sonra;

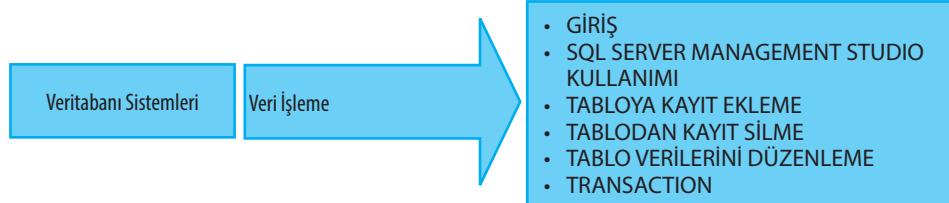
- 🕒 Veri işleme ve veri işleme komutlarının gerekliliğini açıklayabilecek,
- 🕒 INSERT, DELETE ve UPDATE komutlarını örnekler verebilecek,
- 🕒 Birden fazla tablo ile eylem sorgularının örnek uygulamalarını yapabilecek,
- 🕒 TRANSACTION, ROLLBACK ve COMMIT ifadelerinin kullanımını açıklayabilecek

bilgi ve becerilere sahip olacaksınız.

Anahtar Kavramlar

- Veri İşleme
- SQL Komutları
- INSERT, DELETE ve UPDATE Komutları
- TRANSACTION, ROLLBACK ve COMMIT

İçindekiler



Veri İşleme

GİRİŞ

Veritabanı Sistemleri, Word, Excel ya da PowerPoint gibi ofis yazılımlarının dosya yapısından farklı olarak birlikte çalışması gereken nesnelerin (tablo, görünüm, indeks, saklı yordam vb.) oluşturduğu bir topluluktur. Bahsedilen tüm bu nesnelerin birbiri ile uyumlu ve düzgün çalışabilmesi için bir takım tasarım ilkelerine uyması gerekmektedir. Bu tasarım ilkelerinin kötü tasarılanması veya tasarım kurallarına uyulmaması hâlinde problemlerle karşılaşılması kaçınılmaz olur ve başarısız veya tümüyle işe yaramayan veritabanı uygulamaları ortaya çıkabilir. Veritabanı tasarımını yapılrken sadece tablolar ve tablolar arasındaki bağlantıların nasıl olacağına karar verilmez. Hangi tablonun hangi amaçla kullanılacağı önemlidir.

Bu kitap kapsamında örnek uygulamaların gerçekleştirildiği Northwind veritabanında yer alan [*Müşteriler*] tablosunu incelersek, tablonun ilgili şirketin satış işlemi gerçekleştirdiği birey ya da firmalara ait bilgilerinin tutulduğunu görürüz. Aynı tablo müşterilere ait sipariş bilgilerinin de depolanması için tasarlanabilirdi. Ancak bu durumda her siparişte müşteri bilgilerinin tekrar edilmesi gerekirdi. Bu ve bunun gibi sebepler ilişkisel veritabanı sistemlerinin veri tekrarı ve kaybının engellenmesi, veri ekleme, güncelleme ve işlememe kolaylığının sağlanması için geliştirilmiş yapılardır. Ayrıca bir veritabanında yer alan tablolar, indeksler ve benzeri yapıların tasarılanması sistemin hatasız ve yüksek performansla çalışması için oldukça önemlidir. Veritabanında yer alan tablo sayısı, tablolar arasındaki ilişkilerin karmaşıklığı; verilerin güncelleme, ekleme ve silme işlemlerini etkilemektedir.

Verilerin tablolara eklenmesi tabloların tasarımını ile yakından ilişkilidir. Her tablo bir kullanım amacıyla oluşturulurken tablodaki alanların her biri yalnızca bir tür veri kabul edecek şekilde tanımlanmalıdır. Örneğin, [*Müşteriler*] tablosundaki [*No*] alanına rakam, [*Adres*] alanına ise metin girilebilecek şekilde tasarılanması uygun olacaktır. Eğer rakam girilecek bir şekilde tasarılmış bir alana metin girilmesi hâlinde veritabanı hata mesajı üretecektir. Bu sayede her alan için yanlış bilgilerin girilmesini engellemek mümkündür. Genelde veritabanı tasarımını yapılrken bir tablodaki alanların her biri yalnızca bir tür veri kabul eder. Ancak bu katı, değişmez bir kural değildir. [*Müşteriler*] tablosundaki [*Posta Kodu*] alanı metin veri türüne ayarlanmış olsa bile bu alanda sayılar depolanabilir ancak o veri üzerinde toplama, çıkartma vb. gibi aritmetik hesaplamalar doğrudan yapılamaz. Bazı istisnalar dışında, bir kayıttaki alanlar yalnızca bir değer kabul etmelidir. Aynı tablodaki [*Adres*] alanına birden fazla adres girilmemesi gereklidir. Bu yaklaşım, sınırlı türde veri kabul edecek şekilde ayarlanmadıkça, varsayılan olarak hücrelere istediğiniz kadar ad, adres veya resim girmenize olanak tanıyan Excel programı ile tamamen farklı bir durumdur. Bir veritabanına veri girmenin temel olarak iki yolu vardır.

Bunlardan ilki veritabanına klavyeden kullanıcı girişi ile istenilen değerlerin girilmesidir. Bu durum veritabanı yapısının bilgi girmesi gereken kişi tarafından görülmeye neden olabilir. Ayrıca tabloların her alanı için hangi tür verinin girilmesi gerektiği bilinmelidir. Bunun yerine veritabanına veri girerken bir uygulama tarafından hazırlanacak olan veri giriş formları kullanmak daha uygun olacaktır. Veri giriş formları, veritabanına verilerin daha kolay, daha hızlı ve daha doğru girilmesini sağlayabilir. Veri giriş formları kullanılarak aynı anda bir veya birden fazla tablonun farklı alanlarına bilgi girişini sağlamak mümkün olmaktadır. Örneğin, hazırlanacak bir form ile örnek veritabanındaki *[Müşteriler]* ve *[Siparişler]* tablolarının gerekli alanlarına aynı anda veri girişini sağlamak mümkündür. Ayrıca bir tablodaki alanlardan bazlarının alabileceği değerleri bilgi girişi yapacak kişiye seçenekli olarak sunulabilir veya bazı giriş alanları da otomatik olarak doldurarak kullanıcının daha kolay ve daha az hata yapabilmesi sağlanabilir. *[Müşteriler]* tablosundaki *[Şehir]* alanına bilgi girişi yapacak kişinin şehir ismini yazması yerine açılır metin girici ile seçim yaparak girmesi hem bilgi girişini hızlandıracak hem de hatalı girişlerin azalmasını sağlayacaktır. Oluşturulacak bilgi girişi formu ile bazı alanlarda otomatik olarak veri tanımlanabilmektedir. Örnek veritabanındaki *[Siparişler]* tablosunun alanını varsayılan değer olarak sistem zamanını getiren bir fonksiyonu kullanmaktadır. Böylece *[Sipariş Tarihi]* verisinde hem hatalı giriş hem de format hataları engellenmiş olur. Formlarda listeler, metin kutuları, düğmeler ve veri sayfaları gibi denetimlerden veri giriş ve düzenleme işlemlerinde kullanılabilir. Buna karşılık, formdaki denetimlerin her biri esas alınan bir tablo alanından veri okur veya bu tablo alanına veri yazar. Belirli bir denetimin kullanım kısıtları; esas alınan tablo alanı için ayarlanmış olan veri türüne, bu alan için ayarlanmış özelliklere ve veritabanı tasarımcısının her bir denetim için belirlediği bazı özelliklere bağlıdır. Örneğin, müşterilerin çoğunluğu Eskişehir ilinden ise veri giriş formunda varsayılan olarak Eskişehir seçili olarak gelebilir. Bununla beraber, telefon alanı için seçilen ile göre şehir alan kodunun da veri giriş formu tarafından otomatik olarak doldurulması sağlanabilir. Bu gibi yapılabilecek her türlü farklı uygulamalar sayesinde hem daha hızlı hem de daha doğru bilgilerin girişi sağlanabilir.

Bir veritabanı uygulamasının temel amacı kayıtların veritabanında saklanması ve bu kayıtların istenildiği zaman sorgulanabilmesidir. Ancak bazı durumlarda veritabanına girilen kayıtların silinmesi gerekmektedir. Hatalı kayıtlar, güncellliğini yitirmiş kullanılmayan kayıtların veritabanından silinmesi gerekmektedir. Örneğin, bir firmada çalışan personel herhangi bir sebeple firmadan ayrıldığı zaman o personele ait bilgilerin veritabanında saklanması ihtiyaç kalmayabilir. Veritabanında kayıt silme işlemi iki şekilde yapılmaktadır: Bir tablodaki herhangi bir kaydın silinmesi ikincisi de o tablodaki tüm kayıtların silinmesidir.

DİKKAT



Veritabanında bir kayıt silme işlemi gerçekleştiğinde sadece belirtilen tablodaki kayıt veya kayıtlar silinmekte yanı tablo fiziksel olarak ortadan kaldırılmamaktadır.

Bazı durumlarda ise tablonun bir veya birden fazla alanındaki değerlerin değiştirilmesi gerekebilir. Örneğin, personellerimizden birinin soyadı evlilik dolayısıyla değişmiş olabilir. Böyle bir durumda *[Çalışanlar]* tablosundaki *[Soyadı]* alanında tutulan personelin soyadının yeni soy adı ile değiştirilmesi/güncellenmesi gerekmektedir. Bu değiştirilme işlemi iki şekilde yapılabilir. Bunlardan bir tanesi soy adı değiştirilmek istenen personel ile ilgili kaydın tablodan silinip, soy adı değiştirilmiş olarak tüm bilgileri ile yeni kayıtın veritabanına eklenmesidir. Bu işlemi yapabilmek için ilk önce silme işlemi daha sonra ise yeni kayıt işlemi olmak üzere iki farklı işlemin yapılması gerekmektedir. Bunun yerine sadece değiştirilmek istenen kayittaki alanın değerinin değiştirilmesi daha kolay bir işlemidir.

Veri işleme komutları kullanarak bir tabloya yeni bir kayıt eklemek, mevcut kayıt üzerinde değişiklik yapmak veya kaydı silmek tek başlarına kullanılabileceği gibi bu işlemlerin tek bir adımda arkaya arkaya yapılması da gerekebilir. Örneğin, bir firmaya bir ürün satın

almak üzere bir istek geldiği zaman, ilk önce ürünün stoklarda olup olmadığı kontrolünün yapılması, ürün var ise bu konuda bir siparişin oluşturulması, bu ürünün müşteriye gönderilmesi ve firmadaki stoktan düşülmesi arka arkaya yapılması gereken işlemlerdir. Veri işleme komutları kullanarak tüm bahsedilen işlemler ayrı ayrı yapılabilir. Ancak tüm bu işlemler aslında tek bir işlemin alt kollarıdır ve bütün işlemlerin beraber gerçekleştirilmesi gerekmektedir. İşlemlerden bir tanesinin yapılamaması durumunda diğer işlemlerin de iptal edilmesi gerekmektedir. SQL dili arka arkaya yapılması gereken bir veya birden fazla veri işleme komutlarının hepsini tek bir işlem altında toplama imkânı vermektedir. Ayrıca grup içinde bir işlemin başarısız olması hâlinde yapılan işlemlerin geri alınabilmesini, işlemler tamamlandığı durumda ise kalıcı değişikliklerin yapılmasını sağlar.

SQL SERVER MANAGEMENT STUDIO KULLANIMI

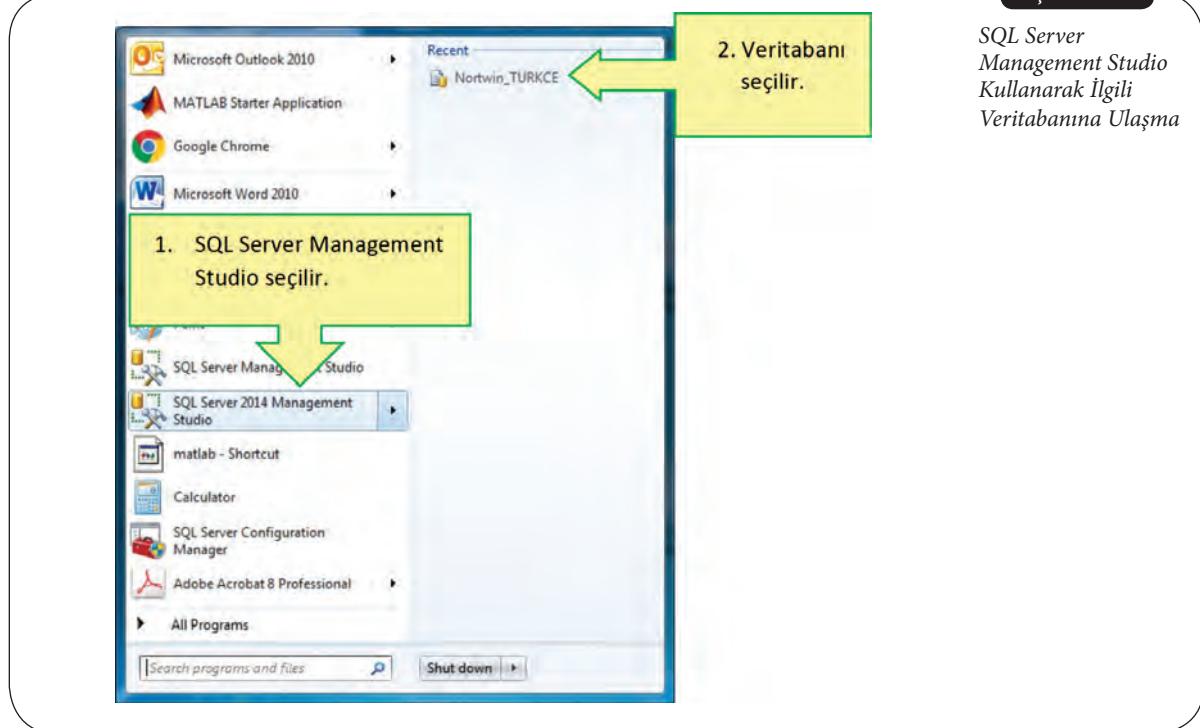
SQL Server Management Studio kullanılarak veritabanına klavyeden kullanıcı girişleri ile istenilen değerlerin girilmesi, silinmesi veya değiştirilmesi mümkündür.

Örnek Northwind veritabanında yer alan [Çalışanlar] tablosundaki verilere klavye yolu ile ekleme, silme ve güncelleme işlemi yapınız.

ÖRNEK 7.1

Bunun ilk önce SQL Server Management Studio'nun açılması gerekmektedir. Başlat tuşuna tıklanılarak SQL Server Management Studio'ya ulaşılıp Şekil 7.1'de görüldüğü gibi ilgili veritabanı seçimi yapılır.

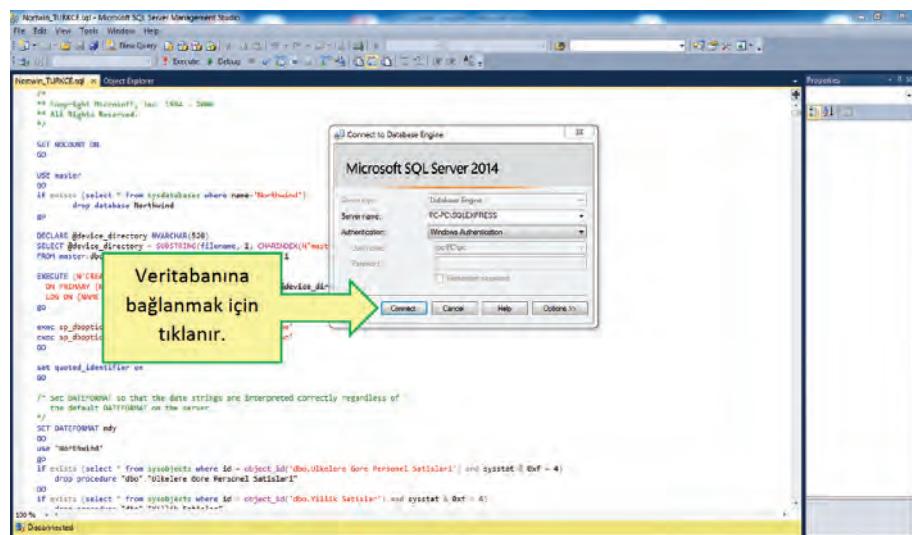
Sekil 7.1



SQL Server Management Studio programlar menüsünden çalıştırıldığı zaman ilk önce veritabanına bağlanması için Şekil 7.2'de görüldüğü gibi veritabanına bağlama işleminin gerçekleştirileceği bağlantı ekranı ile karşılaşılır. Bağlama ekranında bağlanılacak veritabanının ismi (Server Name) ve Authentication seçimi yapılır. Veritabanı kendi bilgisayarımız üzerinde çalışıyor ise Windows Authentication modunda bağlantı düğmesine (Connect) basılarak SQL Server Management Studio'ya bağlanma işlemi gerçekleştirilir. Bağlama işlemi için gerekli olan adımlar Şekil 7.2'de gösterilmiştir.

Sekil 7.2

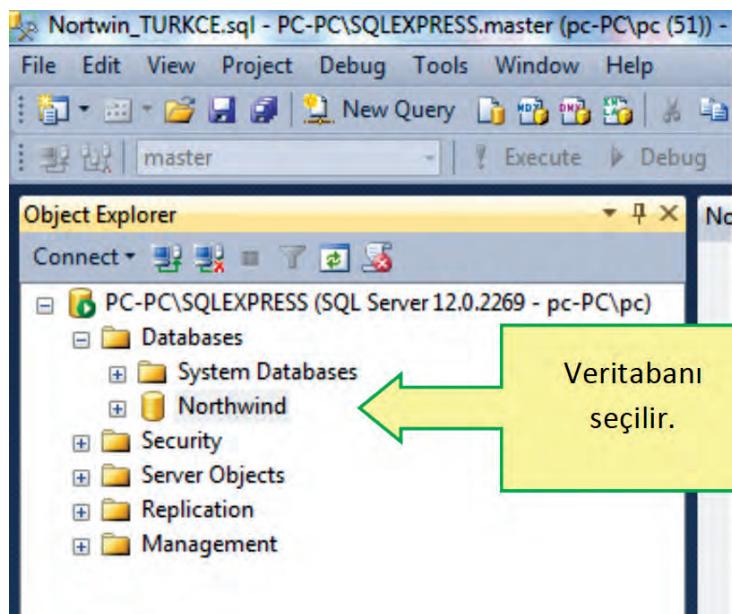
*SQL Server
Management
Studio Kullanarak
İlgili Veritabanına
Bağlanması İşlemi*



Veritabanına bağlanma işleminden sonra açılan pencerenin sol tarafında Object Explorer alanında SQL Server Management Studio üzerinde yer alan veritabanları ve diğer nesneler bulunmaktadır. Burada görülen her öğrenin yanında bulunan artı işaretine tıklanarak genişletme işlemi gerçekleşir. Genişletme işlemi gerçekleştiği zaman artı işaret eksi işaret hâlini alır. Eksi işaretine tıklanarak daraltma işlemi yapılabilir. Object Explorer alanı aşağıdaki şekilde gösterilmektedir.

Sekil 7.3

*Object Explorer
Alanının Gösterimi*



Object Explorer alanında sistemde kurulu olan tüm veritabanları, veritabanları içinde yer alan tüm tablolar ve tablolara ait olan alanlar seçim yapılarak görülebilmektedir. Bununla beraber tablodaki her alanın özelliklerini görebilme ve yetki dahilinde değişirebilme işlemlerini de buradan yapabilme imkanı mevcuttur. Örnek bir tablo ve alanları aşağıdaki şekilde görülebilir.

Şekil 7.4

Object Explorer

- Northwind
 - Tables
 - dbo.Calisanlar
 - Columns

PC-PC\SQLEXPRESS...d - dbo.Calisanlar

Column Name	Data Type	Allow Nulls
No	int	<input type="checkbox"/>
Şirket	nvarchar(50)	<input checked="" type="checkbox"/>
Soyadı	nvarchar(50)	<input checked="" type="checkbox"/>
Ad	nvarchar(50)	<input checked="" type="checkbox"/>
[E-posta Adresi]	nvarchar(50)	<input checked="" type="checkbox"/>
[İş Unvanı]	nvarchar(50)	<input checked="" type="checkbox"/>
[İş Telefonu]	nvarchar(25)	<input checked="" type="checkbox"/>
[Ev Telefonu]	nvarchar(25)	<input checked="" type="checkbox"/>
[Cep Telefonu]	nvarchar(25)	<input checked="" type="checkbox"/>
[Faks Numarası]	nvarchar(25)	<input checked="" type="checkbox"/>

Column Properties

(General)

- (Name) Şirket
- Allow Nulls Yes
- Data Type nvarchar
- Default Value or Binding
- Length 50

Tablo ve Alanlarının Gösterimi

Örnek olarak [*Çalışanlar*] tablosundaki veriler değiştirilmek istenildiği zaman aşağıdaki şekilde de görüldüğü gibi verilerinde değişiklik yapılmak istenen tablo seçimi yapıldıktan sonra farenin sağ tuşuna basılır. Açıılır pencere ekranında 'Edit Top 200 rows' seçeneği seçilir.

Şekil 7.5

Object Explorer

- dbo.Calisanlar
- Columns

PC-PC\SQLEXPRESS...d - dbo.Calisanlar

Context menu for the 'Calisanlar' table:

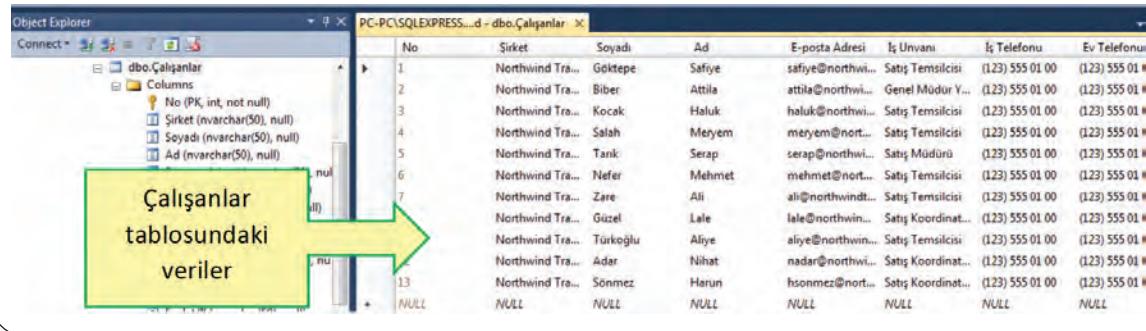
- Table...
- Design
- Select Top 1000 Rows
- Edit Top 200 Rows** (highlighted with a green arrow)
- Script Table as
- View Dependencies
- Full-Text index
- Policies
- Facets
- Start PowerShell
- Reports
- Rename
- Delete
- Refresh
- Properties

Tablodaki Verilerin Değiştirilmesi

Verileri değiştirilmesi istenen tablodaki veriler Object Explorer alanın sağ tarafında görülmektedir.

Şekil 7.6

[Çalışanlar] Tablosundaki Veriler



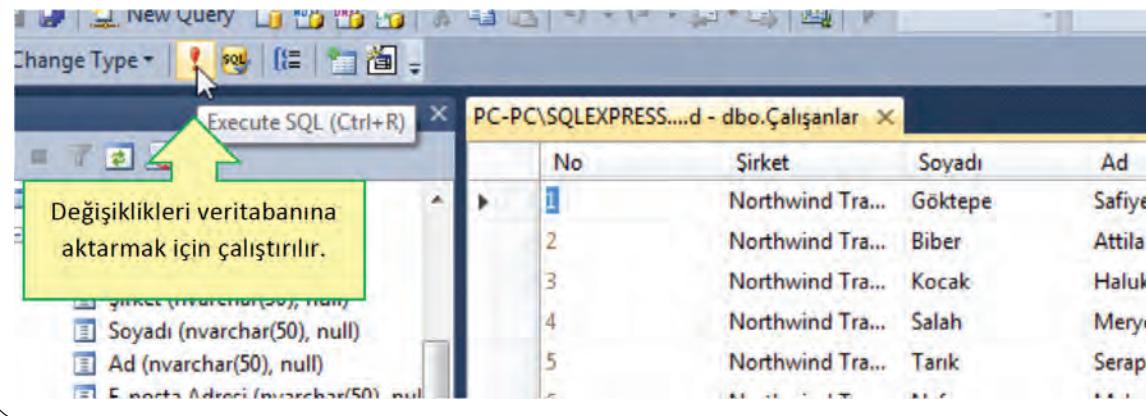
The screenshot shows the Object Explorer on the left with 'dbo.Çalışanlar' selected. The 'Columns' node under it lists 'No (PK, int, not null)', 'Şirket (nvarchar(50), null)', 'Soyadı (nvarchar(50), null)', and 'Ad (nvarchar(50), null)'. The main window displays the 'Çalışanlar' table with 13 rows of data. A yellow callout box points to the table with the text: 'Çalışanlar tablosundaki veriler'.

No	Şirket	Soyadı	Ad	E-posta Adresi (nvarchar(50), null)	İş Unvanı	İş Telefonu	Ev Telefonu
1	Northwind Tra...	Göktepe	SafİYE	safие@northwi...	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 #
2	Northwind Tra...	Biber	Atilla	attila@northwi...	Genel Müdür Y...	(123) 555 01 00	(123) 555 01 #
3	Northwind Tra...	Kocak	Haluk	haluk@northwi...	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 #
4	Northwind Tra...	Salah	Meryem	meryem@nort...	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 #
5	Northwind Tra...	Tank	Serap	serap@northwi...	Satış Müdürü	(123) 555 01 00	(123) 555 01 #
6	Northwind Tra...	Nefer	Mehmet	mehmet@nort...	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 #
7	Northwind Tra...	Zare	Ali	ali@northwindt...	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 #
8	Northwind Tra...	Güzel	Lale	lale@northwi...	Satış Koordinat...	(123) 555 01 00	(123) 555 01 #
9	Northwind Tra...	Türkoglu	Aliye	aliye@northwin...	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 #
10	Northwind Tra...	Adar	Nihat	nadar@northwi...	Satış Koordinat...	(123) 555 01 00	(123) 555 01 #
11	Northwind Tra...	Sönmez	Harun	hsonmez@nort...	Satış Koordinat...	(123) 555 01 00	(123) 555 01 #
12	NULL	NULL	NULL	NULL	NULL	NULL	NULL
13	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Değiştirilmek istenen değer üzerine gelerek klavye ile bilgi girişi yapılır. Bilgi girişi yapıldıktan sonra aşağıdaki şekilde de gösterildiği gibi Execute komutu ile değişikliklerin veritabanına aktarılması sağlanır.

Şekil 7.7

[Çalışanlar] Tablosundaki Verilere Yapılan Değişikliklerin Aktarılması



The screenshot shows the 'Execute SQL (Ctrl+R)' button being clicked. A yellow callout box points to this button and the text: 'Değişiklikleri veritabanına aktarmak için çalıştırılır.' Below the button, the 'Change Type' dropdown is set to 'SQL'. The main window shows the 'Çalışanlar' table with 13 rows of data.

No	Şirket	Soyadı	Ad
1	Northwind Tra...	Göktepe	SafİYE
2	Northwind Tra...	Biber	Atilla
3	Northwind Tra...	Kocak	Haluk
4	Northwind Tra...	Salah	Meryem
5	Northwind Tra...	Tank	Serap
6	Northwind Tra...	Nefer	Mehmet
7	Northwind Tra...	Zare	Ali
8	Northwind Tra...	Güzel	Lale
9	Northwind Tra...	Türkoglu	Aliye
10	Northwind Tra...	Adar	Nihat
11	Northwind Tra...	Sönmez	Harun
12	NULL	NULL	NULL
13	NULL	NULL	NULL

Yukarıda anlatılan işlemler ile veritabanındaki istenilen bir tablonun verilerine ekleme, silme ve değiştirme işlemi yapılabilir. Ancak böyle bir yaklaşım, veritabanı yapısının bilgi girmesi gereken kişi tarafından görülmesine neden olur. Ayrıca tablolardan her alanı için hangi tür verinin girilmesi gerektiği bilinmelidir. Bununla beraber farklı tablolara veri girilmesi için her tablo için aynı işlemlerin yapılması gerekmektedir. Yapılması zor olan bu işlemin yerine veritabanında kayıt ekleme, silme ve güncelleme işlemleri için DML (Data Manipulation Language) kullanılır. İllerleyen kesimlerde DML komutlarına örnek vermek için Northwind veritabanı tabloları kullanılmıştır.

DİKKAT



SQL dilinde Veri İşlemleri Dili (DML, Data Manipulation Language) kullanılarak, veritabanı tablosuna kayıt ekleme, silme ve güncelleme işlemleri yapılır.

TABLOYA KAYIT EKLEME

Veritabanı yönetim sistemlerinde verilerin ilişkisel olarak depolandığı tablolara birçok farklı yöntemlerle veri girilebilmektedir. Klavyeden el ile veri girişi, diğer tablolardan elde edilecek veriler, dış kaynaklardan alınabilecek verilerin tablolara eklenmesi gerçekleştirilebilir. Veritabanı tablolarında her bir satır ilgili tablonun gerçek dünyada temsil varlığını

bir örneğidir. Dolayısıyla bir tabloya veri eklemeye yeni bir nesne, satır ya da kayıt eklemeye işlemidir. Veritabanı tablolarında işlem tabloları gibi yeni sütun eklemeye çok yaygın değildir. Bunun nedeni veritabanı tasarılanırken tabloda depolanacak varlığın özelliklerinin (tablonun sütunlarının) başlangıçta belirlenmesidir. Tablolara sütun (alan) eklemeye veri tanımlama dilindeki **ALTER** komutu ile gerçekleştirilmektedir. Veritabanı tablolarına bir veya daha çok satır (kayıt) eklemek için SQL dilinde **INSERT INTO** deyi̇mi̇ kullanılır. Bu işlem “Ekleme Sorgusu” olarak adlandırılır.

INSERT INTO Komutu

Bir tabloya doğrudan veya başka tablolardan sorgulama sonucu elde edilen kayıtları eklemek için **INSERT INTO** komutu kullanılır. Bu komut bir tabloya istenilen değerlerin yazılıarak girilmesi ya da başka bir sorgu ile elde edilen veri kümelerinin girilmesi için kullanılabilir. **INSERT INTO** komutunun bu iki farklı işlem için yazım kuralı aşağıda verilmiştir.

```
-- Doğrudan veri ekleme
INSERT INTO TabloAdı [(alan1, alan2, ...)]
VALUES (değer1, değer2, ...)

-- Sorgu sonucunun tabloya eklenmesi
INSERT INTO TabloAdı [(alan1, alan2, ...)]
SELECT değer1, değer2, ... FROM Tablo
```

INSERT INTO komutundaki TabloAdı verinin ekleneceği tablo adını, parantez içesindeki alan1, alan2 eklenecek tablodaki alan adlarını işaret etmektedir. Alan adlarının köşeli parantez içinde verilmesi alan adlarının mutlaka tanımlama zorunluluğu olmadığını gösterir. Ancak bu durumda tablonun tüm alanlarına sırası ile verinin girilmesi gereklidir. Eklenecek değerler Values ifadesini takiben parantez içerisinde belirtilmelidir. Buradaki önemli ayrıntı alan ve değerlerin eşleşmesidir. Değer1, alan1'e ve değer2, alan2'ye eklenecektir. Dolayısıyla alan sayısına eşit değerin komutta yer olması zorunludur. Değer ve alanlar virgül ile ayrılmaktadır.

Sorgu sonucunun bir tabloya eklenmesi için kullanılan komut yapısında ise **VALUES** yerine **SELECT** ile başlayan sorgu ifadesi yer almaktadır. Sadece başka bir tablodan değil kullanıcının tanımlayacağı bir veri kümelerinden elde edilen sorgu sonucu ilgili tabloya eklenir. MS SQL Server'in son yillardaki sürümlerinde aynı sunucudaki başka veritabanından elde edilen sorgu sonuçları da tablolara eklenebilmektedir.

SQL dilinde tablolara kayıt eklemek için **INSERT INTO** komutu kullanılır.

INTO ifadesi MS SQL Server ve birçok veritabanı yönetim sisteminde zorunlu değildir. Bu ifade bir sonraki tabloya veri giri̇si yapıldığını gösteren İngilizce dil yapısından gelen bir bağlaçtır.



DİKKAT

Örnek Northwind veritabanında yer alan [Siparişler] tablosuna yeni bir sipariş bilgisi DML komutları kullanarak ekleyiniz.

ÖRNEK 7.2

INSERT INTO yazım kuralında da verildiği gibi öncelikle eklenmek istenilen tablo ve tablo alanları tanımlanmalıdır. Bu nedenle ilk kısımda tablonun alanları teker teker yazılır. Örnek veritabanındaki alan adlarında Türkçe harflerin ve boşlukların bulunması alan adlarının köşeli parantez ya da çift tırnak içine alınmasını gerektirmektedir. **VALUES** bölümünde ise alan adları ile aynı sıradan ve istenilen veri türünde tabloya eklenecek değerler yazılmalıdır. Örneğin ilk alan siparişi alan çalışana ait bir numaradır. Benzer şekilde

tüm eklenecek bilgiler veri türlerine göre sırası ile yazılır. Metin ifadeler tek tırnak içerisinde belirtilmelidir. Rakamlarda ise ondalık ayraç her zaman İngilizce dil yazım kuralı nedeniyle nokta şeklinde yazılır. Tarih bilgisi de “Yıl-Ay-Gün” biçiminde tanımlanmalıdır. Tarih ve saat gösterimleri dile bağlı olduğu için istenir ise dönüşüm sağlayan işlevler ile de kullanılabilir. Tablo adından sonra açılan parantez alan adlarından sonra kapatılmalı ve aynı şekilde Values ifadesinden sonra açılan parantez de son ifadeden sonra kapatılmalıdır. Aşağıdaki ekleme komutu bu kurallar dikkate alınarak oluşturulur.

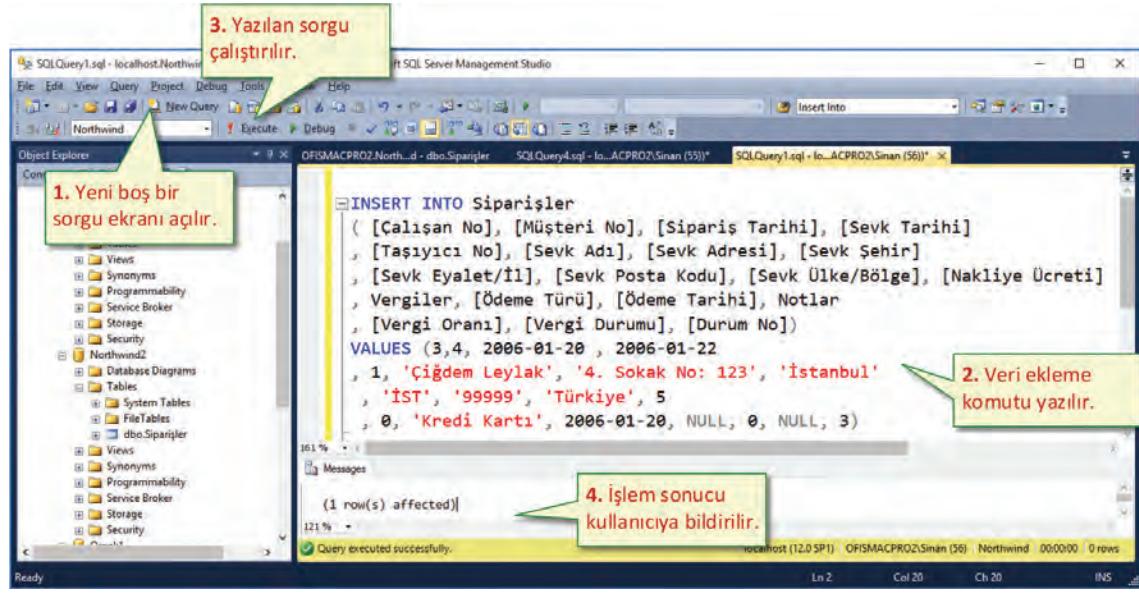
INSERT INTO Siparişler

```
([Çalışan No], [Müşteri No], [Sipariş Tarihi], [Sevk Tarihi],
[Taşıyıcı No], [Sevk Adı], [Sevk Adresi], [Sevk Şehir],
[Sevk Eyalet/İl], [Sevk Posta Kodu], [Sevk Ülke/Bölge],
[Nakliye Ücreti], Vergiler, [Ödeme Türü], [Ödeme Tarihi], Notlar,
[Vergi Oranı], [Vergi Durumu], [Durum No])
VALUES (3,4, 2006-01-20 , 2006-01-22, 1, 'Çiğdem Leylak',
'4. Sokak No: 123', 'İstanbul', 'İST', '99999', 'Türkiye', 5, 0,
'Kredi Kartı', '2006-01-20', NULL, 0, NULL, 3)
```

Şekil 7.8'de ekleme sorgusunun çalıştırılma adımları yer almaktadır. Buna göre yeni bir sorgu ekranı oluşturularak komut yazılır ve “Execute” komutu ile çalıştırılır. Çalıştırılan komut sonrası ekranın alt bölümünde sonuç ifadesi görüntülenir. Tabloya kayıt ekleme işleminin başarıyla sonuçlandığı “(1 row(s) affected)” ifadesiyle bildirilir. Bu işlem sonucunda tabloda 1 satırın etkilendiği bildirilir. Eğer hata ile sonuçlanırsa kırmızı renkte bir hata ifadesi ile bildirilecektir.

Şekil 7.8

[Siparişler] Tablosuna Kalayıyeden Girilen Değerlerin Eklenmesi İçin Oluşturulan Insert Into Komutunun Çalıştırılması



DİKKAT



Örnek uygulama 7.2'de yer alan ekleme komutunun bazı alan değerlerinin boş olarak kaydedilmesi NULL ifadesi ile sağlanır.

[Çalışanlar] tablosuna [YeniPersonel] tablosundaki kayıtların eklenmesini sağlayan sorgu yazınız.

ÖRNEK 7.3

INSERT INTO ekleme sorgusunun bir seçme sorgusu ile çalışmasında dikkat edilmesi gereken konu alan sayılarının eşleşmesi ve alan içerisindeki veri türlerinin uyum göstermesidir. Bu uyum alan türünün aynı olmasında bile sağlanamayabilir. Örneğin metin türündeki iki alan, uzunlukları nedeniyle hataya sebep olabilmektedir. Select komutu aşağıdaki basit bir formda tek bir tablodan sağlanabileceği oldukça karmaşık yapıda oluşturulmuş seçme sorgularından sağlanabilmektedir.

```
INSERT INTO Çalışanlar (Şirket, Soyadı, Ad, [E-posta Adresi],  
[İş Unvanı], [İş Telefonu], [Ev Telefonu], [Cep Telefonu],  
[Faks Numarası], Adres, Şehir, [Eyalet/İl], [Posta Kodu],  
[Ülke/Bölge], [Web Sayfası], Notlar, Ekler)
```

```
SELECT Şirket, Soyadı, Ad, [E-posta Adresi], [İş Unvanı],  
[İş Telefonu], [Ev Telefonu], [Cep Telefonu], [Faks Numarası],  
Adres, Şehir, [Eyalet/İl], [Posta Kodu], [Ülke/Bölge],  
[Web Sayfası], Notlar, Ekler  
FROM YeniPersonel
```

(2 row(s) affected)

Şekil 7.9

[Çalışanlar] Tablosuna [YeniPersonel] Tablosundaki Verinin Eklenmesi Sonrası Oluşan Tablo İceriği

The screenshot shows the SQL Query window with two queries: 'Select * from YeniPersonel' and 'Select * from Çalışanlar'. The results grid displays 11 rows of data, combining the data from both tables. The columns are: No, Şirket, Soyadı, Ad, E-posta Adresi, İş Unvanı, İş Telefonu, Ev Telefonu, Cep Telefonu, Faks Numarası. The data includes entries like Northwind Traders, Göktepe, Safiye, safiye@northwindtraders.com, Satış Temsilcisi, (123) 555 01 00, (123) 555 01 02, NULL, (123) 555 01 00, and so on for other employees.

No	Şirket	Soyadı	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev Telefonu	Cep Telefonu	Faks Numarası
1	Northwind Traders	Göktepe	Safiye	safiye@northwindtraders.com	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555
2	Northwind Traders	Biber	Atilla	atilla@northwindtraders.com	Genel Müdür ...	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555
3	Northwind Traders	Kocak	Haluk	haluk@northwindtraders.com	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555
4	Northwind Traders	Salah	Meryem	meryem@northwindtraders.com	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555
5	Northwind Traders	Tank	Serap	serap@northwindtraders.com	Satış Müdürü	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555
6	Northwind Traders	Nefer	Mehmet	mehmet@northwindtraders.com	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555
7	Northwind Traders	Zare	Ali	ali@northwindtraders.com	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555
8	Northwind Traders	Güzel	Lale	lale@northwindtraders.com	Satış Koordinatör...	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555
9	Northwind Traders	Türkoglu	Aliye	aliye@northwindtraders.com	Satış Temsilcisi	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555
10	Northwind Traders	Adar	Nihat	nadar@northwindtraders.com	Satış Koordinatör...	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555
11	Northwind Traders	Sönmez	Harun	hsonmez@northwindtraders.c...	Satış Koordinatör...	(123) 555 01 00	(123) 555 01 02	NULL	(123) 555

Query executed successfully. | localhost (12.0 SP1) | OFISMACPRO2\Sinan (54) | Northwind | 00:00:00 | 11 rows

Şekil 7.9'daki [*Çalışanlar*] tablosu incelendiğinde 10 ve 11. satırda yeni verinin eklendiği görülmektedir. Ancak ekleme komutunda [*No*] alanı olmadığı hâlde no alanlarında 12 ve 13 değerlerinin olduğu görülecektir. Çalışanlar tablosunun [*No*] alanını **IDENTITY** olarak tanımlandığından bu alana veri girilmesine izin verilmez zira bu tanımlama bu alana otomatik olarak bir sayıça vasıtası ile değer verilmesini sağlamaktadır. Bu nedenle bu alana değer eklenmek istediği hata iletisi alınacaktır.

Tabloya Satır Ekleme İşlemlerinde Karşılaşılan Hatalar

Yazım Hataları: SQL sorgu ifadelerinin yazımında hatalar kullanıcıya sorgu ekranındaki konumla bildirilir. Örneğin Şekil 7.8'de çalıştırılan sorgudaki bir virgülün unutulması durumunda kullanıcıya

“**Msg 102, Level 15, State 1, Line 2 Incorrect syntax near 'Müşteri No'**”

hatası ilettilir. Ayrıca eklenen verideki alan sayısı ile hedef tablodaki alan sayısında farklılık varsa bu hata da işlemin yapılmasını engelleyen bir hatadır ve kullanıcıya

“**There are more columns in the INSERT statement than values specified in the VALUES clause. The number of values in the VALUES clause must match the number of columns specified in the INSERT statement.**”

mesajı ile hata hakkında nerede fazla sütun alan olduğu belirtilerek düzeltilemesine yönelik bilgi sağlanır.

Veri Tipi Uyum Hataları: Tablo eklemeye yaygın olarak karşılaşılan hatalardan biri de eklenen verinin türü ile eklenmek istenen tablodaki alanın türü arasındaki uyuşmazlıktır. Metin türündeki bir alana sayısal veri eklenmesi durumunda veritabanı yönetim sistemi yazılımları bu hatayı tolere edebilmektedir ancak sayısal veri türündeki bir alana metin girilmesi hata üretilmesine neden olur. Örneğin metin sayı türündeki bir alana metin verisi eklenmek istenildiğine

“**Msg 245, Level 16, State 1, Line 1
Conversion failed when converting the varchar value 'rtrt3' to data type int.**”

hatası ürettilir. Bu hata mesajında SQL veritabanı motorunun metin değerini sayıya çevirmeye çalışıldığı ancak hata ile sonuçlandığı bildirilmektedir. Veritabanı yönetim yazılımları bu tür hataları gidermek için çeşitli yöntemler kullanırlar. Metin türündeki verinin sayıya dönüştürülmesi ve ondalıklı sayıların tam sayıya dönüştürülmesi, hataları en azı indirmek için uygulanabilmektedir. Tarih türündeki verileri veritabanı tablolara eklemeye farklı dillerde tarih yazılımlarının farklı olması sorun yaratılmaktadır. Bu sorunu çözmek için tarih verisini dönüştürme ile ilgili kurallar kullanılabilir. Ancak el ile tarih bilgisi girmede ‘Yıl/Ay/Gün’ biçiminde verinin yazılması hatayı engelleyemektedir. Tarih bilgisi hatalı girildiğinde aşağıdakine benzer bir hata mesajı alınabilir.

“**Msg 242, Level 16, State 3, Line 1
The conversion of a varchar data type to a datetime data type resulted in an out-of-range value. The statement has been terminated.**”

Tablo Alanları Kısıtlayıcıları: Tablolara veri ekleme SQL komutu yazarken yazım hatası ya da veri tipi uyumsuzluğunun dışında tablo alanlarında tanımlanmış kısıtlayıcılar da hatalara neden olabilmektedir. Örneğin bir alanın boş bırakılamayacağına ilişkin bir tasarım yapıldıysa veri ekleme esnasında o alanın boş bırakılamayacağı konusunda kullanıcuya hata üretilir.

`“Cannot insert the value NULL into column ‘a’, table ‘Northwind.dbo.Calisanlar’; column does not allow nulls. INSERT fails”`

Ayrıca **CHECK** ifadesi ile oluşturulmuş alanların alması gereken değerlerin sınırları dışında bir işlem de hataya neden olabilir. Aşağıda [*Calışanlar*] adlı tablonun [*Ucret*] alanına eklenmek istenen bir değerin kurala uymadığı hatasını veren bir örnek yer almaktadır.

`“The INSERT statement conflicted with the CHECK constraint “CHK_Ucret”. The conflict occurred in database “Northwind2”, table “dbo.Calisanlar”, column ‘Ucret’”`

Tablolar arasında tanımlanan yabancı anahtar ilişkileri de bir alana girilecek değerin farklı bir tabloda olmasına zorlayabilir. Bu durumda da kullanıcıya hata benzer bir iletiyle bildirilir.

Bu ve benzer hatalar, **VALUES** komutu ile gerçekleştirilen veri ekleme işlemlerinde kolayca fark edilerek çözülebilir. Ancak binlerce satır verinin başka tablolardan elde edilecek ekleme işlemlerinde bu hatalar hayli can sıkıcı olabilmektedir. Veritabanı ve tabloların yapılarını bilmek, kaynak verilerin türü ve içeriği konusunda bilgi sahibi olmak bu tür hataların çözülmesinde oldukça önemli olmaktadır.

Veri Girişinde Yaygın Kullanılan İşlevler

Verilerin tablolara eklenmesi için eklenecek verilerin düzenlenmesi gerekmektedir. Bu gibi durumlarda SQL server dilindeki işlevler kullanılabilimekte ve verinin istenilen forma getirilmesi sağlanabilmektedir. Programlama dillerinde sıklıkla kullanılan ve belli işleri yapmak üzere tasarlanmış kod blokları işlevler olarak isimlendirilmektedir. İşlevlerin en büyük avantajı tekrarlanan işleri yapmak üzere gereksiz kod tekrarını önlemektir. Bu kapsamda veritabanına veri girerken SQL dilinin getirmiş olduğu fonksiyonları kullanabilmek de mümkündür. SQL fonksiyonları da diğer programlama dillerindeki aynı mantıkla üretilen program parçalarıdır. Fonksiyonlar temel olarak gerekli parametreleri alıp (bazı fonksiyonlar parametre almadan da kullanılacak şekilde tasarlanabilir), aldığı parametreleri kullanarak çalışır ve bulunan sonuçları geri döndürürler. SQL dilinde tanımlı birçok kategori altında çok sayıda işlev mevcuttur. Bu işlevler ekleme de kullanılabilecek en önemlilerine Tablo 7.1'de yer verilmiştir.

Tablo 7.1
Veri Ekleme ve
Düzenleme İşlevlerinde
Kullanılabilecek İşlevler

İşlev	Açıklaması	Kullanım Şekli
CAST	Sorgu esnasında veri türünün dönüştürülmesini sağlar. (Veritabanı yapısını değiştirmez)	<code>Cast ([Tutar] as nvarchar (20))+ 'TL dir.'</code> Sonuç: 20.00TL dir.
CONVERT	Cast ile aynı işlevi gerçekleştirir.	<code>Convert (nvarchar (20), [Tutar]) + 'TL dir.'</code>
PARSE	Bir ifadedeki bilgili ilgili dile göre metin içerisindeń çıkarır.	<code>Parse('19 Haziran 1982' AS datetime USING 'Tr-TR')</code> Sonuç: 1982-06-19 00:00:00 dir.
LOWER	Metinleri küçük harfe çevirir.	<code>Lower ('KÜÇÜK HARF')</code> Sonuç: küçük harf
UPPER	Metinleri büyük harfe çevirir.	<code>Upper('büyük harf')</code> Sonuç: BÜYÜK HARF
ROUND	Ondalıklı bir sayının istenen basamağa kadar yuvarlanmasıyı sağlar	<code>ROUND(123.4545, 2);</code> Sonuç: 123.4500
GETDATE	SQL Server yazılımının çalıştığı bilgisayarın sistem tarihini döndürür.	<code>GetDate()</code> Sonuç: 2017-06-01 11:00:12.98
DAY, MONTH, YEAR	Tarih veri türündeki değişkenlerin sırasıyla gün, ay ve yıl verisine dönüştüren işlevlerdir.	<code>Day('2018/5/17')</code> Sonuç: 17 <code>Month(FaturaTarihi)</code> Sonuç: 05 <code>Year(GetDate())</code> Sonuç: 2017
LTRIM, RTRIM	Karakter türündeki değişkenlerin başındaki (LTRIM) ya da sonundaki (RTRIM) boşluk karakterlerini kaldırır.	<code>LTRIM(' TÜRKİYE')</code> Sonuç: 'TÜRKİYE'

INTERNET



SQL Sever işlevlerine <https://msdn.microsoft.com/en-us/library/ms174318.aspx> internet adresinden ulaşabilirsiniz.

SIRA SİZDE



Bir metin verinin sonundaki gereksiz boşlukların kaldırılması için hangi işlev kullanılabilir araştırınız. 'Veritabanları Sistemleri' metninin sonundaki boşlukları kaldırarak **[Kitaplar]** tablosunun **[Kitapadi]** alanına ekleyen komutu yazınız.

NULL Değerler

Herhangi bir kayıt satırında ilgili sütuna hiç bir veri girilmez ise alana **NULL** değer atanır. Örnek veritabanında **[Müşteriler]** tablosunda müşteri numarası **[No]**, müşterinin şirket ismi **[Şirket]**, Adres **[Adres]**, Şehir **[Şehir]**, ev telefonu **[Ev Telefonu]** ve faks **[Faks Numarası]** gibi bilgiler tutulmaktadır. Bu bilgilerden bazılarının doldurulması zorunlu olmasına rağmen bazı alanları ise opsionel olarak istenirse boş bırakılabilir. **[Ev Telefonu]** alanı bu tür opsionel olarak doldurulmak üzere tasarlanmış bir alandır. Müşteri bilgileri alınırken müşterinin telefonu olmadığı veya müşteri telefon numarasını vermek istemediği durumlarda bu alanın boş bırakılması gerekmektedir. Oluşturulan bir tablodaki yukarıdaki örnekte olduğu gibi bir veya birden fazla alana değer girilmemesi veya başka bir ifadeyle **NULL** değer girilmesi gerekebilir. Bunun için kullanılabilen iki farklı yöntem mevcuttur. Bu yöntemlerden ilki aşağıda da gösterildiği gibi **INSERT INTO** komutu içerisinde

NULL değer alacak alanların yazılmasına karşıdır. Aşağıdaki örnekte, [*İş Telefonu*] alanından sonraki alanlara ait değerler yazılmadığından dolayı aşağıdaki şekilde de görüldüğü üzere bu alanların değerleri **NULL** değeri almıştır.

[Müşteriler] tablosuna yeni bir müşterinin eklenmesini sağlayan sorgu yazınız.

ÖRNEK 7.4

```
INSERT INTO [dbo].[Müşteriler]
([Şirket],[Soyadı],[Ad],[E-postaAdresi],
[İş Unvanı],[İş Telefonu])
VALUES
('Şirket DD','CAN','Ali','ali@can.com',
'Satış Müdürü','(123) 555 01 00')
```

Sekil 7.10

[Müşteriler] Tablosundaki Bazı Alanlara Veri Girilmediği Durumundaki Verinin Eklenmesi Sonrası Oluşan Tablo İçeriği

No	Şirket	Soyadı	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev Telefonu	Cep Telefonu	Faks Numarası	Adres	Şehir	Eyalet/İl	Posta Kodu
1	Şirket DD	CAN	Ali	ali@can.com	Satış Müdürü	(123) 555 01 00	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Uygulanabilecek diğer yöntem ise boş bırakılmak istenilen alanlara **NULL** yazılmasıdır ve bir örneği aşağıda verilmiştir.

```
INSERT INTO [dbo].[Müşteriler]
([Şirket],[Soyadı],[Ad],[E-posta Adresi],[İş
Unvanı],[İş Telefonu],
[Ev Telefonu],[Cep Telefonu],[Faks Numar
ası],[Adres],[Şehir],[Eyalet/İl],[Posta
Kodu],[Ülke/Bölge],
[Web Sayfası],[Notlar],[Ekler],[tamp])
VALUES
('Şirket DD','CAN','Ali','ali@can.com','Satış Müdü
rı','(123) 555 01 00',
NULL,NULL,NULL,'6 Evler Mah. 25.Sok No
40',UPPER('eskisehir'),
NULL,'26040','TÜRKİYE',NULL, NULL,NULL,)
```

Yukarıda verilen örnekte görüldüğü üzere bir müşterinin sadece [*Şirket*], [*Soyadı*], [*Ad*], [*E-posta Adresi*], [*İş Ünvanı*], [*İş Telefonu*], [*Adres*], [*Şehir*], [*Posta Kodu*] ve [*Ülke/Bölge*] bilgileri girilmiş geri kalan diğer tüm bilgileri (*Ev Telefonu*), (*Cep Telefonu*), (*Faks Numarası*), (*Eyalet/İl*), (*Web Sayfası*), (*Notlar*), (*Ekler*) alanları boş olarak bırakılmış yani **NULL** değerler girilmiştir. Ancak veritabanına **NULL** değerlerinin girilmesi ve **NULL** değere sahip olan bu veriler üzerinde çalışırken bir takım veri tutarsızlıkları ile karşılaşılabilir. **NULL** değerlerin saklandığı alanlarda yapılan sorgulamalarda bazı hatalar oluşabileceği gibi ortaya çıkan sonuçlar da yaniltıcı olabilir. Örneğin, tüm müşterilerimizin bilgilerini aşağıdaki sorgu ile elde edebilmemiz mümkündür.

```
SELECT * FROM [dbo].Müşteriler
```

Şekil 7.11*[Müşteriler] Tablosundaki Tüm Kayıtlar*

No	Sirket	Soyadi	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev ...	Cep T...	Faks Numarası	Adres	Şehir	Eyalet/İl	Pc
1	Şirket A	Berber	Aliye	NULL	Sahibi	(123) 555 01 00	NU...	NULL	(123) 555 01 01	1. Sokak No: 123	ESKİSEHIR	SIN	9%
2	Şirket B	Giresunlu	Temel	NULL	Sahibi	(123) 555 01 00	NU...	NULL	(123) 555 01 01	2. Sokak No: 123	ESKİSEHIR	BUR	9%
3	Şirket C	Aksak	Timur	NULL	Satınalma Temsilcisi	(123) 555 01 00	NU...	NULL	(123) 555 01 01	3. Sokak No: 123	Antalya	ANT	9%
4	Şirket D	Leylak	Çağdem	NULL	Satınalma Müdürü	(123) 555 01 00	NU...	NULL	(123) 555 01 01	4. Sokak No: 123	İstanbul	IST	9%
5	Şirket E	Türkoğlu	Metin	NULL	Sahibi	(123) 555 01 00	NU...	NULL	(123) 555 01 01	5. Sokak No: 123	ESKİSEHIR	NEV	9%
6	Şirket F	Olçay	Feridun	NULL	Satınalma Müdürü	(123) 555 01 00	NU...	NULL	(123) 555 01 01	6. Sokak No: 123	ESKİSEHIR	MUŞ	9%
7	Şirket G	Çırılı	Mine	NULL	Sahibi	(123) 555 01 00	NU...	NULL	(123) 555 01 01	7. Sokak No: 123	Bilecik	BIL	9%

Query executed successfully. | USER\SQLEXPRESS (12.0 RTM) | user\user (56) | Northwind | 00:00:00 | 30 rows

Sorgu sonucu yukarıdaki şekilde gösterilmiş olup toplamda 30 adet kayıt sorgu sonucu olarak görülmektedir. Bu müşterilerden ESKİSEHIR ilindeki müşterilerin listesi ve ne kadar müşteri olduğu öğrenilmek istenirse aşağıdaki sorgunun yapılması yeterli olacaktır.

```
SELECT * FROM Müşteriler WHERE Şehir='ESKİSEHIR'
```

Şekil 7.12*[Müşteriler] Tablosundaki ESKİSEHİR İlinden Olan Tüm Kayıtlar*

No	Sirket	Soyadi	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev Tel...	Cep Tel...	Faks Numarası	Adres	Şehir	Eyalet
1	Şirket A	Berber	Aliye	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	1. Sokak No: 123	ESKİSEHIR	SIN
2	Şirket B	Giresunlu	Temel	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	2. Sokak No: 123	ESKİSEHIR	BUR
3	Şirket C	Aksak	Timur	NULL	Satınalma Temsilcisi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	3. Sokak No: 123	Antalya	ANT
4	Şirket D	Leylak	Çağdem	NULL	Satınalma Müdürü	(123) 555 01 00	NULL	NULL	(123) 555 01 01	4. Sokak No: 123	İstanbul	IST
5	Şirket E	Türkoğlu	Metin	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	5. Sokak No: 123	Nevşehir	NEV

Query executed successfully. | USER\SQLEXPRESS (12.0 RTM) | user\user (56) | Northwind | 00:00:00 | 5 rows

Sorgu sonucunda elde edilen müşteriler yukarıdaki şekilde gösterilmiş olup toplamda 5 adet müşteri Eskişehir ilindendir. Eskişehir dışındaki müşterilerin bulmak için ise aşağıdaki sorgu çalıştırılmalıdır. Sorgu sonucu aşağıdaki şekilde verilmiştir.

```
SELECT * FROM Müşteriler WHERE Şehir !='ESKİSEHIR'
```

Şekil 7.13*[Müşteriler] Tablosundaki ESKİSEHİR Dışından Olan Tüm Kayıtlar*

No	Sirket	Soyadi	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev Telefonu	Cep Telefonu	Faks Numarası	Adres	Şehir	Eyalet
1	Şirket A	Berber	Aliye	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	1. Sokak No: 123	Sinop	SIN
2	Şirket B	Giresunlu	Temel	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	2. Sokak No: 123	Bursa	BUR
3	Şirket C	Aksak	Timur	NULL	Satınalma Temsilcisi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	3. Sokak No: 123	Antalya	ANT
4	Şirket D	Leylak	Çağdem	NULL	Satınalma Müdürü	(123) 555 01 00	NULL	NULL	(123) 555 01 01	4. Sokak No: 123	İstanbul	IST
5	Şirket E	Türkoğlu	Metin	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	5. Sokak No: 123	Nevşehir	NEV
6	Şirket F	Olçay	Feridun	NULL	Satınalma Müdürü	(123) 555 01 00	NULL	NULL	(123) 555 01 01	6. Sokak No: 123	Muş	MUŞ
7	Şirket G	Çırılı	Mine	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	7. Sokak No: 123	Bilecik	BIL

Yukarıdaki sekilden de görüldüğü üzere müşterilerin 20 adeti de Eskişehir dışındandır. Yukarıda verilen üç sorgunun sonucunda toplamda 30 adet müşteri bulunmakta, bunların 5 adedi Eskişehir ilinden olup 20 adedi ise Eskişehir dışındandır. Eskişehir içindeki ve Eskişehir dışındaki müşterilerin toplamı 25 olup toplam müşteri sayısından (30) farklıdır. Bunun sebebi [Şehir] sütunununda bazı verilerin NULL değere sahip olmasıdır. Eğer

tabloda **NULL** değer alabilecek olarak tanımlanan bir alanın içeriği listelenmek istenilir ise yapılan sorguda **NULL** değerlerin dikkate alınması gerekmektedir. **NULL** değerleri de içerecek şekildeki sorgulama örneği aşağıda verilmiştir.

```
SELECT * FROM Müşteriler
WHERE Şehir !='ESKİŞEHİR' or Şehir IS NULL
```

Şekil 7.14

[Müşteriler] Tablosundaki ESKİŞEHİR Dışından ve Değeri **NULL** Olan Tüm Kayıtlar

No	Şirket	Soyadı	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev Telefonu	Cep Telefonu	Faks Numarası	Adres	Şehir
1	Şirket C	Aksak	Timur	NULL	Satınalma Temsilcisi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	3. Sokak No: 123	Antalya
2	Şirket D	Leylak	Çağdem	NULL	Satınalma Müdürü	(123) 555 01 00	NULL	NULL	(123) 555 01 01	4. Sokak No: 123	İstanbul
3	Şirket G	Çınli	Mine	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	7. Sokak No: 123	Bilecik
4	Şirket H	Ankaralı	Dilek	NULL	Satınalma Temsilcisi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	8. Sokak No: 123	Ordu
5	Şirket I	Mor	Serdar	NULL	Satınalma Müdürü	(123) 555 01 00	NULL	NULL	(123) 555 01 01	9. Sokak No: 123	Sakarya
6	Şirket J	Vize	Remzi	NULL	Satınalma Müdürü	(123) 555 01 00	NULL	NULL	(123) 555 01 01	10. Sokak No: 123	Çanakkale

Sorgulama sonucu yukarıda verilmiştir. Sorgulama sonucunda 25 adet sorgu sonucu elde edilmiştir. Eskişehir içindeki ve Eskişehir dışındaki müşterilerin toplam sayısının tüm müşteri kayıtlarının getirildiği sorgulama sonucuya eşit olduğu görülmektedir. Eğer şehir bilgisi girilmiş tüm müşterilerin listesi alınmak istenirse ise aşağıdaki sorgu yapılır.

```
SELECT * FROM Müşteriler WHERE Şehir IS NOT NULL
```

Şekil 7.15

[Müşteriler] Tablosundaki Şehir İsimleri Olan Tüm Kayıtlar

No	Şirket	Soyadı	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev Te...	Cep ...	Faks Numarası	Adres	Şehir	Eyale
1	Şirket A	Berber	Aliye	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	1. Sokak No: 123	ESKİŞEHİR	SİN
2	Şirket B	Giresunlu	Temel	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	2. Sokak No: 123	ESKİŞEHİR	BUR
3	Şirket C	Aksak	Timur	NULL	Satınalma Temsilcisi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	3. Sokak No: 123	Antalya	ANT
4	Şirket D	Leylak	Çağdem	NULL	Satınalma Müdürü	(123) 555 01 00	NULL	NULL	(123) 555 01 01	4. Sokak No: 123	İstanbul	İST
5	Şirket E	Türkoğlu	Metin	NULL	Sahibi	(123) 555 01 00	NULL	NULL	(123) 555 01 01	5. Sokak No: 123	ESKİŞEHİR	NEV
6	Şirket F	Olcay	Feridun	NULL	Satınalma Müdürü	(123) 555 01 00	NULL	NULL	(123) 555 01 01	6. Sokak No: 123	ESKİŞEHİR	MUŞ

Müşterilerimizin büyük çoğunluğu Türkiye'den ise her müşteri için [Müşteriler] tablosundaki [*Ülke/Bölge*] alanına ülke bilgisinin girilmediği durumlarda otomatik olarak Türkiye girilmesi sağlanabilir. Bu işlem varsayılan değer (**DEFAULT**) olarak isimlendirilir. Kullanıcı kayıt girişü sırasında varsayılan değer tanımlanmış alana herhangi bir veri girmezse, SQL Server tarafından otomatik olarak o alana ait varsayılan değer atanır. Varsayılan değerler tablo oluşturulurken tanımlanır.

TABLODAN KAYIT SİLME

DELETE komutu kullanarak bir tabloda bulunan kayıt veya kayıtların istenildiği zaman silinmesi mümkündür. Genel yapısı aşağıdaki gibidir.

DELETE komutu kullanırken dikkatli olunması gerekmektedir.



DİKKAT

```
DELETE FROM TabloAdı WHERE Koşul
```

Yukarıdaki ifadede görüldüğü gibi **DELETE** komutu kullanılırken **FROM** ekinden sonra yazılan **TabloAdı**'ndaki kayıtları silme işlemi gerçekleştirmek için kullanılır. **WHERE** ifadesi ise kendisinden sonra yazılan **Koşul** ifadesi ile tablodaki hangi kayıtların silinmesi gerektiğini belirlemek için kullanılır. Eğer **WHERE** ifadesi yazılmaz ise bir koşul belirlenmez, böylece tablodaki bütün değerler silinir. Bu yüzden **WHERE** ifadesinden sonra koşul kullanmaya dikkat edilmesi gerekmektedir.

ÖRNEK 7.5

[Müşteriler] tablosunda müşteri numarası 29 olan kayıtların silinmesini sağlayan sorgu yazınız.

[No]'su '29' olan müşterinin [Müşteriler] tablosundaki kayıtları aşağıdaki sorguya elde edilmiş ve elde edilen sorgu sonucu aşağıdaki şekilde gösterilmiştir.

```
SELECT * FROM Müşteriler WHERE No=29
```

Şekil 7.16

[Müşteriler] Tablosundaki Müşteri Numarası 29 Olan Kayıt

No	Şirket	Soyadı	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev Telefonu	Cep Telefonu	Faks Numarası	Adres	Şehir
1	29 Şirket CC	Leylak	Mehmet Ali	NULL	Satınalma Müdürü	(123) 555 01 00	NULL	NULL	(123) 555 01 01	29. Sokak No: 789	Denizli

Bahsedilen müşterinin kayıtlarının [Müşteriler] tablosundan kaydı silinmesi istenir ise aşağıdaki komut yazılar ve SQL komut sonucu aşağıdaki şekilde gösterilmiştir. Müşterinin kaydının silindiği şekilde gözükmektedir.

DİKKAT



DELETE komutu ile silmeden önce kullanacağınız **SELECT** komutu ile silinecek verileri inceleyebilirsiniz. Silmeyi tamamlamak için **SELECT** komutundaki **WHERE** ifadesinden sonraki kısmı **DELETE** komutunda kullanabilirsiniz.

```
DELETE Müşteriler WHERE No=29
```

Şekil 7.17

[Müşteriler] Tablosundaki [No] Alanı 29 Olan Kaydın Silinmesi

No	Şirket	Soyadı	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev Telefonu	Cep Telefonu	Faks Numarası	Adres	Şehir	Eyalet/İl	Posta Kodu	Ülke/Bölge

Silme Komutu

DELETE Müşteriler

WHERE ifadesi olmadan yazılp çalıştırılırsa [Müşteriler] tablosundaki bütün kayıtlar silinecektir. Bir tablodaki tüm kayıtları silmek için yukarıda verilen komuttaki gibi **WHERE** ifadesi kullanmadan **DELETE** ya da **TRUNCATE** komutu kullanılır. İşlem günlüğüne yazıldığından dolayı **TRUNCATE** komutu ile yapılan silme işlemi geri alınamaz. **TRUNCATE** komutunun kullanımı aşağıdaki gibidir.

```
TRUNCATE TABLE TabloAdı
```

Komutun çalıştırılması için sadece işlemin yapılacak tablonun adı gereklidir. **TRUNCATE** komutu, **DELETE** komutuna göre daha hızlı çalışır ve daha az sistem kaynağı kullanıldığından dolayı tercih edilir. Aşağıdaki gibi yazılacak komut ile [Müşteriler] tablosundaki tüm kayıtların silinmesi sağlanır.

TRUNCATE TABLE Müşteriler

ÖRNEK 7.6

[Siparişler] tablosundaki [Sipariş Tarihi] 1 Nisan 2006 tarihinden sonra yapılan kayıtların silinmesini sağlayan sorgu yazınız.

Aşağıdaki komut, [Siparişler] tablosundaki satışlardan [Sipariş Tarihi] 1 Nisan 2006 tarihinden sonra yapılan siparişleri getirmekte, getirilen değerler ise aşağıdaki şekilde görülmektedir.

```
SELECT * FROM Siparişler WHERE [Sipariş Tarihi]>'2006-04-01'
```

Şekil 7.18

[Siparişler] tablosundaki 1 Nisan 2006 Tarihinden Sonra Yapılan Siparişlerin Kayıtları										
Sipariş No	Çalışan No	Müşteri No	Sipariş Tarihi	Sevk Tarihi	Taşıyıcı No	Sevk Adı	Sevk Adresi	Sevk Şehir	Sevk Eyalet/İl	
1 45	1	28	2006-04-07 00:00:00.000	2006-04-07 00:00:00.000	3	Amire Yalnız	28. Sokak No: 789	Manisa	MAN	
2 46	7	9	2006-04-05 00:00:00.000	2006-04-05 00:00:00.000	1	Serdar Mor	9. Sokak No: 123	Sakarya	SAK	
3 47	6	6	2006-04-08 00:00:00.000	2006-04-08 00:00:00.000	2	Feridun Olcay	6. Sokak No: 123	Muş	MUŞ	
4 48	4	8	2006-04-05 00:00:00.000	2006-04-05 00:00:00.000	2	Dilek Ankaralı	8. Sokak No: 123	Ordu	ORD	
5 50	9	25	2006-04-05 00:00:00.000	2006-04-05 00:00:00.000	1	Can Yolcu	25. Sokak No: 789	Çanakkale	ÇAN	
6 51	9	26	2006-04-05 00:00:00.000	2006-04-05 00:00:00.000	3	Sunay Beyaz	26. Sokak No: 789	Izmir	İZM	

Eğer bu satışlar silinmek isteniyor ise aşağıdaki komutun çalıştırılması yeterlidir. 1 Nisan 2006 tarihinden sonra yapılan satışlar tekrar sorgulandığında aşağıdaki şekilde gösterilen sorgu sonucu elde edilir. Şekilden de görüleceği gibi kayıtların silinme işlemi başarılı olarak gerçekleşmiştir.

```
DELETE Siparişler WHERE [Sipariş Tarihi]>'2006-04-01'
```

Şekil 7.19

[Siparişler] Tablosundaki 1 Nisan 2006 Tarihinden Sonra Yapılan Siparişlerin Silinmesi											
SatisID	MusteriID	PersonelID	SatisTarihi	OdemeTarihi	SevkTarihi	ShipVia	NakliyeUcreti	SevkAdi	SevkAdresi	SevkSehri	Sevk

[Siparişler] tablosundaki kayıtlardan 1 Mart 2006 ile 1 Ocak 2006 arasındaki siparişleri içeren kayıtları silmek için gerekli SQL komutunu yazınız.



SIRA SİZDE

Birden Fazla Tablo ile Eylem Sorguları

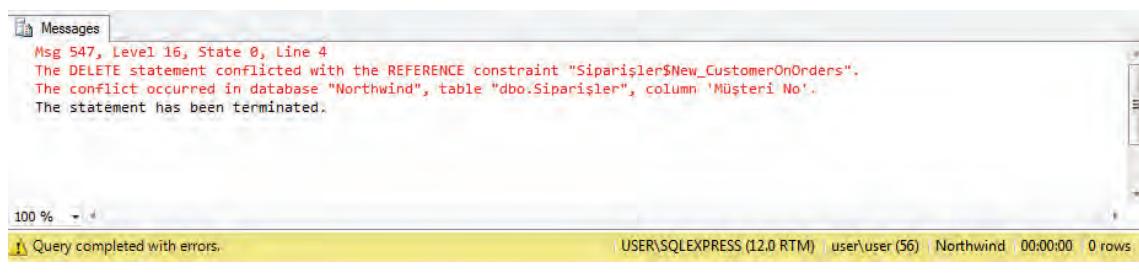
Aşağıdaki komut yardımıyla [Müşteriler] tablosundaki müşteri numarası 28 olan müşterinin kaydının silinmesi istenmektedir. Bunun için yazılan komut aşağıda verilmiştir.

```
DELETE FROM Müşteriler WHERE [No]=28
```

Ancak yukarıdaki komut çalıştırıldığı zaman aşağıdaki şekilde verilen hata alınmakta ve kayıt silinememektedir.

Şekil 7.20

[Müşteriler] Tablosundaki Müşteri Numarası 28 Olan Müşterinin Kaydının Silinmesi Sırasında Alınan Hata



Bu hatanın alınma sebebi silinmek istenen müşterinin [Siparişler] tablosunda kayıtlarının olmasıdır. Aşağıdaki verilen SQL komutu ile silinmek istenen müşterinin [Siparişler] tablosundaki kayıtları bulunabilir. Müşterinin [Siparişler] tablosunda 4 adet kaydı bulunmaktadır, bu kayıtlar aşağıdaki şekilde gösterilmektedir.

```
SELECT * FROM Siparişler WHERE [Müşteri No]=28
```

Şekil 7.21

[Siparişler] Tablosundaki Müşteri Numarası 28 Olan Müşterinin Kayıtları

The screenshot shows the SQL Server Management Studio interface with the 'Results' tab selected. The results grid displays four rows of data from the Siparişler table, corresponding to the four orders placed by customer ID 28. The columns shown are Sipariş No, Çalışan No, Müşteri No, Sipariş Tarihi, Sevk Tarihi, Taşıyıcı No, Sevk Adı, Sevk Adresi, Sevk Şehir, Sevk Eyalet/İl, and Sevk Posta Kodu. The bottom status bar indicates 'Query executed successfully.' and '100 %' completion. The bottom status bar shows connection details: 'USER\SQLEXPRESS (12.0 RTM) | user\user (57) | Northwind | 00:00:00 | 4 rows'.

Sipariş No	Çalışan No	Müşteri No	Sipariş Tarihi	Sevk Tarihi	Taşıyıcı No	Sevk Adı	Sevk Adresi	Sevk Şehir	Sevk Eyalet/İl	Sevk Posta Kodu
1	38	9	2006-03-10 00:00:00.000	2006-03-11 00:00:00.000	3	Amire Yalnız	28. Sokak No: 789	Manisa	MAN	99
2	45	1	2006-04-07 00:00:00.000	2006-04-07 00:00:00.000	3	Amire Yalnız	28. Sokak No: 789	Manisa	MAN	99
3	65	9	2006-05-11 00:00:00.000	2006-05-11 00:00:00.000	3	Amire Yalnız	28. Sokak No: 789	Manisa	MAN	99
4	72	1	2006-06-07 00:00:00.000	2006-06-07 00:00:00.000	3	Amire Yalnız	28. Sokak No: 789	Manisa	MAN	99

Silme işlemi yapılrken tablolar arasında bulunan ilişkilerin bozulmamasına dikkat edilmesi gerekmektedir. [Müşteriler] tablosundaki müşteri silinme işlemi gerçekleşmiş olsaydı (bu müşteriye ait siparişler hâlâ [Siparişler] tablosunda bulunmaktadır) iki tablo arasındaki kayıtlar arasında tutarsızlık ortaya çıkacaktır. Bundan dolayı veritabanı oluşturulurken bu iki tablo birbirine bağlanmış, tablolar arasında ortaya çıkabilecek tutarsızlıkların önüne geçilmiştir. Bu müşterinin [Müşteriler] tablosundaki kaydının silinmesi için öncelikle bağlı olduğu [Siparişler] tablosundaki kayıtların silinmesi gerekmektedir. [Siparişler] tablosundaki bu müşteriye ait kayıtlar aşağıdaki SQL komutu çalıştırılarak silinebilir.

```
DELETE FROM Siparişler WHERE [Müşteri No]=28
```

Yukarıda bahsedilen bu yapı kayıt ekleme ve silme işlemlerini kısıtlar gibi görünse de tablo üzerindeki veri bütünlüğünü korumaktadır. Birbiri ile bağlanmış tablolarda ana tablodaki kaydın silinmesi ile onunla ilişkili kayıtların silinmesini sağlayacak olan bir **CASCADE DELETE** (kademeli silme) yapısı SQL dilinde mevcuttur. Eğer yukarıdaki işlemlerin sıra ile veritabanı tarafından otomatik yapılması istenir ise (bir kaydın [Müşteriler] tablosundan silinmesi istenildiği zaman sırayla ona bağlı olan [Siparişler], ve diğer bağlı tablolarındaki kayıtların otomatik olarak silinmesi) tablo oluşturulurken **ON DELETE CASCADE** komutu kullanılabilir. Bu komutla temel olarak amaçlanan veri bütünlüğünü sağlamak olmasına rağmen hangi tablonun hangi tablo ile bağlantılı olduğunun takibini

yapmadan silme işlemi gerçekleştirmek mümkün olmaktadır. Kademeli silme özelliği ile bir tablodan veri silindiğinde otomatik olarak tüm bağlı tablolardaki kayıtlar da silinir, bu sayede tablolar arasındaki veri bütünlüğü de sağlanmış olur.

Eğer tablo aşağıdaki şekilde **ON DELETE CASCADE** ifadesi ile oluşturulur ise [*Müşteriler*] tablosundan silinen herhangi bir müşterinin [*Siparişler*] tablosundaki kayıtları da silinecektir. Aynı ifadenin diğer bağlı tablolar içinde tanımlanması gerekmektedir.

```
CREATE TABLE [dbo].[Siparişler](
    [Sipariş No] [int] IDENTITY(1,1) NOT NULL,
    [Çalışan No] [int] NULL,
    [Müşteri No] [int] NULL,
    [Sipariş Tarihi] [datetime] NULL,
    [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,
    [Vergiler] [money] NULL,
    [Ödeme Türü] [nvarchar](50) NULL,
    [Ödeme Tarihi] [datetime] NULL,
    [Notlar] [nvarchar](max) NULL,
    [Vergi Oranı] [float] NULL,
    [Vergi Durumu] [tinyint] NULL,
    [Durum No] [tinyint] NULL,
    [SSMA_TimeStamp] [timestamp] NOT NULL,
CONSTRAINT [PK_Siparişler] PRIMARY KEY CLUSTERED
ON DELETE CASCADE
```

Eğer tablo daha önceden oluşturulmuş ise ve o tabloya kademeli silme işlemi eklenmesi isteniyor ise aşağıdaki komutta verildiği gibi bu sefer **ALTER TABLE** işlemi ile **CASCADE DELETE** yapısı oluşturulması mümkündür.

```
ALTER TABLE [dbo].[Siparişler] WITH NOCHECK ADD CONSTRAINT [FK_Satışlar_Müşteriler] FOREIGN KEY([MüşteriID])
REFERENCES [dbo].[Müşteriler] ([MüşteriID])
ON DELETE CASCADE
```

TABLO VERİLERİNİ DÜZENLEME

Veritabanı üzerinde veri güncelleme işlemleri tipki **DELETE** işlemlerinde olduğu gibi çok dikkat gerektiren işlemlerdir ve sıkılıkla kullanılmaktadır. **UPDATE** komutu ile bir veritabanı üzerinde bir tabloda istenilen bir kayıt ve kayıtların güncelleme işlemi yapılabilir. Bununla beraber **UPDATE** komutu kullanarak bir tabloda bulunan kayıtların bir kısmı veya tamamının da herhangi bir alanındaki veya tüm alanlarındaki değerleri istenildiği zaman değiştirmek/güncellemek mümkündür. Genel yapısı aşağıdaki gibidir.

```
UPDATE TabloAdı
SET alan1 = değer1, alan2 = değer2, ...
WHERE Koşul
```

Yukarıdaki ifadede **UPDATE** komutu kendisinden sonra yazılan **TabloAdı**'ndaki verileri güncelleme işlemi gerçekleştirmek için kullanılır. **SET** ifadesi güncellenecek alanlar ve bu alanların alacakları yeni değerleri belirlemek için kullanılır. **WHERE** ifadesi ise kendisinden sonra yazılan **Koşul** ifadesi ile tablodaki hangi kayıtların güncellenmesi gerektiğini belirlemek için kullanılır. Eğer **WHERE** ifadesi yazılmaz ise bir koşul belirlenmez, böylece tablodaki **SET** ifadesinden sonra belirtilen tüm alanların değerlerinde güncelleme işlemi gerçekleşir. Ancak bu durumda tüm kayıtlar etkilenir. Bu nedenle **UPDATE** komutunda **WHERE** ifadesinden sonra koşul kullanmaya dikkat edilmesi gerekmektedir.

ÖRNEK 7.7

[Müşteriler] tablosundaki müşteri numarası 25 olan kaydın ev telefon numarasını '(222) 222 33 44' olarak değiştirmesini sağlayan sorgu yazınız.

Aşağıdaki SQL komutu çalıştırıldığı zaman müşteri numarası 25 olan müşterinin tüm bilgileri getirilmektedir. Geri dönüş değeri aşağıdaki şekilde gösterilmiştir.

```
SELECT * FROM Müşteriler WHERE No=25
```

Şekil 7.22

[Müşteriler] Tablosundaki Müşteri Numarası 25 Olan Müşterinin Kayıtları

No	Şirket	Soyadı	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev Telefonu	Cep Telefonu	Faks Numarası	Adres	Şehir	Eyalet
1	25	Şirket Y	Yolcu	Can	NULL	Satınalma Müdürü	(123) 555 01 00	NULL	NULL	(123) 555 01 01	25. Sokak No: 789	Çanakkale ÇAN

Şekilden de görüldüğü üzere müşteri numarası 25 olan müşterinin ev telefon numarası **NULL** değerdedir. Eğer bu müşterinin telefon numarasını değiştirmek / güncellemek istenirse **UPDATE** ifadesi aşağıdaki gibi yazılır. Komut çalıştırıldığı zaman aşağıdaki şekilde görüldüğü gibi telefon numarası güncellenmiş olur.

```
UPDATE Müşteriler SET [Ev Telefonu] ='(222) 222 33 44'
WHERE No=25
```

Şekil 7.23

[Müşteriler] Tablosundaki Müşteri Numarası 25 Olan Müşterinin Ev Telefonunun Güncellenmesi

No	Şirket	Soyadı	Ad	E-posta Adresi	İş Unvanı	İş Telefonu	Ev Telefonu	Cep Telefonu	Faks Numarası	Adres	Şehir	
1	25	Şirket Y	Yolcu	Can	NULL	Satınalma Müdürü	(123) 555 01 00	(222) 222 33 44	NULL	(123) 555 01 01	25. Sokak No: 789	Çanakkale

Eğer **WHERE** ifadesi ile birlikte bir koşul yazılmaz ise [Müşteriler] tablosundaki tüm müşterilerin telefon numarası aynı telefon numarası ile güncellenecektir.

```
UPDATE Müşteriler SET [Ev Telefonu] ='(222) 222 33 44'
```

Yukarıdaki komut çalıştırıldığı zaman [Müşteriler] tablosundaki bütün müşterilerin telefon numarası '(222) 222 33 44' olarak güncellenmiştir. Eğer tüm müşterilerin telefon numaralarını isteyerek '(222) 222 33 44' yapmak istenmiyor ise **UPDATE** komutu kullanırken **WHERE** ifadesinden sonra koşul yazılması unutulmamalıdır.

Veri ekleme ve silme işlemlerinde olduğu gibi güncelleme işlemlerinde de güncellenmek istenen veri Select komutu ile bir sorgu sonucu kullanarak elde edilabilir. Select komutu basit bir formda tek bir tablodan sağlanabileceği gibi aşağıda örnekte de görüldüğü gibi oldukça karmaşık yapıda oluşturulmuş seçme sorgularından sağlanabilmektedir. Aşağıdaki örnekte [Çalışanlar] tablosundaki [*ToplamSatış*] alanı [*Sipariş Ayrıntıları*] ve [*Siparişler*] tablosundaki veriler sorgulanarak güncellenmesi gerçekleşmiştir.

```

UPDATE Çalışanlar SET ToplamSatis = topsat.TS
FROM
(SELECT SUM([Sipariş Ayrıntıları].Miktar*[Sipariş Ayrıntıları].
[Birim Fiyat]) AS TS,
[Sipariş Ayrıntıları].No AS CalNo
FROM Siparişler INNER JOIN [Sipariş Ayrıntıları]
ON Siparişler.[Sipariş No] = [Sipariş Ayrıntıları].
[Sipariş No]
GROUP BY [Sipariş Ayrıntıları].No) AS topsat
INNER JOIN Çalışanlar ON topsat.CalNo = Çalışanlar.No

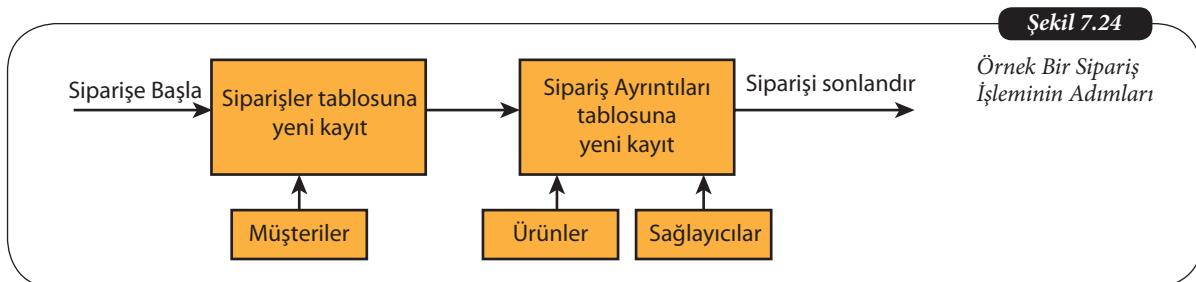
```

Birbiri ile bağlanmış tablolarda ana tablodaki kaydın güncellenmesi ile onunla ilişkili bağlı kayıtların güncellemesini sağlayacak yapı nedir?



TRANSACTION

Örnek veritabanımızda bir sipariş yapıldığı zaman ilk önce [Siparişler] tablosuna yeni bir kaydın eklenmesi, buradan elde edilecek [Sipariş No] numarası ile birlikte [Sipariş Ayrıntıları] tablosuna da yeni bir kaydın yapılması gerekmektedir. Bir ürünün siparişini olması için gereken adımlar aşağıdaki şekilde resmedilmiştir.



Ancak müşterinin istediği ürün, sağlayıcılarından temin edilememesi hâlinde veya [Sipariş Ayrıntıları] tablosuna yeni bir kayıt eklemek için yazılacak olan SQL komutunda bir hata yapılması hâlinde [Sipariş Ayrıntıları] tablosuna yeni bir kayıt eklemek mümkün olmayacağından emin olmak gerekmektedir. Ancak bu durumda da [Sipariş Ayrıntıları] tablosuna yeni bir kayıt yapılmadığı hâlde daha önce [Siparişler] tablosuna kayıt edilen kayıtlar veritabanında hâlâ dursmakta ve bunun önüne geçebilmek için tüm bu iki adımda yapılan işlemin tek bir işlem gibi yapılması gerekmektedir. Bu tür sorunları önlemek için **TRANSACTION** yapıları kullanılır.

Bir veya birden fazla SQL ifadesi arkaya arkaya tek bir işlem gibi çalıştırılmak istenildiği zaman **TRANSACTION** yapısı kullanılır. **TRANSACTION** özellikle ardı ardına gelen ve birbirine bağlı birden fazla SQL komutu tek bir SQL komutu olarak kullanılmasını sağlar. Bu sayede bir **TRANSACTION** kullanılarak yazılan SQL komutlarının ya tamamını gerçekleştirir veya hiçbirini gerçekleştirmez. **TRANSACTION** işlemleri tek bir işlem olarak ele alacağı için herhangi birisi gerçekleşmediği zaman diğer gerçekleşen işlemleri de yok sayacaktır. Yani gerçekleşen işlemi geri alacaktır(**ROLLBACK** kavramı). Eğer işlemlerin tamamı sorunsuz bir şekilde gerçekleşse, tüm işlemleri kalıcı(**COMMIT** kavramı) hâle gelecektir. **TRANSACTION** kullanımının genel yapısı aşağıdaki gibidir.

```
BEGIN TRANSACTION
    İşlem 1
    İşlem 2
    .
    .
    .
İşlem n
ROLLBACK veya COMMIT
```

Yukarıda verilen **TRANSACTION** yapısı ile iki adımda gerçekleşen işlemlerin sanki tek bir işlem gibi davranış sağlanmıştır. **TRANSACTION** yapısının avantajlarından bir tanesi ise işlemlerden herhangi birinin yapılmaması veya işlem yapılrken birinde ortaya çıkacak hatadan dolayı tüm işlemlerin de yapılmamasını sağladığıdır. **TRANSACTION** konusu ileride ayrıntılı olarak işlenecektir.

ROLLBACK ve COMMIT Kavramı

Genelde yoğun işlem yapılan ve işlemlerin tamamen bitmesinin çok önemli olduğu zamanlarda kullanılan **TRANSACTION** yapısı tüm işlemlerin tek bir işlem gibi kullanılmasına imkân vermektedir. Eğer sorgular **TRANSACTION** yapısı içinde yazılmış ise işlemlerin hepsi başarılı olana kadar tüm işlemler başarısız sayılır. **TRANSACTION** yapısını sonlandırma iki şekilde olmaktadır.

- **ROLLBACK:** Transaction işlemindeki SQL komutu tarafından yapılmış olan tüm değişikliklerin geri almak için kullanılmaktadır. Yazılan SQL komutlarının herhangi birinde hata meydana gelmesi hâlinde bir sorun karşısında **ROLLBACK** işlemi ile kayıtları ilk hâline (transaction başladığı duruma) geri getirir.
- **COMMIT:** Oluşturulan **TRANSACTION** işlemi başarılı bir şekilde gerçekleştirildiğinde veritabanında tablolar üzerinde yapılan işlemlerin (örneğin **INSERT**, **UPDATE**, **DELETE**) tablolara kalıcı olarak aktarılmasıdır.

Aşağıdaki şekilde örnek veritabanındaki [*Stok Hareket Türleri*] tablosundaki kayıtlar görülmektedir.

Sekil 7.25

[*Stok Hareket Türleri*] Tablosundaki Kayıtlar

No	Tür Adı
1	Satın Alındı
2	Satıldı
3	Tutuluyor
4	Atık

Bu tabloya iki adet yeni stok hareket türü eklenmesi istenildiği zaman aşağıdaki **TRANSACTION** yapısı ile yapmak mümkündür. **TRANSACTION** yapısının sonuna **ROLLBACK** ifadesi yazıldığından dolayı **INSERT** işlemi ile eklenen kayıtlar geri alınma işlemi yapılmıştır. Altta ki şekilde 7.26 (a)'da ekleme işleminin (**1 rows(s) affected - 1 kayıt etkilendi**) başarılı olarak gerçekleştiği ancak **ROLLBACK** ifadesinden dolayı [*Stok Hareket Türleri*] tablosunun ilk hâline geri geldiği şekil 7.26 (b)'de görülebilir.

```
BEGIN TRANSACTION
    INSERT INTO [Stok Hareket Türleri]
        VALUES (5,'Sipariş Verildi')
    INSERT INTO [Stok Hareket Türleri]
        VALUES (6,'Sipariş İptal Edildi')
ROLLBACK
```

Şekil 7.26

(a) Başarılı **INSERT** İşlemi Sonrası (b) **ROLLBACK** İşlemi Sonrası

No	Tür Adı
1	Satin Alındı
2	Satıldı
3	Tutuluyor
4	Atık

(a)

No	Tür Adı
1	Satin Alındı

(b)

Eğer aynı **TRANSACTION** yapısı aşağıdaki örnekte görüldüğü gibi **ROLLBACK** yerine **COMMIT** ifadesi ile yazıldysa kayıtlar Bolge tablosuna kalıcı olarak aktarıldığı alttaki şekilde görülmektedir.

BEGIN TRANSACTION

```
INSERT INTO [Stok Hareket Türleri]
VALUES (5,'Sipariş Verildi')
INSERT INTO [Stok Hareket Türleri]
VALUES (6,'Sipariş İptal Edildi')
```

COMMIT

Şekil 7.27

COMMIT İfadesi ile Kayıtların Veritabanına Kalıcı Olarak Aktarılması

No	Tür Adı
1	Satin Alındı
2	Satıldı
3	Tutuluyor
4	Atık
5	Sipariş Verildi
6	Sipariş İptal Edildi

No	Tür Adı
1	Satin Alındı
2	Satıldı
3	Tutuluyor
4	Atık
5	Sipariş Verildi
6	Sipariş İptal Edildi

Ancak **TRANSACTION** yapısı içerisinde yazılmış olan komutların herhangi birinde hata olması durumunda **TRANSACTION** yapısının sonunda **COMMIT** yazılısa bile hatalan dolayı işlemlerden biri gerçekleşmeyeceği için (**TRANSACTION** bütün işlemleri tek bir işlem olarak ele aldığından ve başarım için bütün işlemlerin başarılı olması gerekişinden dolayı) alttaki şekilde gösterildiği gibi kayıt işlemleri gerçekleştirmeyecektir. Aşağıdaki örnekte [*Stok Hareket Türleri*] tablosunun [*No*] alanına bir metin girilmesi istendiğinden dolayı kayıt ekleme işlemi gerçekleşmemiştir. Çünkü, [*Stok Hareket Türleri*] tablosunun [*No*] alanı *int* değerler alacak şekilde tanımlanmıştır. İlk kayıt ekleme işlemi başarılı olsa bile ikinci işlemede hata yapıldığından dolayı başarılı olan ilk işlemdeki eylem de geri alınmıştır.

```
BEGIN TRANSACTION
```

```
INSERT INTO [Stok Hareket Türleri] VALUES (7,'Stoktan Kalktı')
INSERT INTO [Stok Hareket Türleri] VALUES ('a','Stok İptal Edildi')
```

```
COMMIT
```

Şekil 7.28

Kayıt Ekleme Sırasında Yapılan Hata Mesajı



Şekil 7.29

Hatalan Dolayı **TRANSACTION** Yapısı Kayıt Eklemeye İzin Vermemesi

No	Tür Adı
1	Satin Alındı
2	Satıldı
3	Tutuluyor
4	Atık
5	Sipariş Verildi
6	Sipariş İptal Edildi

Query executed successfully. | PC-PC\SQLEXPRESS (12.0 RTM) | pc-PC\pc (58) | master | 00:00:00 | 6 rows

Özet



Veri işleme ve veri işleme komutlarının gerekliliğini açıklamak

Bir veritabanı uygulamasının temel amacı kayıtların sisteme girilmesi ve bu kayıtların istenildiği zaman sorgulanabilmesidir. Ancak bazı durumlarda veritabanına girilen kayıtların silinmesi veya değiştirilmesi gerekebilmiştir. Bu silme gerekliliği veritabanına girilen hatalı kayıtlar olabileceği gibi artık kullanılmayan kayıtların silinmesi de olabilir. Bununla beraber yine hatalı girilmiş bir kayıt veya kayıt ile ilgili bilgilerin değişmesi durumunda güncellenme ihtiyacı ortaya çıkabilemektedir. Bahsedilen tüm isterler için tabloya yeni bir kayıt eklemek, mevcut kayıt üzerinde değişiklik yapmak veya bir kaydı silmek istediğimiz durumlarda, Select komutu kullanarak yapmış olduğumuz sorgulamalar işe yaramaz. Bunun için SQL altındaki veri işleme (DML, Data Manipulation Language) dili komutları kullanılarak, veritabanı tablosuna kayıt ekleme, silme ve güncelleme işlemleri yapılabilmektedir.



***INSERT, DELETE** ve **UPDATE** komutlarını örnekler verebilmek*

Bir tabloya doğrudan veya başka bir tablo veya tablolardan sorgulama sonucu elde edilen kayıt veya kayıtları eklemek için **INSERT** komutu kullanılır. **INSERT** komutu, doğrudan veri eklemek için **INTO** ve **VALUES** ifadeleri ile birlikte, başka bir tablodan elde edilen verileri eklemek için ise **INSERT SELECT** yapısı ile birlikte kullanılır. **DELETE** komutu kullanarak bir tabloda bulunan kayıt veya kayıtların istenildiği zaman silinmesi mümkündür. Silinmek istenen kayıt koşul ifadesi ile bulunur. Tablodaki tüm kayıtların silinmesi için koşul ifadesi kullanılmadan **DELETE** komutu kullanılabileceği gibi **TRUNCATE** komutunu da kullanmak mümkündür. **TRUNCATE** komutu ile yapılan silme işlemi geri alınamaz. Veritabanı üzerinde veri güncelleme işlemleri de tipki **DELETE** işlemlerinde olduğu gibi çok dikkat gerektiren işlemlerden dir. **UPDATE** komutu ile bir veritabanı üzerinde bir tabloda istenilen bir kayıt veya kayıtların güncelleme işlemi yapılabilir. Veritabanındaki kayıtların silinmesi veya güncelleme işlemleri yapılmırken dikkatli olunması gerekmektedir.



Birden fazla tablo ile eylem sorgularını örnek uygulamalarını yapmak

Birbiri ile bağlanmış tablolarda ana tablodaki kaydın silinmesi ile onunla ilişkili bağlantılı kayıtların silinmesini sağlayacak olan bir **CASCADE DELETE** (kademeli silme) yapısı SQL dilinde mevcuttur. Bu yapı tablo oluşturulurken **ON DELETE CASCADE** komutu ile yapılabilir. Bu komutta temel olarak amaçlanan veri bütünlüğünü sağlamak olmasına rağmen hangi tablonun hangi tablo ile bağlantılı olduğunun takibi ni yapmadan silme işlemi gerçekleştirmek mümkün olmaktadır.



***TRANSACTION, ROLLBACK** ve **COMMIT** ifadelerinin kullanımını açıklamak*

Veri işleme komutları kullanarak bir tabloya yeni bir kayıt eklemek, mevcut kayıt üzerinde değişiklik yapmak veya bir kaydı silmek tek başına kullanılabileceği gibi bu işlemler tek bir adımda arka arkaya yapılması da gerekebilir. SQL dili **TRANSACTION** yapısı ile arka arkaya yapılması gereken bir veya birden fazla veri işleme komutlarının hepsini tek bir işlem altında toplama imkânı vermektedir. Eğer işlemlerden biri yapılmaz ise yapılan diğer işlemlerde geri alınması gerekmektedir. Bunun için **ROLLBACK** kavramı geliştirilmiştir. Bu sayede veritabanındaki yapılan değişikliklerin geri alınabilmesi sağlanmıştır. Yapılan işlemlerin veritabanında kalıcı olarak aktarılması isteniyorsa **COMMIT** ifadesi kullanılır.

Kendimizi Sınayalım

- 1.** Aşağıdaki komutlardan hangisi veri işleme komutlarından biri **değildir**?
- INSERT
 - CREATE
 - DELETE
 - UPDATE
 - TRUNCATE
- 2.** SQL dilinde aşağıdaki komutlardan hangisi ile kayıt ekleme işlemi yapılabilir?
- INSERT
 - CREATE
 - DELETE
 - UPDATE
 - TRUNCATE
- 3.** Aşağıdaki fonksiyonlardan hangisi metindeki küçük harfleri büyük harfe çevirmek için kullanılır?
- LEFT()
 - UPPER()
 - LOWER()
 - ROUND()
 - LTRIM()
- 4.** Alan tipi metin olarak seçilmiş tabloya değer girebilmek için metinler aşağıdaki hangi iki karakterler arasında yazılmalıdır?
- & (ve işaretti)
 - % (yüzde işaretti)
 - ^ (üssel işaretti)
 - ((parantez işaretti)
 - ' (tek tırnak işaretti)
- 5.** Bir tabloya kayıt ekleme, silme ve güncelleme yapabilmek için yazılan programlama dilinin adı aşağıdakilerden hangisidir?
- Veri İşleme Dili (Data Manipulation Language)
 - Veri Tanımla Dili (Data Definition Langauge)
 - İlişkisel Veritabanı Dili (Relational Database Language)
 - Veritabanı Dili (Database Language)
 - Veri Seçme Dili (Data Select Language)
- 6.** Aşağıdaki komutlardan hangisi TRANSACTION yapısıını sonlandırmak için kullanılır?
- VIEW
 - COMMIT
 - SELECT
 - FLASHBACK
 - END
- 7.** Değişiklik yapılan kayıtların veritabanına kalıcı olarak aktarılması için kullanılan komut aşağıdakilerden hangisidir?
- ROLLBACK
 - COMMIT
 - SELECT
 - FLASHBACK
 - END
- 8.** Aşağıdaki komutlar çalıştırıldığında veritabanı nasıl etkilenecek?
- ```
BEGIN TRANSACTION
INSERT INTO Siparişler [Müşteri no]
VALUES (29)
ROLLBACK
```
- Rollback komutundan dolayı kayıt ekleme işlemi geri alınır, veritabanı değişmez.
  - Rollback komutundan dolayı kayıt ekleme kalıcı olarak veritabanına aktarıldığından dolayı işlemi veritabanı değiştir.
  - Rollback komutundan dolayı [Siparişler] tablosundaki bütün kayıtlar silinir.
  - Rollback komutundan dolayı [Siparişler] tablosunda ki müşteri numarası '29' olan kayıtlar silinir.
  - Rollback komutundan dolayı [Siparişler] tablosunda ki bütün kayıtlar güncellenir.
- 9.** Birbiri ile bağlantılı tablolarda bir kayıt silindiği zaman bağlantılı olan diğer kayıtların kademeli olarak silinmesi için aşağıdakilerden hangisi kullanılır?
- CREATE
  - INSERT
  - UPDATE
  - ON DELETE CASCADE
  - DELETE

- 10.** Aşağıdaki komutlardan hangisi kayıtların güncellenmesi için kullanılır?

- TRANSACTION
- INSERT
- UPDATE
- SELECT
- DELETE

## Kendimizi Sınavalım Yanıt Anahtarı

1. b Yanınız yanlış ise “Giriş” konusunu yeniden gözden geçiriniz.
2. a Yanınız yanlış ise “INSERT Komutu Yapısı” konusunu yeniden gözden geçiriniz.
3. b Yanınız yanlış ise “INSERT Komutu Yapısı” konusunu yeniden gözden geçiriniz.
4. e Yanınız yanlış ise “INSERT Komutu Yapısı” konusunu yeniden gözden geçiriniz.
5. a Yanınız yanlış ise “Giriş” konusunu yeniden gözden geçiriniz.
6. b Yanınız yanlış ise “ROLLBACK ve COMMIT Kavramı” konusunu yeniden gözden geçiriniz.
7. b Yanınız yanlış ise “ROLLBACK ve COMMIT Kavramı” konusunu yeniden gözden geçiriniz.
8. a Yanınız yanlış ise “ROLLBACK ve COMMIT Kavramı” konusunu yeniden gözden geçiriniz.
9. d Yanınız yanlış ise “Birden Fazla Tablo ile Eylem Sorguları” konusunu yeniden gözden geçiriniz.
10. c Yanınız yanlış ise “UPDATE Komutu” konusunu yeniden gözden geçiriniz.

## Sıra Sizde Yanıt Anahtarı

### Sıra Sizde 1

Metin sonundaki boşlukların temizlenmesi için **RTRIM()**, metin başında yer alan boşlukların temizlenmesi için **LTRIM()** işlevleri kullanılabilir.

```
INSERT INTO Kitaplar(KitapAdı)
Values(RTRIM('Veritabanları Sistemleri'))
```

### Sıra Sizde 2

**WHERE** den sonra yazılan ifadeler ile iki tarih arasındaki ilişkileri bulunarak **DELETE** komutu ile silme işlemi gerçekleştirilen komut aşağıda verilmiştir.

```
DELETE Siparişler
WHERE [Sipariş Tarihi]>='2015-03-01' and
[Sipariş Tarihi]<='2016-03-01'
```

Bununla beraber tarih ifadeleri **BETWEEN** ifadesi ile birlikte kullanılarak ta yazmak mümkündür.

```
DELETE Satislar
WHERE SatisTarihi BETWEEN '2015-03-01'
and '2016-03-01'
```

### Sıra Sizde 3

Birbirile bağlanmış tablolarda ana tablodaki kaydın güncellenmesi ile onunla ilişkili bağlantılı kayıtların güncellemesini sağlayacak yapı **ON UPDATE CASCADE** yapısıdır. **ON DELETE CASCADE** yapısı gibi tanımlanmaktadır ve kullanılmaktadır.

## Yararlanılan ve Başvurulabilecek Kaynaklar

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

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

Transact-SQL Reference: <https://msdn.microsoft.com/en-us/library/bb510741.aspx>, son erişim tarihi: 07.11.2016

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

Yarımagañ, Ü (2010). **Veritabanı Yönetim Sistemleri**, Akademî Yayıncılık.

<https://support.office.com/tr-tr/article/Veritaban%C4%B1na-bir-veya-daha-fazla-kayıt-ekleme-48307788-b214-4b84-bb42-cf627d447f64>, son erişim tarihi: 07.11.2016

# 8

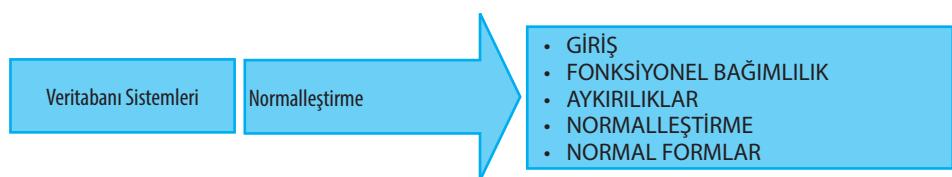
### Amaçlarımız

- Bu üniteyi tamamladıktan sonra;
- 🕒 İyi tasarılmış bir veritabanının özelliklerini sıralayabilecek,
  - 🕒 Fonksiyonel bağımlılığı açıklayabilecek,
  - 🕒 Ekleme, silme ve güncelleme aykırılıklarını tanımlayabilecek,
  - 🕒 Normalleştirmenin amaçlarını açıklayabilecek,
  - 🕒 Normal formları ifade edebilecek,
  - 🕒 Birinci, ikinci ve üçüncü normal formlara uygun şekilde normalleştirme yapabilecek bilgi ve becerilere sahip olacaksınız.

### Anahtar Kavramlar

- Fonksiyonel Bağımlılık
- Kısmi Bağımlılık
- Geçişli Bağımlılık
- Aykırılık
- Normalleştirme
- Normal Form

### İçindekiler



# Normalleştirme

## GİRİŞ

Veritabanı tasarımları kavramsal, mantıksal ve fiziksel olmak üzere üç aşamada gerçekleştirilir. Kavramsal modelleme aşamasında veriler çözümlenir ve veriler arasındaki ilişkiler incelenerek varlık ilişki diyagramları (Entity Relationship - ER) oluşturulur. Bu aşamanın devamında mantıksal modelleme ile veritabanının yönetim sistemi bağımsızlığı, teknoloji, veri depolama veya organizasyonel kısıtları sağlanır. Mantıksal tasarımda artık veriler ve ilişkiler veritabanı tabloları ve tablolar arası ilişkilere dönüştürülür. Son aşamada da veri süreklilik teknolojisini bir sürümü ile tanımlanmış fiziksel veri modeli oluşturulur. Veritabanı tasarım aşamalarını oluşturan modellemeler daha geniş olarak 2. Üniteye anlatılmıştır.

Veritabanı modelleme aşamalarına ilave olarak tasarımımız **normalleştirme** kurallarına uygun olarak düzenlenmelidir. Normalleştirme ile tablolar ve tablolar arasındaki ilişkiler tekrar düzenlenerek tutarsız bağımlılıklar kaldırılır ve artıklıklar (redundancy) en aza indirilir. İlk defa tasarlanan bir veritabanı mantıksal modelinde ilişkiler (tablolar ve tablolar arasındaki ilişki tanımları) normalleştirme kuralları ile iyi yapılandırılmış hâle getirilir. Benzer şekilde mevcut bir veritabanındaki ilişkilerin iyileştirilmesi içinde normalleştirme kullanılır. Normalleştirme kuralları iki öznitelik arasındaki kısıtlara dayanan fonksiyonel bağımlılık kavramı ile test edilir. Her aşamada uygulanan kurala Normal Form (NF) adı verilir. İlk kural uygulanıyorsa veritabanı tasarıminin birinci normal formda (1NF) olduğu, ikinci kural uygulanıyorsa ikinci normal formda (2NF) olduğu şeklinde ifade edilir. Her aşamada uygulanan normalleştirme kurallarına göre veritabanı tasarım seviyesi tanımlanmış olur. Kisaca ilgili normal formun adı verildiğinde, veritabanı ilişkilerinin hangi kurallara uyacak şekilde normalleştirildiği anlaşılır.

Veritabanı modelinden sorunlu fonksiyonel bağımlılıklarının aşama aşama kaldırılması işlemine **normalleştirme** denir.

Bu üitede öncelikle normal formların kurallarını oluşturan fonksiyonel bağımlılıklar ve veritabanında oluşabilecek aykırılıklar (anomaly) tanımlanacaktır. Aykırılıklar aynı zamanda veritabanının normal formlara uymaması anlamına gelir. Ünitenin devamında ise normal formların nasıl test edileceği ve bağımlılıkların sebep olduğu aykırılıkların ve tekrarlanan grupların nasıl giderileceği anlatılacaktır.

## FONKSİYONEL BAĞIMLILIK

Normalleştirme fonksiyonel bağımlılıkların analizine dayalı olarak yapılır. Fonksiyonel bağımlılık iki set öznitelik arasındaki kısıtlardır. Bir veritabanındaki herhangi bir tablo T ve bu tablodaki iki öznitelik A ve B olsun. Eğer A özniteliğinin değeri B özniteliğinin değerini belirliyorsa B özniteliğinin A özniteliğine bağımlı olduğu söylenir. B'nin A'ya fonksiyonel bağımlılığı ok işaretleri ile  $A \rightarrow B$  şeklinde gösterilir. Burada tanımlanan fonksiyonel bağımlılık

matematiksel bağımlılık değildir. Yani B'nin değeri A'nın değerinden hesaplanmaz. Bunun yerine A'da var olan her bir değer için B'de bir değer olduğunu gösterir. Bu ilişki A özniteligi B özniteligini belirler şeklinde ifade edilir. Fonksiyonel bağımlılığı Şekil 8.1'de verilen bir faturadaki bazı bilgilerin tutulduğu örnek satış kayıtları tablosu ile açıklayalım. Satış kayıtları tablosunda; satışa ait bilgiler, müşteriye ait bilgiler, satılan ürünne ait bilgiler ve ürünün tedarikçi bilgileri bulunmaktadır. Verilen tabloda, ilk üç satırda üç ayrı faturada birer ürün satışı gösteren ve son iki satırda da tek faturada iki ürün satışı gösteren kayıtlar bulunmaktadır. Bu tabloda Satış Id ve Ürün Id'sinin birlikte bütün alanların değerlerini belirlediği görülür. Bu nedenle tabloda Satış Id ve Ürün Id birleşik anahtar olarak tanımlanmıştır.

DİKKAT



**Alan, sütun ve öznitelik ifadelerinin hepsi veritabanı tablosundaki alanları belirmek için kullanılan terimlerdir.**

Bu örnekte;

Satış Id, Ürün Id →

Satış Tarihi, Müşteri Id, Müşteri Adı, Müşteri Soyad, Müşteri Adres, Ürün Adı, Birim Fiyat, Tedarikçi Yeri, Tedarikçi Şehir, Satış Miktarı, Toplam Fiyat

fonksiyonel bağımlılığı yazılabilir. Çünkü bu tabloda Satış Id ve Ürün Id alanları birlikte benzersiz değerler almaktadır. Bunun yanı sıra aşağıda verildiği gibi diğer tüm alanlar ile birleşik anahtarın arasında fonksiyonel bağımlılık yazılabilir.

Satış Id, Ürün Id → Satış Tarihi,

Satış Id, Ürün Id → Müşteri Id,

Satış Id, Ürün Id → Müşteri Adı,

⋮

Satış Id, Ürün Id → Toplam Fiyat

Ama Ürün Adı → Satış Miktarı fonksiyonel bağımlılığı yazılamaz. Çünkü Ürün Adı alanı tablodaki iki satır için "Zeytin Yağı" değerini aldığında Satış Miktarı alanı "8" ve "65" gibi iki farklı değer almaktadır. Yani Ürün Adı Satış Miktarını belirlemez. İzleyen kesimde diğer bağımlılık türleri açıklanacak ve örnekler verilirken aşağıdaki satış kayıtları tablosu kullanılacaktır.

**Şekil 8.1**

Satış Kayıtları tablosu için fonksiyonel bağımlılıklar

Tam Fonksiyonel Bağımlılık

| Geçişli Bağımlılık |              |            |             |               |               | Geçişli Bağımlılık |             |             |                |                 |               |              |
|--------------------|--------------|------------|-------------|---------------|---------------|--------------------|-------------|-------------|----------------|-----------------|---------------|--------------|
| Satış Id           | Satış Tarihi | Müşteri Id | Müşteri Adı | Müşteri Soyad | Müşteri Adres | Ürün Id            | Ürün Adı    | Birim Fiyat | Tedarikçi Yeri | Tedarikçi Şehir | Satış Miktarı | Toplam Fiyat |
| 1                  | 1.01.2014    | 1          | Aliye       | Berber        | 1.sokak...    | NWTB-1             | Hint Çayı   | 1,33        | Depo1          | Eskişehir       | 100           | 133          |
| 2                  | 1.05.2016    | 2          | Temel       | Giresunlu     | 2.sokak...    | NWTS-8             | Köri Sosu   | 20          | Depo2          | Ankara          | 6             | 120          |
| 3                  | 1.08.2016    | 1          | Aliye       | Berber        | 1.sokak...    | NWTO-5             | Zeytin Yağı | 10          | Depo1          | Eskişehir       | 8             | 80           |
| 4                  | 15.08.2016   | 6          | Feridun     | Olcay         | 6.sokak...    | NWTC-3             | Soda        | 18          | Depo2          | Ankara          | 4,5           | 731          |
| 4                  | 15.08.2016   | 6          | Feridun     | Olcay         | 6.sokak...    | NWTO-5             | Zeytin Yağı | 10          | Depo1          | Eskişehir       | 65            | 731          |

Kısmi Bağımlılık

Kısmi Bağımlılık

## Kısmi Bağımlılık (Partial Dependence)

Bir tabloda birincil anahtar bir veya daha fazla öznitelikten oluşabilir. Eğer birincil anahtar iki veya daha fazla öznitelikten oluşuyorsa bu tür birincil anahtarlarla birleşik anahtar (composite key) adı verilir. Bu tür tablolarda anahtar olmayan öznitelik, birleşik anahtarın sadece bir kısmı ile belirleniyorsa buna kısmi bağımlılık denir. Bir veritabanında yer alan tabloda A, B, C ve D gibi toplam dört özniteliğin olduğunu varsayıyalım. Bu tablonun birincil anahtarı da (A, B) olsun. Bu durumda  $AB \rightarrow CD$  fonksiyonel bağımlılığı yazılabilir. Bu tabloda  $A \rightarrow C$  fonksiyonel bağımlılığını da varsayıyalım. Bu durumda C özniteliği birleşik anahtarın sadece bir kısmı olan A özniteliğine bağlıdır. Yani C özniteliği A'ya kısmı bağlıdır.

Kısmi bağımlılığı Şekil 8.1'de verilen örnek tablo üzerinden açıklayalım. Satış bilgilerini içeren tabloda Satış Id, Ürün Id birlikte veri tekrarını önlemek amacıyla birleşik anahtar oluştursun. Böylece bir siparişte hangi üründen, hangi müşteriye ne miktarda satıldığı ve toplam fiyatı bu birleşik anahtar ile benzersiz olarak belirlenebilmektedir. Bununla birlikte Satış Tarihi, Müşteri Id, Müşteri Adı, Müşteri Soyad, Müşteri Adres alanları sadece birleşik anahtarın Satış Id alanına, Ürün Adı, Birim Fiyat, Tedarikçi Yeri, Tedarikçi Şehir alanları ise birleşik anahtarın sadece Ürün Id alanına bağlıdır. Bu tür oluşan bağımlılığa kısmi bağımlılık denir. Verilen örnekte kısmi bağımlılıklar; Satış Id  $\rightarrow$  Satış Tarihi, Müşteri Id, Müşteri Adı, Müşteri Soyad, Müşteri Adres ve Ürün Id  $\rightarrow$  Ürün Adı, Birim Fiyat, Tedarikçi Yeri, Tedarikçi Şehir olarak gösterilir.

## Geçişli Bağımlılık (Transitive Dependence)

Bir tabloda yer alan bazı özniteliklerin başka bir öznitelik aracılığıyla üçüncü bir öznitelijke bağımlı olması geçişli bağımlılık olarak adlandırılır. Bir veritabanında yer alan tablonun A, B, C ve D şeklinde toplam dört özniteliği olsun. Bu tabloda A birincil anahtar olsun.  $A \rightarrow B$  ve  $B \rightarrow C$  fonksiyonel bağımlılıklarının olduğunu varsayıyalım. Verilen bu iki fonksiyonel bağımlılık incelendiğinde, C özniteliğinin B özniteliği aracılığıyla A özniteliğine bağımlı olduğu görülür. Bu durum geçişli bağımlılık olup  $A \rightarrow C$  fonksiyonel bağımlılığı yazılabilir. Başka bir deyişle geçişli bağımlılıkta; C özniteliği, A özniteliği ile bir diğer öznitelik B üzerinden dolaylı olarak belirlenir.

Örneğin Şekil 8.1'de Müşteri Id, Müşteri Adı, Müşteri Soyad, Müşteri Adres alanları Satış Id alanına geçişli bağımlıdır. Burada Satış Id alanı Müşteri Id'yi belirler, Müşteri Id ise Müşteri Adı, Müşteri Soyad, Müşteri Adres alanlarını belirler. Bu ilişkide Müşteri Adı, Müşteri Soyad, Müşteri Adres alanları, Satış Id alanına Müşteri Id alanı üzerinden dolaylı olarak bağımlıdır.

## Tam Fonksiyonel Bağımlılık (Full Functional Dependence)

Bir özniteliğin değeri bağlı olduğu anahtar ile benzersiz olarak belirleniyorsa bu ilişki tam bağımlılık olarak adlandırılır. Örneğimizde Satış Miktarı ve Toplam Fiyat alanları Satış Id, Ürün Id birleşik anahtarına tam bağımlıdır. Satış Miktarı ve Toplam Fiyat sadece Satış Id veya sadece Ürün Id alanı ile benzersiz olarak belirlenemez.

## Çok Değerli Bağımlılık (Multiple Valued Dependence)

Tabloda bir alandaki değerler virgülle ayrılarak oluşturulan liste veya dizi değerlerinden oluşuyorsa çok değerli bağımlılık vardır. Şekil 8.1'de verilen satış kayıtları tablosunda fakturadaki her bir müşteri ve satılan ürün bilgisi ayrı satırda saklanmaktadır. Bu tabloda her bir fakturaki satışlar tek satırda saklanmak istenirse Şekil 8.2'de olduğu gibi düzenlenebilir. Bu durumda her bir fakturadaki satışlar tek bir alan (Satış Id) ile benzersiz olarak belirlenebilir. Ancak böyle tasarlanmış bir tabloda, bir ürünün satıldığı müşterileri bulmak için daha karmaşık sorgular gerekecektir. Böyle bir sorguda Ürün Id hesaplanarak kullanılacağından, sorgular daha karmaşık olacak ve performans sorunu oluşacaktır. Bu nedenle tablo alanlarında değerler atomik (Her bir satır ve sütun kesişiminde tutulan veri tek değerli olmalıdır.) olarak tutulmalıdır.

**Şekil 8.2**

Satış Kayıtları tablosunda çok değerli alanlar

| Satış Id | Satış Tarihi | Müşteri Id | Müşteri Adı | Müşteri Soyad | Müşteri Adres | Ürün Id        | Ürün Adı          | Birim Fiyat | Tedarikçi Yeri | Tedarikçi Şehir   | Satış Miktarı | Toplam Fiyat |
|----------|--------------|------------|-------------|---------------|---------------|----------------|-------------------|-------------|----------------|-------------------|---------------|--------------|
| 1        | 1.01.2014    | 1          | Aliye       | Berber        | 1.sokak...    | NWTB-1         | Hint Çayı         | 1,33        | Depo1          | Eskişehir         | 100           | 133          |
| 2        | 1.05.2016    | 2          | Temel       | Giresunlu     | 2.sokak...    | NWTS-8         | Köri Sosu         | 20          | Depo2          | Ankara            | 6             | 120          |
| 3        | 1.08.2016    | 1          | Aliye       | Berber        | 1.sokak...    | NWTO-5         | Zeytin Yağı       | 10          | Depo1          | Eskişehir         | 8             | 80           |
| 4        | 15.08.2016   | 6          | Feridun     | Olcay         | 6.sokak...    | NWTC-3; NWTO-5 | Soda; Zeytin Yağı | 18;10       | Depo2; Depo1   | Ankara; Eskişehir | 4,5;65        | 731          |

Çok Değerli Alanlar

### Döngüsel Bağımlılık (Cyclic Dependency)

Döngüsel bağımlılıkta döngü kapalı halka, tekrar etme anlamında kullanılmaktadır. Veritabanında bir öznitelik A, diğer öznitelik B'ye bağımlı iken aynı zamanda B özniteligi de A'ya bağımlı ise döngüsel bağımlılık olarak adlandırılır. Döngüsel bağımlılık genelde tablolardaki iki ya da daha fazla alandan oluşan birleşik anahtarlarda oluşur.

SIRA SİZDE

1

**Şekil 8.3'te öğretim elemanları ve bölümlerde verdikleri derslerin bilgilerinin kayıtlı olduğu tablo verilmiştir. Bu tabloda Id ve Kod (ders kodu) birleşik anahtar olarak tanımlıdır. Bu tabloya göre kısmi ve geçişli bağımlılığa birer örnek veriniz.**

**Şekil 8.3**Öğretim elemanı  
ders bilgisi tablosu

| Id | Yıl  | Adı   | Mahalle   | Telefon Numarası | Kod | Ders Adı              | Kredi |
|----|------|-------|-----------|------------------|-----|-----------------------|-------|
| 1  | 2015 | Uğur  | Yeşiltepe | 3015             | 101 | Matematik             | 5     |
| 2  | 2015 | Kemal | Gültepe   | 3016             | 201 | Veritabanı Sistemleri | 4     |
| 3  | 2015 | Metin | Esentepe  | 3045             | 303 | Veri Madenciliği      | 3     |
| 3  | 2016 | Metin | Esentepe  | 3045             | 402 | Ağ Güvenliği          | 3     |
| 5  | 2016 | Kemal | Tepebaşı  | 3051             | 201 | Veritabanı Sistemleri | 4     |

### AYKIRILIKLAR

İlişkisel veritabanında bir satırda verinin (veya satırlardaki verilerin) hatalı olarak değiştirilmesine **aykırılık (anomaly)** denir.

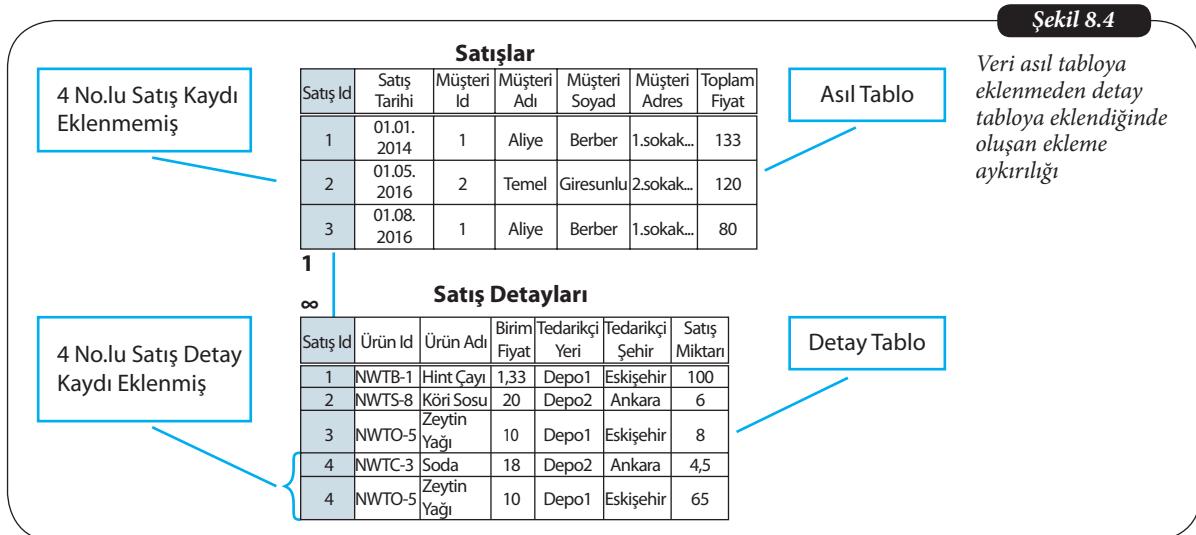
Veritabanında verinin değiştirilmesi sırasında mantıksal olarak bozulmaya yol açan durumlara **aykırılık (anomaly)** denir. Özellikle büyük veritabanlarında büyük miktardaki kayıtlar tek komut ile değiştirildiğinde veya eklendiğinde bir aykırılık oluşursa ortaya çıkan sorunun çözümü de çok zor olacaktır. Bu nedenle iyi tasarılanmış bir ilişkisel veritabanında aykırılıklar giderilmelidir.

Bir veritabanında görülebilecek üç aykırılık vardır. Bunlar ekleme, silme ve güncelleme aykırılıkları olarak sıralanabilir. Eğer bir tabloda bu aykırılıklardan biri veya birkaç varsa bu tablo ve dolayısıyla veritabanının iyi tasarılanmadığı söylenir. Bu nedenle veritabanı tasarımları bu aykırılıkların oluşmayacağı şekilde iyileştirilmelidir.

### Ekleme Aykırılığı

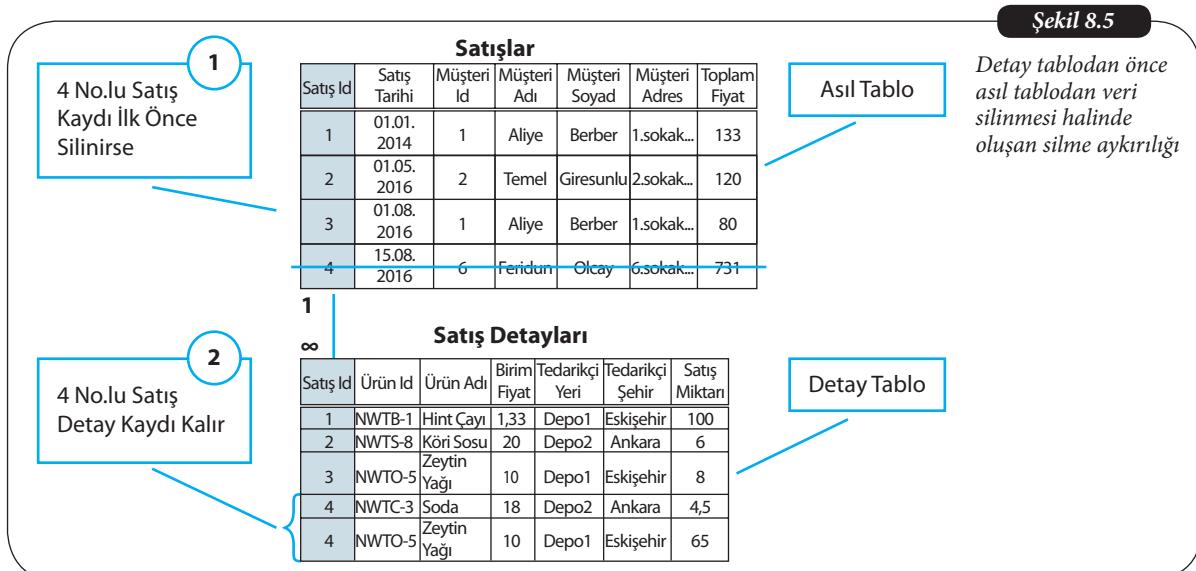
Veritabanında birincil ve yabancı anahtar ile ilişkilendirilmiş iki tablo olsun. Birinci tabloda kayıtların özet bilgilerinin, diğerinde ise ilgili kayıtların detay bilgilerinin tutulduğunu varsayıyalım. Bu iki tablo asıl ve detay tablolar olarak isimlendirilir. Asıl ve detay tablolarda, detay tabloya veri eklenirken asıl tabloya ilgili veri eklenmez ise ekleme

aykırılığı oluşur. Ekleme aykırılığı Şekil 8.4'te gösterilmiştir. Bu örnekte Satış Detayları tablosunda 4 no.lu satışa ait detay bilgiler olduğu halde asıl tablo olan Satışlar tablosuna 4 no.lu satış kaydı eklenmemiştir. Bu durumda veri bütünlüğü bozulmuş olur ve Satışlar tablosu üzerinden 4 no.lu satışın detay hareketlerine erişilemez. Ekleme aykırılığı oluşmaması için öncelikle asıl tabloya kayıt eklenmeli daha sonra detay tabloya ilgili kaydın detayları eklenmelidir.



## Silme Aykırılığı

Silme aykırılığı, asıl ve detay ilişkili tablolarda, kayıtların detay bilgilerini silmeden asıl tablodaki satırlarının silinmesi ile oluşur. Önceki ünitelerde anlatılmış olan “cascade delete” SQL komutu kullanılması hâlinde bu aykırılık oluşmayacaktır. “Cascade” komutu öncelikle detay tablolarındaki kayıtları siler daha sonra detay tablolarının bağlı olduğu asıl tablodaki kaydı siler. Şekil 8.5'te verilen 4 no.lu kayıt (1 nolu çember ile işaretli) öncelikle silinirse, detay tablosu olan Satış Detayları tablosundaki 4 no.lu satışa ait kayıtlar (2 nolu çember ile işaretli) kalacaktır. Bu kayıtlara Satışlar tablosundan Sipariş Id ilişkisi ile erişmek mümkün olmayacağından bu aykırılığı önlemek için öncelikle detay tablodan sonra da asıl tablodan kayıtlar silinmelidir.



## Güncelleme Ayrılığı

Bu aykırılık silme aykırılığına benzer. Asıl ve detay ilişkili tablolarda güncelleme yaparken asıl tablodaki veri yok edilerek detay tablosundaki veriler ilişkisiz bırakılmamalıdır. Bu aykırılığı önlemek için birincil anahtar değerinde güncelleme yaparken detay tablolarda da ilişkili yabancı anahtarlar sırasıyla (cascade) güncellenmelidir.

SIRA SİZDE



**Şekil 8.3'te bir öğretim elemanına ait kayıtları silmek istediğimizde silme aykırılığı oluşan kayıtlar hangileridir? Neden?**

## NORMALLEŞTİRME

Veritabanı tasarlarken kullanılan tablolar ve tablolar arası ilişkiler ile ortaya konulan modelin iyi bir tasarım olup olmadığına karar vermek gerekir. Eğer tablolar veya aralarındaki ilişkiler iyi tanımlanmamış ise daha küçük tablolara bölünmeli ve tablolar arası yeni ilişkiler oluşturulmalıdır. Normal formlar bu iyileştirme işlemlerinin yapılması için rehberlik yapmak üzere tanımlanmıştır. Eğer tanımlanmış olan ilişkisel model bu normal formlar dan herhangi birine uyuyorsa modelden ilgili hatalar giderilmiş demektir. **Normalleştirme** her biri bir normal forma karşılık gelecek şekilde aşamalı olarak gerçekleştirilir.

**Normalleştirme** tablolar ve aralarındaki ilişkilerin aykırılıkları azaltmak üzere aşamalı olarak daha küçük ve iyi yapılandırılmış tablolar ve ilişkilere dönüştürülmesidir.

### Normalleştirme Meninin Amaçları

1. Veri artıklığını (redundancy) minimum yapmak. Böylece aykırılıklar önlenmiş ve fiziksel veri saklama alanından tasarruf edilmiş olur.
2. Veri bütünlüğünü sağlayan kısıtların uygulamasını basitleştirmek. Eğer bir alan tanımı birden fazla tablo var ise farklı tablolardaki bu alanlar bir süre sonra birbirinden farklı değerler içermeye başlayacaktır. Bu tip aynı alanın farklı tablolarda tekrar tanımının önlenmesi gerekir. Ayrıca bir tabloda aynı bilgiler yinelenen satırlarda saklanırsa ekleme, silme ve güncelleme aykırılıklarına sebep olur ve veri bütünlüğünü korunamaz. Aykırılığa sebep olan ilişkiler tekrar yapılandırılmalıdır.
3. Veri işlemeyi (ekleme, güncelleme ve silme) daha basit hâle getirmek. Daha basit SQL komutları daha yüksek performansta veriye erişimi sağlayabilecektir.
4. Gerçek varlık ve ilişkileri daha iyi temsil edecek veritabanı modeli tasarnımını gerçekleştirmek ve ilerideki geliştirmeler için sağlam bir taban oluşturmak.

### Normalleştirme Aşamaları

Normal form fonksiyonel bağımlılıklara göre uygulanan basit kuralların sonucunda oluşturulan veritabanı tasarnımını gösteren modelin aşamasıdır. Bu formlar ve içeriği kurallar aşağıda kısaca tanımlanacak ve izleyen kesimde detaylı olarak açıklanacaktır.

#### 1. Normal Form (1st Normal Form - 1NF)

Tablolar ilişkilendirilebilir şekilde tasarılanır ve çok değerli öznitelikler kaldırılır. Tablonun her bir satır ve sütun kesiminde atomik değer olacak şekilde tablo tekrar yapılandırılır. Birden fazla bilgi tek bir sütunda tutulamaz. Tekrarlanan veriler kaldırılır. 1NF'de bir tablodaki tüm alanlar birincil anahtar (birleşik anahtar olabilir) ile doğrudan ya da dolaylı benzersiz olarak belirlenebilir.

#### 2. Normal Form (2nd Normal Form - 2NF)

Bu aşamada kısmi fonksiyonel bağımlılıklar kaldırılır. Tüm alanlar birincil anahtar ile belirlenebilir olmalıdır. Bu aşamada asıl tablodaki birincil anahtara bağlı olan ancak tekrar eden alanlar yeni bir tabloya taşınır. Asıl tablo ile yeni tablo arasında çoka bir ilişkisi kurulur. Yeni tablo tekrar eden alanları tek kayıt altında birincil anahtar ile belirler. Yeni tablodaki birincil anahtar asıl tabloya yabancı anahtar olarak yerleştirilir. Asıl tabloya yerleştirilen yabancı anahtar aynı tablodaki birincil anahtarın bir parçası değildir.

### 3. Normal Form (3th Normal Form - 3NF)

Geçişli fonksiyonel bağımlılıklar kaldırılır. Yani birincil anahtar tarafından dolaylı olarak belirlenen alanlar yok edilir. Geçişli fonksiyonel bağımlılığı kaldırmak için asıl tablodaki birincil anahtara dolaylı bağımlı alanlar yeni bir tabloya taşınır.

#### *Boyce-Codd Normal Form (BCNF)*

Fonksiyonel bağımlılıktan geri kalan tüm aykırılıklar kaldırılır. Bir tablo 3NF'de ise ve her belirleyici anahtar olarak tanımlandıysa ilgili tablo BCNF formunda denir. Bu tanıma göre eğer bir tablo BCNF ise bu tablo aynı zamanda 3NF'dir. Ama tersi doğru değildir.

### 4. Normal Form (4th Normal Form - 4NF)

Çok değerli hiç bir bağımlılık kalmaz. Birincil anahtar olan alanlar ile diğer alanlar arasındaki her bir bağımsız bire-çoklu ilişki için ayrı tablo oluşturmak gereklidir. Tekrarları önlemek için her tablo mümkün olduğunda küçük parçalara bölünür.

### 5. Normal Form (5th Normal Form - 5NF)

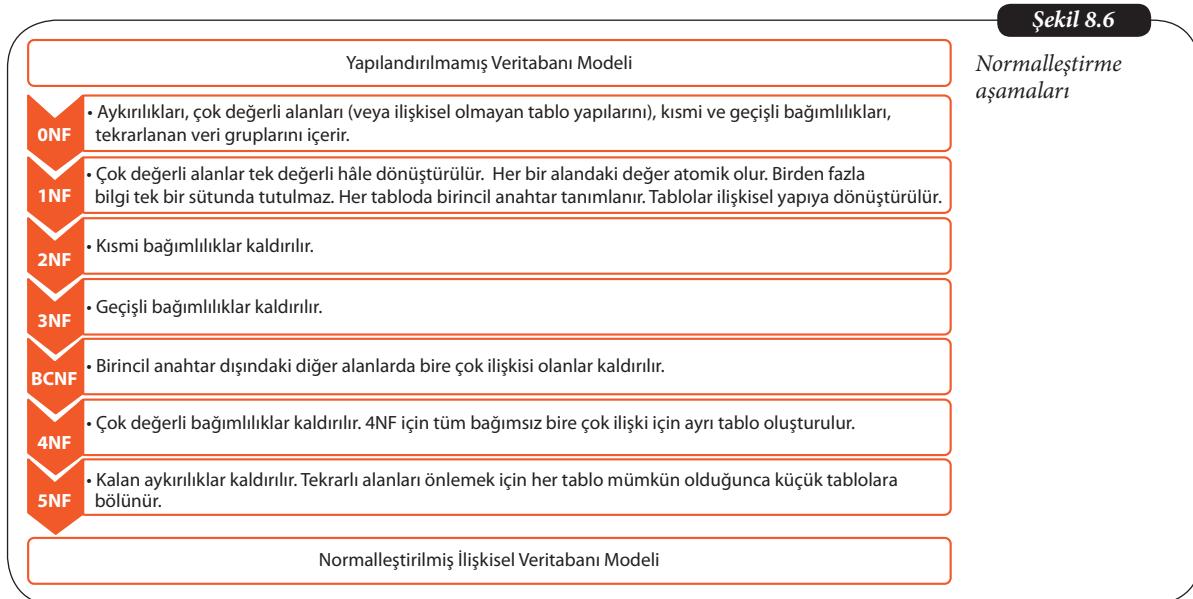
Bu aşamada döngüsel fonksiyonel bağımlılıklar ve kalan tüm aykırılıklar kaldırılır. Projeksiyon Normal Form (PJNF) olarak adlandırılır. 4NF'de kaldırılmamış olan tekrarlamalar daha küçük tablolara bölünerek kaldırılır.

#### *Etki Alanı Anahtarı Normal Formu (Domain Key Normal Form-DKNF)*

Bu aşama bir dönüşüm olmak yerine en üst seviyede normalize edilmiş veritabanı kavramsal modeli seviyesinin ölçüsüdür.

Normal formlar her aşamada daha fazla kısıt uygulamaktadır. Öncelikle 1inci Normal Form ile tablolar ilişkisel yapıya dönüştürülür ve izleyen aşamalarda sırasıyla kısmi ve geçişli fonksiyonel bağımlılıklar ve aykırılıklar mantıksal veritabanı tasarım modelinden kaldırılır. Veritabanı normalleştirmesinde önce 1NF daha sonra 2NF olmak üzere her aşama sırayla uygulanır. Herhangi bir aşama atlanmaz ve sıralama değiştirilmez. Şekil 8.6'da Normal formların uygulanan aşamaları ve her aşamada giderilen bağımlılık ve aykırılıklar verilmiştir.

**Şekil 8.6**



Genel olarak normalleştirme tekrarlanan veriyi kaldırarak verideki artıklı grupları minimize eder. Bunun sonucu olarak aykırılıkların kaldırıldığı daha iyi bir tasarım ve fiziksel saklama alanının daha etkin kullanımı gerçekleştirilmiş olur. Günümüz modern ilişkisel veritabanı uygulamalarında 3NF'den sonraki normalleştirme aşamaları genellikle

kullanılmaz. 3NF'nin ilerisindeki normalleştirme adımları tasarlanan veritabanında çok fazla tablo üretilmesine sebep olur. Dolayısı ile de SQL komutlarındaki bitişme (join) işleminde çok fazla tablo kullanılır. Daha büyük bitişmeler daha karmaşık sorgulara, daha karmaşık sorgular da sisteme performans düşüşüne sebep olur. Genelde iyi performans, ilişkisel veritabanı tasarımındaki ilişkileri mükemmelleştirmek için yapılan daha küçük tablolara bölmeye (4NF, 5NF) tercih edilir. İzleyen kesimde birinciden üçüncüye kadar normal formlar anlatılacaktır.

## NORMAL FORMLAR

İlk aşamada varlıklar tüm öznitelikleri içerecek şekilde bir tabloda ifade edilir. Örnek ve riler tablonun satırlarına kaydedilir. Veri tekrarlarına izin verilir. Bu aşamadaki veritabanı modeli Sıfırıncı Normal Formda (0NF) kabul edilir. Şekil 8.2'de çok değerli alanlar ile oluşturululan 0NF'de tablo gösterilmiştir. Benzer şekilde her satırda bir satış bilgisini saklamak üzere çok değerli alanlar yerine her bir ürün için ayrı alanlar kullanılrsa Şekil 8.7'de verilen tablo elde edilir. Bu tablo da yine 0NF'dedir ve ilişkisel yapıda değildir. Farklı ürünlerin kendine ait alanlarda tutulması hâlinde tablo içinde çok fazla Null değer oluşmaktadır. Null değerler tabloda gri renkte işaretlenmiştir. Satış bilgilerinin bu şekilde kaydedilmesi fiziksels saklama alanının boş yere kullanılmasına ve her bir ürünü sorgulamak için sorguda farklı bitişme kullanılmasına sebep olur. Şekil 8.7'de tablo tek satırı sağlamadığı için tablo parçalarının birleştirileceği bölgeler oklar ile işaretlemiştir.

**Şekil 8.7**

Satışların gösterildiği 0NF'de tablo içeriği ve diyagramı

### Satışlar

| Satış | Satış      | Müşteri | Müşteri | Müşteri   | Müşteri      | Ürün   | Ürün      | Birim  | Ted.  | Ted.      | Satış   | Toplam |
|-------|------------|---------|---------|-----------|--------------|--------|-----------|--------|-------|-----------|---------|--------|
| No    | Tarihi     | Id      | Adı     | Soyad     | Adres        | Id1    | Adı1      | Fiyat1 | Yeri1 | Şehir1    | Miktar1 | Fiyat1 |
| 1     | 1.01.2014  | 1       | Aliye   | Berber    | 1. sokak ... | NWTB-1 | Hint Çayı | 1,33   | Depo1 | Eskişehir | 100     | 133    |
| 2     | 1.05.2016  | 2       | Temel   | Giresunlu | 2. sokak ... |        |           |        |       |           |         |        |
| 3     | 1.08.2016  | 1       | Aliye   | Berber    | 1. sokak ... |        |           |        |       |           |         |        |
| 4     | 15.08.2016 | 6       | Feridun | Olcay     | 6. sokak ... |        |           |        |       |           |         |        |

| Satışlar      |
|---------------|
| Satış No      |
| Satış Tarihi  |
| Müşteri Id    |
| Müşteri Adı   |
| Müşteri Soyad |
| Müşteri Adres |
| Ürün Id1      |
| Ürün Adı1     |
| Birim Fiyat1  |
| Ted. Yeri1    |
| Ted. Şehir1   |
| Satış Miktar1 |
| Toplam Fiyat1 |
| Ürün Id2      |
| Ürün Adı2     |
| Birim Fiyat2  |
| Ted. Yeri2    |
| Ted. Şehir2   |
| Satış Miktar2 |
| Toplam Fiyat2 |
| Ürün Id3      |
| Ürün Adı3     |
| Birim Fiyat3  |
| Ted. Yeri3    |
| Ted. Şehir3   |
| Satış Miktar3 |
| Toplam Fiyat3 |
| Ürün Id4      |
| Ürün Adı4     |
| Birim Fiyat4  |
| Ted. Yeri4    |
| Ted. Şehir4   |
| Satış Miktar4 |
| Toplam Fiyat4 |

| Ürün   | Ürün        | Birim  | Ted.  | Ted.      | Satış   | Toplam |
|--------|-------------|--------|-------|-----------|---------|--------|
| Id2    | Adı2        | Fiyat2 | Yeri2 | Şehir2    | Miktar2 | Fiyat2 |
|        |             |        |       |           |         |        |
|        |             |        |       |           |         |        |
|        |             |        |       |           |         |        |
|        |             |        |       |           |         |        |
| NWTO-5 | Zeytin Yağı | 10     | Depo1 | Eskişehir | 8       | 80     |
| NWTO-5 | Zeytin Yağı | 10     | Depo1 | Eskişehir | 65      | 650    |

| Ürün   | Ürün      | Birim  | Ted.  | Ted.   | Satış   | Toplam |
|--------|-----------|--------|-------|--------|---------|--------|
| Id3    | Adı3      | Fiyat3 | Yeri3 | Şehir3 | Miktar3 | Fiyat3 |
|        |           |        |       |        |         |        |
|        |           |        |       |        |         |        |
|        |           |        |       |        |         |        |
|        |           |        |       |        |         |        |
| NWTS-8 | Köri Sosu | 20     | Depo2 | Ankara | 6       | 120    |

| Ürün   | Ürün | Birim  | Ted.  | Ted.   | Satış   | Toplam |
|--------|------|--------|-------|--------|---------|--------|
| Id4    | Adı4 | Fiyat4 | Yeri4 | Şehir4 | Miktar4 | Fiyat4 |
|        |      |        |       |        |         |        |
|        |      |        |       |        |         |        |
|        |      |        |       |        |         |        |
|        |      |        |       |        |         |        |
|        |      |        |       |        |         |        |
| NWTC-3 | Soda | 18     | Depo2 | Ankara | 4,5     | 81     |

## Birinci Normal Form (1NF)

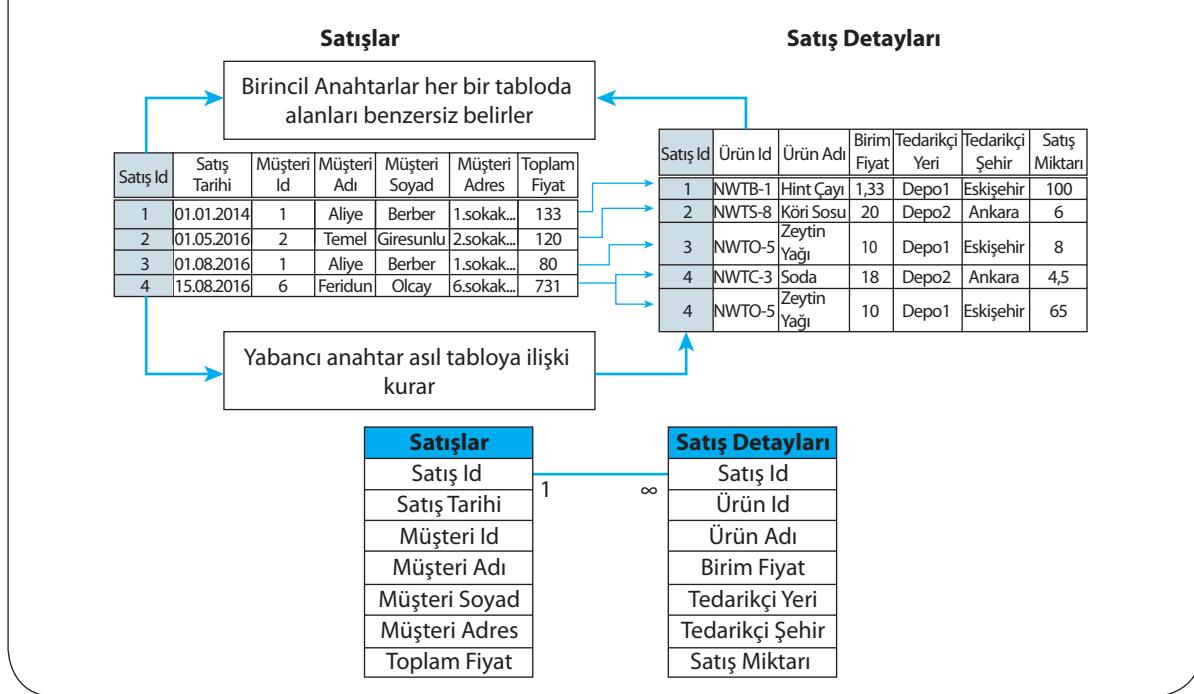
Birinci normal formda çok değerli ve çok parçalı öznitelikler ve tekrar eden gruplar kaldırılır. Veritabanı normalleştirmesinin ilk aşamasında uygulanan kurallar:

- Tekrar eden gruplar kaldırılır.
- Birincil anahtar tanımlanır. Birincil anahtar benzersiz olup tekrarlı değerlere izin vermez.
- Bütün diğer alanlar birincil anahtar ile doğrudan ya da dolaylı benzersiz olarak belirlenebilir.
- Her bir alan atomik değer içermelidir.
- Bir alandaki değerler aynı veritipinde olmalıdır.

1NF'de tekrar eden alanlar yeni tablo oluşturularak kaldırılır. Şekil 8.7'de satıştaki ürünlerin tamamı aynı satırında gösterilmektedir. Bu durumda her bir satış kaydında, satış ile ilgili olmayan ürünü ait alanlar boş kalacak (null değer alır) ve gereksiz yere fiziksel saklama ortamında yer tutacaktır. Bu nedenle 1NF'ye ihtiyaç vardır. Burada Satışlar ve satılan ürünler asıl detay tablolara bire çok ilişki ile bölünmelidir. Bu işlemin sonucu Şekil 8.8'de gösterilmektedir. Satışlar ve satış detayları olarak iki tablo oluşturulmuştur. Birleşik anahtar kaldırılarak her iki tabloda da birincil anahtar tanımlanmıştır. Satışlar tablosundaki birincil anahtar (Satış Id) Satış Detayları tablosuna yabancı anahtar olarak eklenerek bire-çok ilişki oluşturulmuştur. Böylece asıl tabloda (Satışlar) satışa ait bilgilerin ve müşteri bilgilerinin tekrarının önüne geçilmiştir.

**Şekil 8.8**

1NF asıl-detay ilişkili Satışlar ve Satış Detayları tablolarının içerikleri ve diyagramları



1NF sonucunda elde edilen tabloda birincil anahtara kısmi ve geçişli bağımlılıklar olabileceği için ekleme, silme ve güncelleme aykırılıkları ile karşılaşılabilir.

**Ekleme Aykırılığı:** Yeni bir satış bilgisinin eklenmesi için müşteriye ve satılan ürünü ait tedarikçi bilgileri önceden var olsa da tekrar girilmelidir. Benzer şekilde ilişkili tablolarda veri bütünlüğü korunacak şekilde bilgiler her bir tabloya kaydedilmelidir.

**Silme Aykırılığı:** Müşteri bilgisi Satışlar tablosu ile birlikte tutulduğu için ilgili müşteriye ait tüm satışların silinmesi hâlinde müşteri bilgiside kaybedilmektedir. Benzer şekilde kaldırılacak bilgi veri bütünlüğü korunacak şekilde tablolardan doğru sıra ile (cascade) silinmelidir.

**Güncelleme Aykırılığı:** Ürüne ait bir bilgi değiştirilmek istendiğinde birden fazla kaydın güncellenmesi gerekecektir. Bu sorun çok fazla verinin olduğu veritabanlarında performans düşüklüğünü ve tüm alanların güncellenmemesi hâlinde de veri bütünlüğünün bozulmasına sebep olacaktır.

SIRA SİZDE



3

**Şekil 8.9'da öğrencilerin danışmanlarını ve kayıtlı oldukları dersleri gösteren 0NF'de tablo vardır. Bu tabloyu 1NF'ye dönüştürünüz.**

**Şekil 8.9**

Öğrenci ders kayıtlarını gösteren tablo içeriği ve diyagramı

| ÖğrenciNo | Danışman Adı | Danışman Telefon | Danışman Birim | Ders1  | Ders2  | Ders3   |
|-----------|--------------|------------------|----------------|--------|--------|---------|
| 1001      | Kemal        | 3011             | 1521           | Mat101 | Phy103 | Chem104 |
| 1003      | Ahmet        | 3015             | 1512           | İst201 | Num205 | Dif210  |
| 3005      | Ayşe         | 3051             | 1521           | Sys303 | Ağ401  | null    |

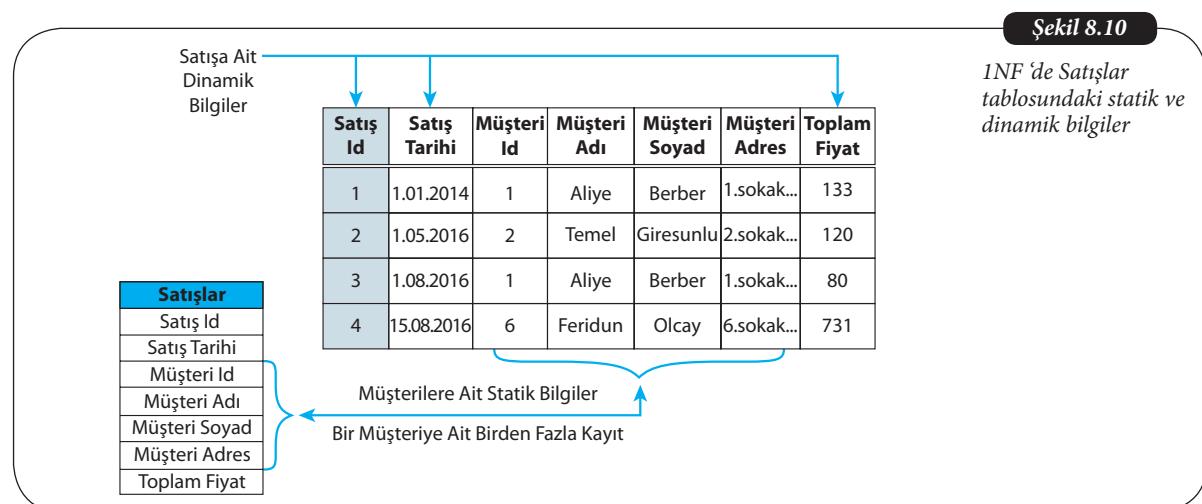
| Öğrenci Ders     |
|------------------|
| ÖğrenciNo        |
| Danışman Adı     |
| Danışman Telefon |
| Danışman Birim   |
| Ders1            |
| Ders2            |
| Ders3            |

## İkinci Normal Form (2NF)

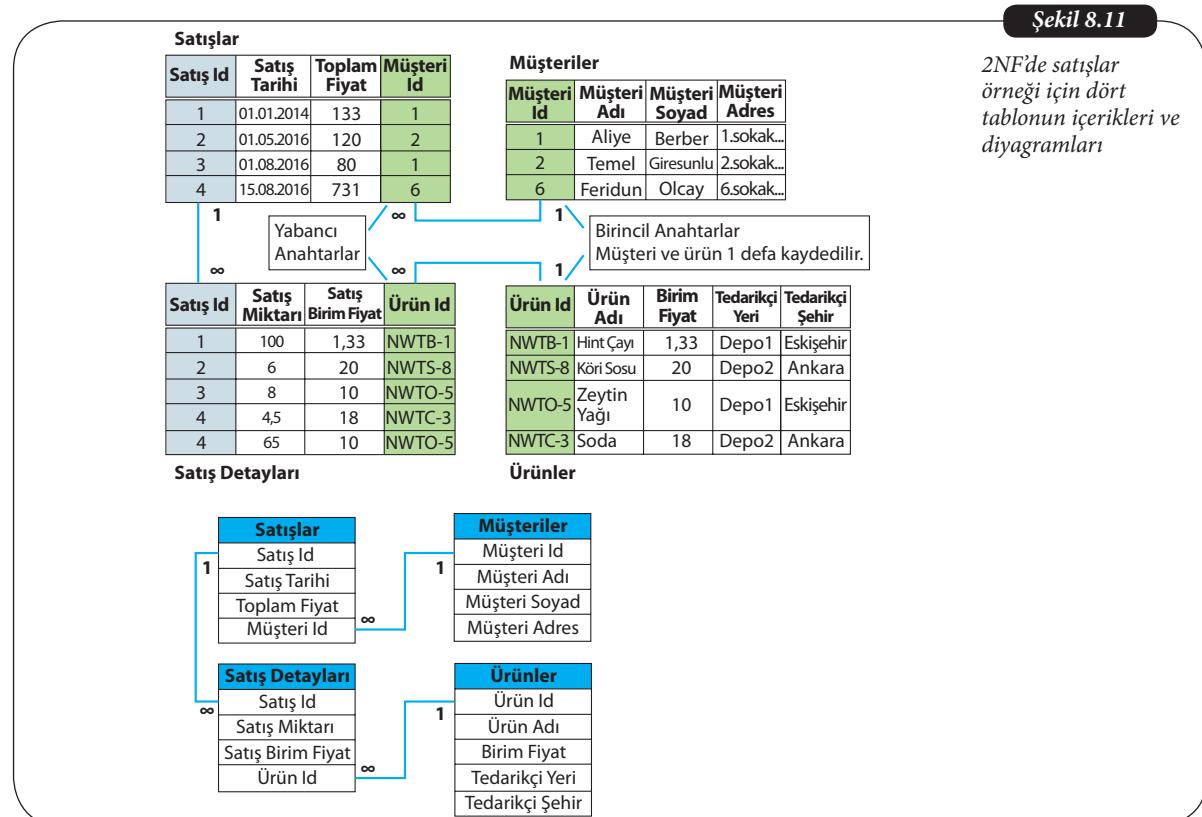
2NF'de tekrar eden değerler yeni tablolara taşınmaktadır. 2NF kuralları aşağıdadır:

- Tablolar 1NF olmalıdır.
- Tüm anahtar olmayan alanlar birincil anahtara tam fonksiyonel bağımlı olmalıdır.
- Kısmi bağımlılıklar kaldırılmalıdır. Önceden de bahsedildiği gibi kısmi bağımlılık fonksiyonel bağımlılığın bir özel durumu olup bir alanın birleşik anahtarın bir parçasına tam bağımlı olması demektir. Kısaca birincil anahtara kısmi bağımlı alanlar yeni bir tablo oluşturarak taşınmalıdır.

Bu aşama sonucunda tablolarda birincil anahtar tek alanda tanımlanmış olur. Şekil 8.10'da 1NF'de kullanılan Satışlar tablosundaki statik ve dinamik alanlar işaretlenmiştir. Her bir satış kaydında sadece dinamik veriler değişmekte statik veriler ise tekrar etmektedir.



2NF'de statik veri 1NF'deki tablodan yeni tabloya çoka bir ilişki ile ayrılmalıdır. Şekil 8.11'de Satışlar ve Satış Detayları tablolardındaki statik bilgiler (müşteri bilgisi ve ürün bilgisi) yeni tablolara ayrılarak tasarımın 2NF'deki durumu verilmiştir. Yeni oluşturulan Müşteriler tablosunda her bir müşteri bir kez yer almaktadır. Böylece yeni tabloda müşterilere ait diğer veriler Müşteri Id birincil anahtar ile benzersiz şekilde belirlenebilmektedir. Müşteriyi satış ile ilişkilendirmek içinde Müşteri Id birincil anahtarı Satışlar tablosuna yabancı anahtar olarak eklenmiştir. Benzer şekilde statik ürün bilgileri için yeni oluşturulan Ürünler tablosundaki birincil anahtar Ürün Id'de Satış Detayları tablosuna yabancı anahtar olarak eklenmiştir. Uzun zaman aralıklarında olsa da Ürünler tablosundaki Birim Fiyat'ın değişebileceği öngörülmüş ve ilave olarak Satış Detayları tablosuna ürünün satışının yapıldığı zamandaki birim fiyatını kaydetmek üzere Satış Birim Fiyat alanı eklenmiştir.



2NF sonucunda elde edilen tablolarda kısmi bağımlılıklar kaldırıldığı için güncelleme aykırılığı çözülür. Ancak birincil anahtara geçişli bağımlılıklar olabileceği için satır eklemeye ve satır silmeye aykırılıkları ile karşılaşılabilir.

**Ekleme Aykırılığı:** Yeni bir ürün bilgisinin eklenmesi için satılan ürüne ait tedarikçi bilgileri önceden var olsa da tekrar girilmelidir.

**Silme Aykırılığı:** Tedarikçi bilgisi ürün ile birlikte tutulduğu için ilgili ürüne ait tüm kayıtların silinmesi hâlinde tedarikçi bilgiside kaybedilmektedir.

SIRA SIZDE



Sıra Sizde 3'te bulduğunuz tabloları 2NF'ye dönüştürünüz.

4

### Üçüncü Normal Form (3NF)

Normalleştirmenin bu aşamasında 2NF'de karşılaşılan aykırılıkları çözmek için geçişli bağımlılıklar da kaldırılır. 3NF kuralları:

- Tablolar 2NF olmalıdır.
- Geçişli bağımlılıklar kaldırılmalıdır. Geçişli bağımlılıkta bir alan birincil anahtar ile ikinci bir alan üzerinden dolaylı olarak belirlenir.
- Ayrılan alanlar için yeni bir tablo oluşturulur.

3NF'de bir tablodaki anahtar olamayan tüm alanlar birincil anahtara bağlı olmak zorundadır. Farklı durumlarda 3NF'nin uygulanması izleyen kesimde örneklerle açıklanacaktır.

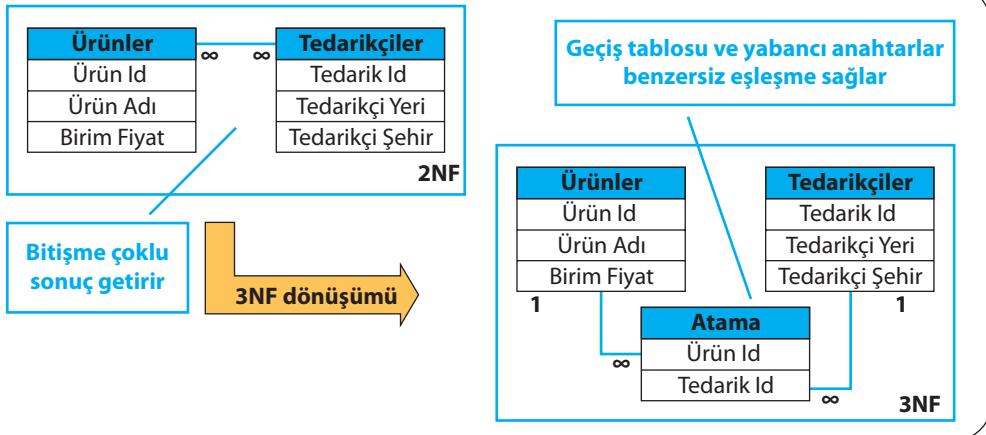
### 3NF'de Çoka Çok İlişki Durumu

İki tablo arasında çoka çok ilişki oluşursa bu tablolarda yapılacak sorgu sonucunda tek bir satır yerine çoklu satır sonuç gelecektir. Bu sorunu gidermek için yeni bir tablo oluşturulur ve çoka çok ilişkili iki tablonun birincil anahtarları bu tabloya yabancı anahtar olarak eklenir.

Şekil 8.12'de Ürünler ve Tedarikçiler arasındaki çoka çok ilişki vardır. Çünkü bir ürün birden fazla tedarikçiden temin edileceği gibi, bir tedarikçiden birden fazla ürün alınıbilir. Bu çoka çok ilişki yeni bir tablo (Atama) aracılığı ile çoka-bir ilişkilere dönüştürülmüştür. Bu dönüşüm sonucunda verinin saklama alanında da küçülme sağlanacaktır.

Şekil 8.12

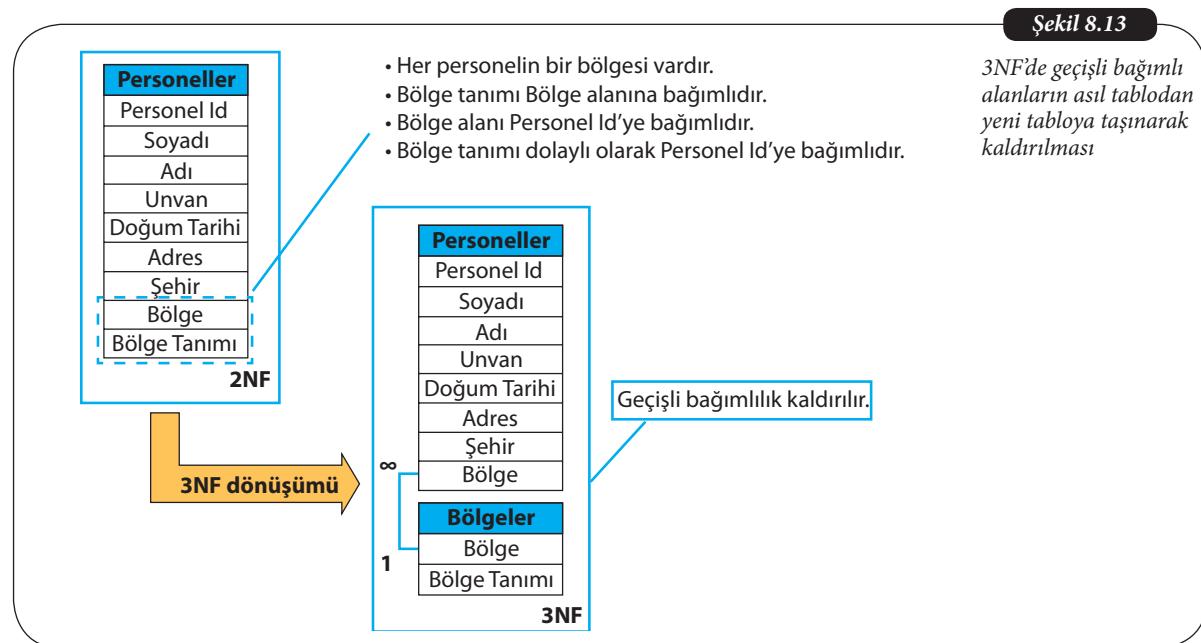
Yeni tablo kullanarak çoka-çok ilişki sorununu giderme



### 3NF'de Geçişli Bağımlılık Durumu

Geçişli bağımlılık durumunun çözümü 3NF'nin en yaygın kullanımındandır. Geçişli bağımlılık birincil anahtara dolaylı olarak bağlı olan alanlardan oluşur. Şekil 8.13'te geçişli bağımlılığın örnek olarak önceki örneklerdeki satışları yapan firmanın personel bilgilerinin tutulduğu tablodaki veriler kullanılmıştır. Bu durumda Bölge Tanımı'nın Personel Id'ye geçişli bağımlılığı açıkça görülmektedir. Bir personel sadece bir bölgede olabilirken bir bölgede birden fazla personel olabilecektir. Geçişli bağımlılığı kaldırmak için yeni bir tablo oluşturulmuş ve yeni tablonun birincil anahtarı asıl tabloya yabancı anahtar olarak eklenmiştir. Böylece asıl tablo ile yeni tablo arasında çoka bir ilişkisi oluşturulmuştur.

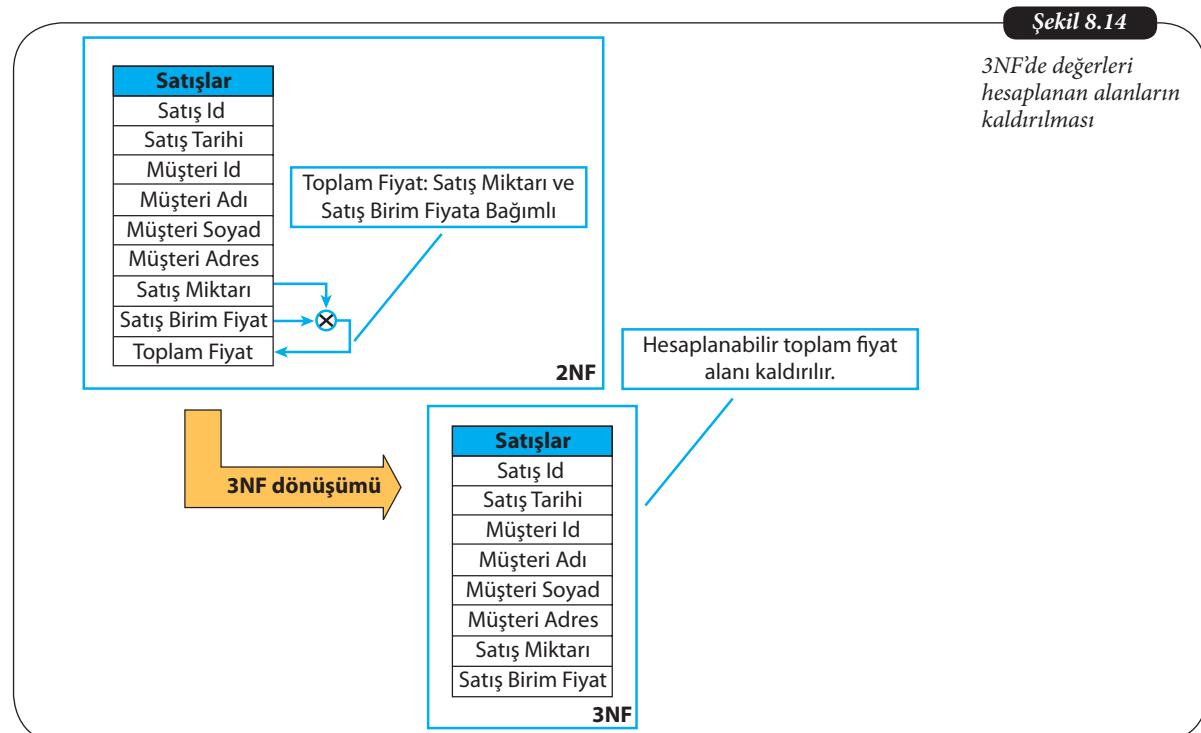
Şekil 8.13



### 3NF'de Hesaplanmış Değerleri Saklayan Alanların Durumu

3NF'de aynı tablodan hesaplanarak bulunan değerler birincil anahtara geçişli olarak bağımlı olduklarıdan kaldırılır. Değer hesaplanabildiği için yeni tablo oluşturmaya gerek kalmaz. Şekil 8.14'te Satışlar tablosundaki Toplam Fiyat alanının değeri Satış Miktarı ve Satış Birim Fiyat alanlarının değerlerinin çarpımı ile hesaplanmaktadır. Örnekte de gösterildiği gibi 3NF'de Satışlar tablosundan Toplam Fiyat alanı kaldırılır. 3NF'ye uygun olmasa da bazı durumlarda performans için oluşturulan görüntülere (view) hesaplanmış değerler eklenebilir.

Şekil 8.14



SIRA SIZDE

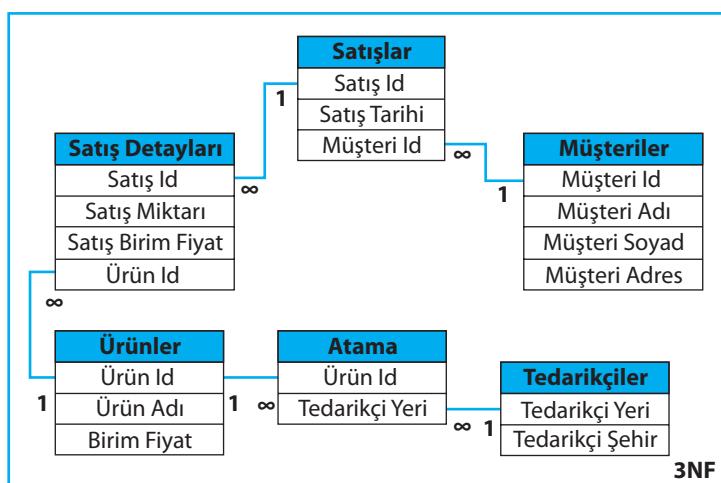
5

Sıra Sizde 4'te bulduğunuz tabloları 3NF'ye dönüştürünüz.

Sonuç olarak Şekil 8.11'de verilen 2NF'deki dört tablonun 3NF'ye dönüştürülmüş hali Şekil 8.15'te verilmiştir. 2NF'de verilen Ürünler tablosundaki Tedarikçi bilgileri Tedarikçiler tablosuna ayrılmıştır. Ürünler ve Tedarikçiler tabloları arasında oluşan çoka çok ilişki Atama geçiş tablosu kullanılarak çoka bir ve bire çok ilişkilere dönüştürülmüştür. İlave olarak Satışlar tablosundaki Toplam Fiyat alanı hesaplanabilir alan olduğu için kaldırılmıştır.

Şekil 8.15

3NF'de satışlar  
örneği için altı  
tablonun içerikleri ve  
diyagramları



### BCNF, 4NF ve 5NF

Günümüzde modern ilişkisel veritabanı modellerinde genelde 3. normal formdan sonrası genelde uygulanmaz. 3NF üzeri normalleştirilmeler veritabanında çok fazla tablo ve ilişki oluşturur. Bunun sonucunda da uygulamada oluşturulacak SQL bitişmeleri çok karmaşık olur. Karmaşık bitişmeler ise sorguların tamamlanma sürelerinin uzamasına sebep olur. Dolayısıyla uygulamada veriye erişimde performans düşüsleri yaşanır. Bu nedenle uygulamacılar tabloları daha küçük parçalara bölmek yerine, 3NF veritabanı modeli ile daha az karmaşık bitişme kullanan SQL komutları kullanır ve olası performans düşüslерinin önüne geçmiş olur.

## Özet



*İyi tasarlanmış bir veritabanının özelliklerini sıralamak*

İlişkisel veritabanları verileri tablolar ve tablolar arasındaki ilişkiler ile modeller. Veritabanı tasarımları tamamlanarak uygulamaya geçirildiğinde veriye SQL olarak adlandırılan veritabanı sorgu dili kullanılarak erişilir. Ancak SQL komutları sonucunda ilgili veriye doğru ve tutarlı olarak erişmek, veri bütünlüğünü korumak ve veritabanı içinde artıklığı minimum yapmak için veritabanı iyi tasarlanmış olmalıdır. İyi bir veritabanı modeli oluşturmanın yöntemi ise normalleştirme uygulamaktır. Eğer veritabanı iyi tasarlanmış ise en az üçüncü normal formda olmalıdır. Böylece veritabanında veri tekrarı ve fonksiyonel bağımlılıklar nedeniyle oluşabilecek aykırılıklar kaldırılır. Uygulama aşamasında daha az saklama yeri kullanması ve daha iyi bir performansta çalışması sağlanmış olur.



### Fonksiyonel bağımlılığı açıklamak

Fonksiyonel bağımlılık veritabanlarının normal formlara uyup uymadığının kontrol edilmesi için kullanılır. Bir veritabanındaki herhangi bir tablo T ve bu tablodaki iki öznitelik A ve B olsun. Eğer A özniteliğinin değeri B özniteliğinin değerini belirliyorsa B özniteliğinin A özniteliğine bağımlı olduğu söylenir. B'nin A'ya fonksiyonel bağımlılığı ok işaretile  $A \rightarrow B$  şeklinde gösterilir.  $A \rightarrow B$ ; A özniteliğinin değerleri B özniteliğinin değerlerini belirler şeklinde okunur. Fonksiyonel bağımlığın çeşitleri olarak tam bağımlılık, kısmi bağımlılık, geçişli bağımlılık, döngüsel bağımlılık ve çok değerli bağımlılık sayılabilir. Normal formların kuralları veritabanı modelinden bu bağımlılıkların kaldırılması veya dönüştürülmesi şeklinde tanımlanır.



*Ekleme, silme ve güncelleme aykırılıklarını tanımlamak*

Veritabanı tasarımdındaki hatalar veya ilişkilerde ortaya çıkan bağımlılıklar aykırılıklara neden olur. Üç tür aykırılık vardır. Bunlar ekleme, silme ve güncelleme aykırılıklarıdır. Aykırılıklar normalleştirme sonucunda düzelir. Veritabanı 3NF'ye dönüştürüldüğünde üç aykırılıkta kaldırılmış olur. Ekleme aykırılığı tabloya veri eklenmek istediği ilave veri veya bilgiye ihtiyaç duyulması anlamına gelir. Silme aykırılığında ise tablodan bir satır silindiğinde faydalı bilgiler de silinebilir. Tablo da gereksiz veri tekrarı olması durumunda güncelleme aykırılığı olur. Güncelleme aykırılığında tekrarlı veri güncellenmek istediği bütün kopyaların aynı şekilde güncellenmesi gereklidir. Aksi hâlde tutarsızlık olur.



### Normalleştirmenin amaçlarını açıklamak

Normalleştirmenin amaçları;

- Veri artıklığını minimum yapmak,
- Aykırılıkları önlemek,
- Veri bütünlüğünü sağlayan kısıtların uygulamasını basitleştirmek,
- Veri işlemeyi (ekleme, güncelleme ve silme) daha basit hâle getirmek,
- Veri tekrarı önlenmiş bir veritabanında ihtiyaç duyulan fiziksel saklama alanı daha az olur. Daha az alanda daha az karmaşık sorgular ile veriye daha hızlı ulaşarak performans artışını sağlamak,
- Gerçek varlık ve ilişkileri daha iyi temsil edecek veritabanı modeli tasarımını gerçekleştirmek ve ileriki geliştirmeler için sağlam bir taban oluşturmak.



### Normal formları ifade etmek

Normal form fonksiyonel bağımlılıklara göre uygulanılan basit kuralların sonucunda oluşturulan/dönüştürülen veritabanı tasarımlının aşamasını belirtir. 1NF, 2NF, 3NF, BCNF, 4NF, 5NF bunlardan yaygın bilinenleridir. Veritabanlarının bu formlara uyması gereklidir. 1NF'de çok değerli öznitelikler atomik değerli özniteliklere dönüştürülür. 2NF'de kısmi bağımlılıklar kaldırılır. 3NF'de geçişli bağımlılıklar kaldırılır. 3NF üzeri Normal Formlarda ise diğer bağımlılık ve aykırılıklar kaldırılır.



### Birinci, ikinci ve üçüncü normal formlara uygun şekilde normalleştirme yapmak

Normal formlara uymayan tablolar normalleştirilir. Normalleştirme tabloların daha küçük tablolara ayrıştırılması işlemidir. Normal form kurallarına göre tablolar daha küçük tablolara bölünür ve tekrar eden değerler bu yeni tablolara taşınır. 1NF'de tekrar eden alanlar detay tabloya taşınır ve her bir alanda atomik ve aynı veri tipinde değerler tutulur. 2NF'de kısmi bağımlılıkları kaldırmak üzere tabloların statik ve dinamik bilgiler ayrılmıştır. Statik bilgiler yeni tabloya taşınır. 3NF'de geçişli bağımlılıkları ortadan kaldırılmak için tablodaki geçişli bağımlı alanlar yeni tabloya taşınır. Bu taşıma sırasında çoka çok ilişki olması durumunda ilişkileri bire çoka dönüştürmek için ilave geçiş tablosu eklenir. Değerli tablodaki alanlardan hesaplanabilen alan geçişli bağımlı olduğundan kaldırılır.

## Kendimizi Sınayalım

**1.** Bir veritabanı tablosunun daha küçük tablolara ayrıstırılma işlemine ne ad verilir?

- a. Aykırılık
- b. Veri silme
- c. Normalleştirme
- d. Veri dağıtma
- e. Veritabanı yönetimi

**2.** Aşağıdaki kurallardan hangisi fonksiyonel bağımlılık çeşitlerinden biri **değildir**?

- a. Tam
- b. Kısmi
- c. Geçişli
- d. Bitişmeli
- e. Döngüsel

**3.**  $X \rightarrow Y$  ve  $Y \rightarrow Z$  fonksiyonel bağımlılıkları verildiğinde,  $Z$  öznitelığının  $Y$  özniteligi vasıtasiyla  $X$  özniteligine bağımlı olduğunun gösterilmesine ne ad verilir?

- a. Geçişli bağımlılık
- b. Kısmi bağımlılık
- c. Döngüsel bağımlılık
- d. Tam bağımlılık
- e. Çok Değerli bağımlılık

**4.** 1NF'deki tablodan bir satır silindiğinde faydalı veriler de siliniyorsa, bu aykırılığa ne ad verilir?

- a. Ekleme
- b. Silme
- c. Güncelleme
- d. Döngüsel
- e. Dönüşürme

**Şekil 8.16:** Öğrenci ve Kayıtlı olduğu dersler tablosu. (Öğrenci No birincil anahtardır)

| Öğrenci No | Bölüm Kodu | Bölüm Adı | Ders Kodu       | Kayıt Yıl |
|------------|------------|-----------|-----------------|-----------|
| 1001       | 15211      | İşletme   | Mat101, İst103  | 2015      |
| 3008       | 15131      | Endüstri  | Mat101, Chem104 | 2015      |
| 3010       | 15211      | İşletme   | Edb205, His203  | 2016      |

Soru 5-7 Şekil 8.16'ya göre cevaplanacaktır.

**5.** Şekil 8.16'da verilen tablo 1NF kuralı ile tek tabloya yapılandırıldığında kaç satırlık tablo oluşur?

- a. 2
- b. 3
- c. 4
- d. 5
- e. 6

**6.** Şekil 8.16'da verilen tablonun 1NF kuralına uyum sağlama için aşağıdaki alanlardan hangisi dönüştürülür?

- a. Sadece Ders Kodu
- b. Ders Kodu ve Kayıt Yıl
- c. Öğrenci No ve Kayıt Yıl
- d. Sadece Öğrenci No
- e. Bölüm Kodu ve Kayıt Yılı

**7.** Şekil 8.16'da verilen tablonun 1NF kuralına uyum sağlama için dönüştürülecek alandaki sorun aşağıdakilerden hangisidir?

- a. Ders Kodu alanında çok değerli veri var.
- b. Kayıt Yıl alanında sadece 2 adet değer var.
- c. Bölüm Adı alanında sadece 2 adet değer var.
- d. Bölüm Kodu alanında sadece 2 adet değer var.
- e. Bölüm Adı Öğrenci No'ya geçişli bağımlıdır.

**8.** Normalize edilmemiş (0NF) bir tablo verildiğinde normalleştirme kuralları hangi sıra ile uygulanmalıdır?

- a. BCNF, 3NF, 2NF, 1NF
- b. 1NF, 2NF, 3NF, BCNF
- c. BCNF, 1NF, 2NF, 3NF
- d. 3NF, 2NF, 1NF, BCNF
- e. Normal Formların uygulanma sırası önemli değildir.

### Kendimizi Sınayalım Yanıt Anahtarı

**Şekil 8.17:** Öğretim elemanlarının çalıştığı bölüm ve kampüs bilgileri tablosu. Sicil No birincil anahtardır.

| Öğretim Elemanları |
|--------------------|
| Sicil No           |
| Ad                 |
| Bölüm              |
| Kampüs             |

Soru 9-10 Şekil 8.17'ye göre cevaplanacaktır.

- 9.** Yukarıda verilen tabloda alanlar arasındaki geçişli bağımlılığı aşağıdakilerden hangisidir?
- Sicil No → Bölüm, Bölüm → Kampüs
  - Ad → Bölüm, Bölüm → Kampüs
  - Sicil No → Ad, Bölüm → Kampüs
  - Sicil No → Ad, Sicil No → Bölüm, Sicil No → Kampüs
  - Sicil No → Ad, Ad → Kampüs

- 10.** Tabloya 3NF kuralı uygulandığında oluşan tabloların alanları aşağıdakilerden hangisinde birlikte ve doğru olarak verilmiştir? (Dikkat: öğretim elemanın sadece bir bölümde çalışmakta olduğunu varsayıyınız)

- |                                |                                    |
|--------------------------------|------------------------------------|
| a. Tablo1: Sicil No, Ad        | Tablo2: Bölüm,<br>Kampüs           |
| b. Tablo1: Sicil No, Ad, Bölüm | Tablo2: Bölüm,<br>Kampüs           |
| c. Tablo1: Sicil No, Ad        | Tablo2: Sicil No,<br>Bölüm, Kampüs |
| d. Tablo1: Sicil No, Ad, Bölüm | Tablo2: Sicil No,<br>Kampüs        |
| e. Tablo1: Sicil No, Ad, Bölüm | Tablo2: Sicil No,<br>Bölüm, Kampüs |

- Yanıtınız yanlış ise “Normalleştirme” konusunu yeniden gözden geçiriniz.
- Yanıtınız yanlış ise “Fonksiyonel Bağımlılık” konusunu yeniden gözden geçiriniz.
- Yanıtınız yanlış ise “Fonksiyonel Bağımlılık” konusunu yeniden gözden geçiriniz.
- Yanıtınız yanlış ise “Aykırılıklar” konusunu yeniden gözden geçiriniz.
- Yanıtınız yanlış ise “Birinci Normal Form” konusunu yeniden gözden geçiriniz.
- Yanıtınız yanlış ise “Birinci Normal Form” konusunu yeniden gözden geçiriniz.
- Yanıtınız yanlış ise “Birinci Normal Form” konusunu yeniden gözden geçiriniz.
- Yanıtınız yanlış ise “Normal Formlar” konusunu yeniden gözden geçiriniz.
- Yanıtınız yanlış ise “Üçüncü Normal Form” konusunu yeniden gözden geçiriniz.
- Yanıtınız yanlış ise “Üçüncü Normal Form” konusunu yeniden gözden geçiriniz.

## Sıra Sizde Yanıt Anahtarları

### Sıra Sizde 1

| <b>Id</b> | <b>Yıl</b> | <b>Adı</b> | <b>Mahalle</b> | <b>Telefon Numarası</b> | <b>Kod</b> | <b>Ders Adı</b>       | <b>Kredi</b> |
|-----------|------------|------------|----------------|-------------------------|------------|-----------------------|--------------|
| 1         | 2015       | Uğur       | Yeşiltepe      | 3015                    | 101        | Matematik             | 5            |
| 2         | 2015       | Kemal      | Gültepe        | 3016                    | 201        | Veritabanı Sistemleri | 4            |
| 3         | 2015       | Metin      | Esentepe       | 3045                    | 303        | Veri Madenciliği      | 3            |
| 3         | 2016       | Metin      | Esentepe       | 3045                    | 402        | Ağ Güvenliği          | 3            |
| 5         | 2016       | Kemal      | Tepebaşı       | 3051                    | 201        | Veritabanı Sistemleri | 4            |

Id ve Kod birleşik anahtardır.

Kısmı Bağımlılık:

Id → Yıl, Adı, Mahalle, Telefon Numarası

Kod → Ders Adı, Kredi

Geçişli Bağımlılık:

Id → Adı ve Adı → Mahalle, Telefon Numarası dolayısıyla; Id → Mahalle, Telefon Numarası

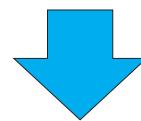
### Sıra Sizde 2

Öğretim elemanları Uğur ve Metin'e ait kayıtlar silinmek istendiğinde (Id'si 1 veya 3 no.lu kayıtlar) silme aykırılığı olur. Çünkü bu durumda bölümde verilmekte olan Matematik, Veri Madenciliği ve Ağ Güvenliği derslerine ait kayıtlar da kaybolacaktır.

### Sıra Sizde 3

Tablolar sadece iki boyutlu olmalıdır. Bu nedenle bir öğrencinin birden çok ders alabileceği (ve aynı dersi birden çok öğrenci alabilir) şekilde tablo tekrar yapılandırılmıştır. Bu nedenle ÖğrenciNo ve Ders birleşik anahtar olarak tanımlanır. Sonuç olarak 1NF'de dönüştürülmüş tablo;

| <b>Öğrenci No</b> | <b>Danışman Adı</b> | <b>Danışman Telefon</b> | <b>Danışman Birim</b> | <b>Ders1</b> | <b>Ders2</b> | <b>Ders3</b> |
|-------------------|---------------------|-------------------------|-----------------------|--------------|--------------|--------------|
| 1001              | Kemal               | 3011                    | 1521                  | Mat101       | Phy103       | Chem104      |
| 1003              | Ahmet               | 3015                    | 1512                  | Ist201       | Num205       | Dif210       |
| 3005              | Ayşe                | 3051                    | 1521                  | Sys303       | Ağ401        | null         |



### Öğrenci Ders

| <b>Öğrenci No</b> | <b>Danışman Adı</b> | <b>Danışman Telefon</b> | <b>Danışman Birim</b> | <b>Ders</b> |
|-------------------|---------------------|-------------------------|-----------------------|-------------|
| 1001              | Kemal               | 3011                    | 1521                  | Mat101      |
| 1001              | Kemal               | 3011                    | 1521                  | Phy103      |
| 1001              | Kemal               | 3011                    | 1521                  | Chem104     |
| 1003              | Ahmet               | 3015                    | 1512                  | Ist201      |
| 1003              | Ahmet               | 3015                    | 1512                  | Num205      |
| 1003              | Ahmet               | 3015                    | 1512                  | Dif210      |
| 3005              | Ayşe                | 3051                    | 1521                  | Sys303      |
| 3005              | Ayşe                | 3051                    | 1521                  | Ağ401       |

### Öğrenci Ders

Öğrenci No

Danışman Adı

Danışman Telefon

Danışman Birim

Ders

**Sıra Sizde 4**

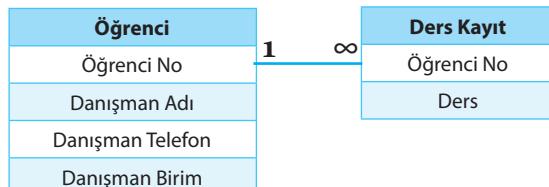
Sıra Sizde 3'teki 1NF'deki tabloda her bir öğrenci için birden fazla ders kaydı olduğu için öğrenciye ait (Danışman bilgisi) statik bilgiler tekrar etmektedir. Bu da saklama alanında gereksiz yer tutar ve ekleme, silme ve güncelleme aykırılığına sebep olur. Bu nedenle tablodaki dinamik bilgi yeni oluşturulacak detay tablosuna taşınmalıdır. Öğrenci No Öğrenci tablosunda birincil anahtar olarak tanımlanır. Dinamik bilgiler için Ders Kayıt tablosu oluşturulur. Öğrenci No Ders Kayıt tablosuna yabancı anahtar olarak eklenir.

**Öğrenci**

| Öğrenci No | Danışman Adı | Danışman Telefon | Danışman Birim |
|------------|--------------|------------------|----------------|
| 1001       | Kemal        | 3011             | 1521           |
| 1003       | Ahmet        | 3015             | 1512           |
| 3005       | Ayşe         | 3051             | 1521           |

**Ders Kayıt**

| Öğrenci No | Ders    |
|------------|---------|
| 1001       | Mat101  |
| 1001       | Phy103  |
| 1001       | Chem104 |
| 1003       | İst201  |
| 1003       | Num205  |
| 1003       | Dif210  |
| 3005       | Sys303  |
| 3005       | Ağ401   |

**Sıra Sizde 5****Öğrenci**

| Öğrenci No | Danışman Adı |
|------------|--------------|
| 1001       | Kemal        |
| 1003       | Ahmet        |
| 3005       | Ayşe         |

**Ders Kayıt**

| Öğrenci No | Ders    |
|------------|---------|
| 1001       | Mat101  |
| 1001       | Phy103  |
| 1001       | Chem104 |
| 1003       | İst201  |
| 1003       | Num205  |
| 1003       | Dif210  |
| 3005       | Sys303  |
| 3005       | Ağ401   |

**Danışmanlar**

| Danışman Adı | Danışman Telefon | Danışman Birim |
|--------------|------------------|----------------|
| Kemal        | 3011             | 1521           |
| Ahmet        | 3015             | 1512           |
| Ayşe         | 3051             | 1521           |

| Danışmanlar      |
|------------------|
| Danışman Adı     |
| Danışman Telefon |
| Danışman Birim   |

| Öğrenci      | Ders Kayıt |
|--------------|------------|
| Öğrenci No   | Öğrenci No |
| Danışman Adı | Ders       |

3NF'de birincil anahtara bağlı olmayan tüm veriler farklı tabloya taşınır. Her öğrencinin bir danışmanı olduğunu ancak bir danışmanın birden fazla öğrencisi olduğunu varsayılmıştır. 2NF'de Danışman Telefon ve Danışman Birim alanları Danışman Adı üzerinden Öğrenci No'ya bağımlıdır. 3NF'de danışmanlara ait bilgiler yeni tabloya taşınarak bu geçişli bağımlılık kaldırılmıştır.

## Yararlanılan ve Başvurulabilecek Kaynaklar

- Garcia-Molina H., Ullman J.D., Widom J. (2008). **Database Systems: The Complete Book, 2nd Edition.** New Jersey, Pearson Education Inc.
- Gavin Powell (2006). **Beginning Database Design,** Indianapolis, Wiley Publishing, Inc.
- Ramakrishnan R., Gehrke J. (2003). **Database Management Systems,** 3rd Edition. New York, McGraw-Hill Companies Inc.
- Özseven, T. (2011). **Veritabanı Yönetim Sistemleri,** Trabzon: Murat Han Yayınevi.