

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.

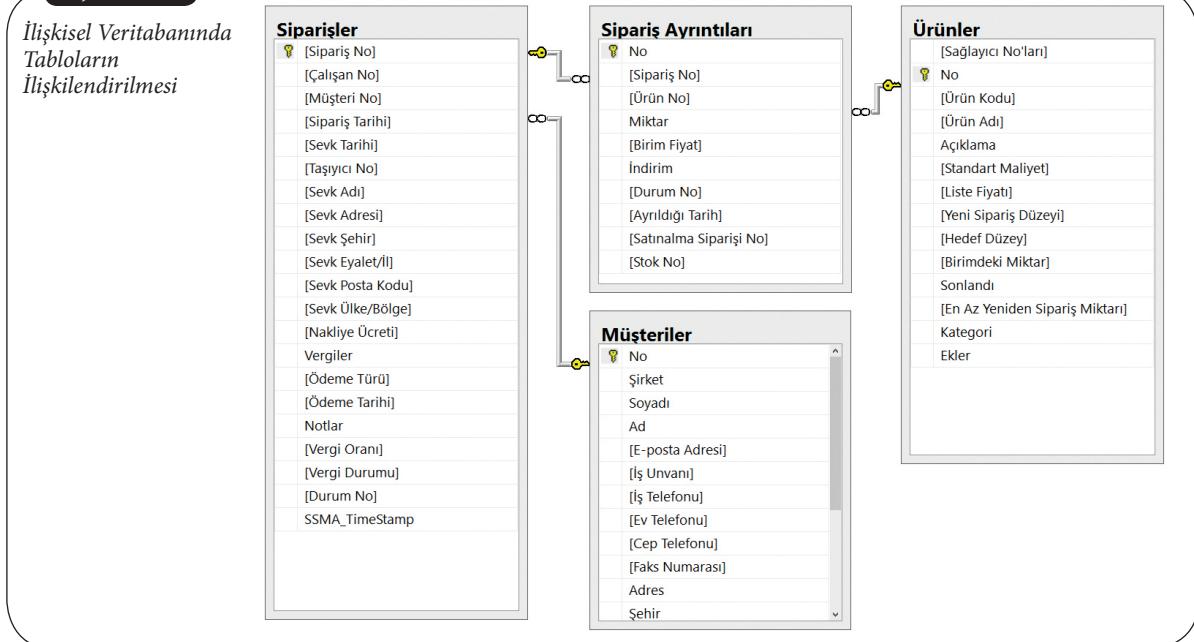
İlişkisel modelde tablolar arası ilişkiler bire-bir, bire-çok ve çoka-çok olarak sınıflanır (Tablolar arasındaki ilişkiler ünite 2'de detaylı olarak anlatılmıştır). İki tablo arasındaki bir ilişki bir ortak alanın eşleşen değerleri üzerinden direk olarak kurulur. Örnek olarak, Şekil 3.1'de [*Siparişler*] 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 SIZDE



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 örneklenecektir.

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. $\pi_{\text{isim}, \text{Adres}}(R)$ 'nın 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. $\sigma_{\text{Cinsiyet} = F}(R)$ 'nın 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.

Sekil 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	1	2	3	2	3	12
15	4	8	2	3	14	1	2	3	2	3	14
19	4	8	4	8	21	1	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.

Sekil 3.4

A	T.B	T.C	V.B	V.C	D
1	2	3	2	3	12
1	2	3	2	3	14
1	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

$T \bowtie A < D V$ Şartlı Bitişme İşlemi

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 (TxV) getirdiği kayıtlar Şekil 3.5'te gösterilmektedir.

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 3.5

A	B	C	D
1	2	3	12
1	2	3	14
15	4	8	21
19	4	8	21

$T \bowtie V$ Doğal Bitişme İşlemi

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
1	2	3	2	3	12
1	2	3	2	3	14
1	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?

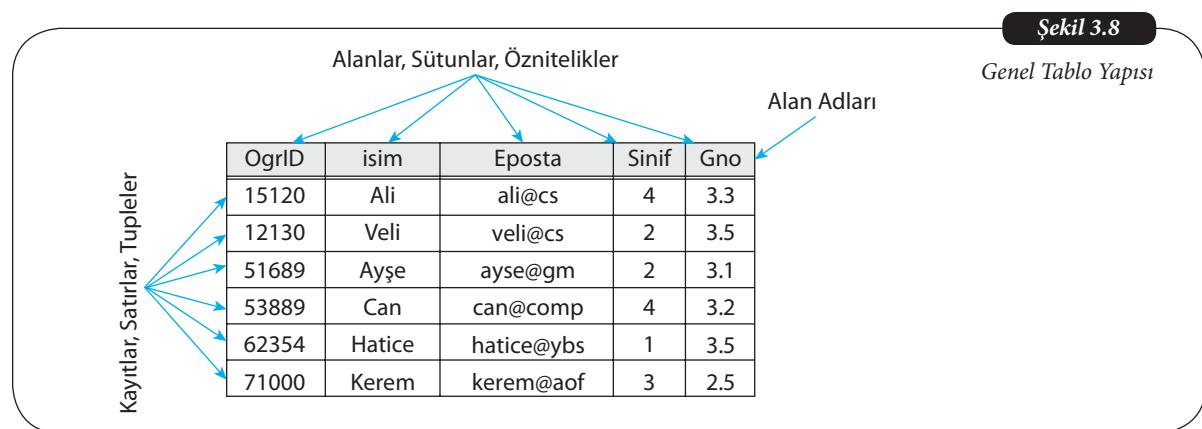
İLİŞKİSEL VERİTABANI NESNELERİ VE KAVRAMLAR

İVTYS ilişkisel veritabanı modelini esas alan yönetim yazılımıdır. İVTYS ilişkisel veritabanının yönetimini, veritabanı ile iletişimini sağlayan arayüzlerden oluşur. 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 tasarlannmış 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 tasarlannmış 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 ayırtmak 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 tipi türleri Basit, Karmaşık, Özelleştirilmiş olarak gruplanarak anlatılacaktır.

Basit Veri Tipleri

Bu veri tiplerinde tek bir değer üzerinde bir örüntü 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 geleneksel 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

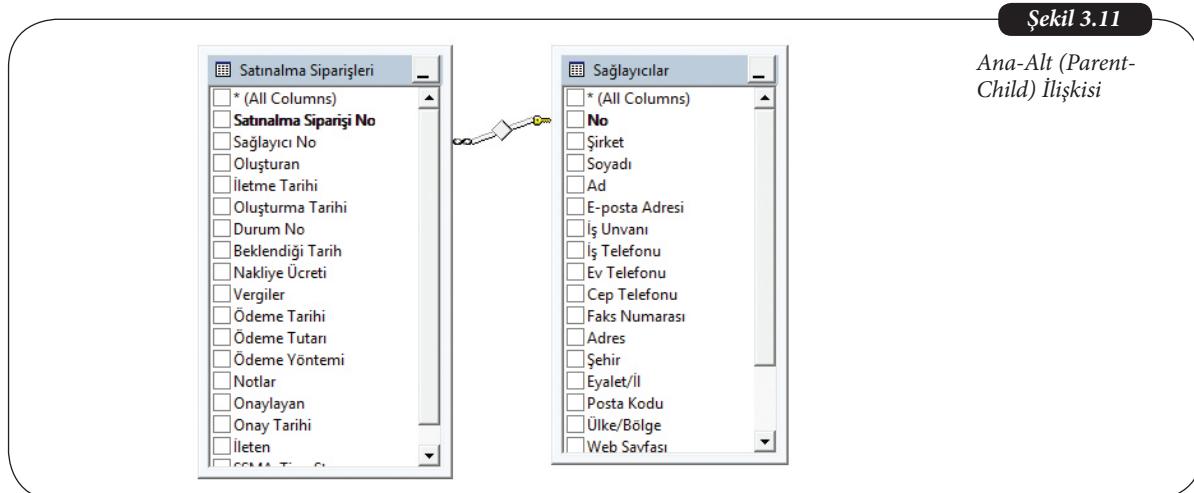
Veritabanı tablosunun alanlarının mutlaka dolu olması gerekiğinde boş bırakılmaması için uygulanan bir zorlamadır. Eğer bir alanın özelliği NOT NULL olarak belirlenmiş ise tabloya yeni bir kayıt eklenirken bu alana ait değerin boş bırakılmasına izin verilmez. Kullanıcı bir hata mesajı ile bilgilendirilir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



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 kayıta erişmek için yeterli değildir. Aynı bireyin oluşturduğu birçok satınalma siparişi olabilir. Birincil anahtar tek bir alan ile tanımlanabileceğ 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 [Satınalma Siparişleri] ve [Çalışanlar] alanlarına CHECK kısıtı tanımlanarak ülke değeri olarak Türkiye girildiğinde [Posta Kodu] alanına da sayısal değer girilmesi zorlanabilir.

CHECK ve yabancı anahtar kısıtları tanımlanan alanına 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 [Satınalma Siparişleri] ve [Çalışanlar] tablolarında birincil ve yabancı anahtarlar hangileri olabilir?



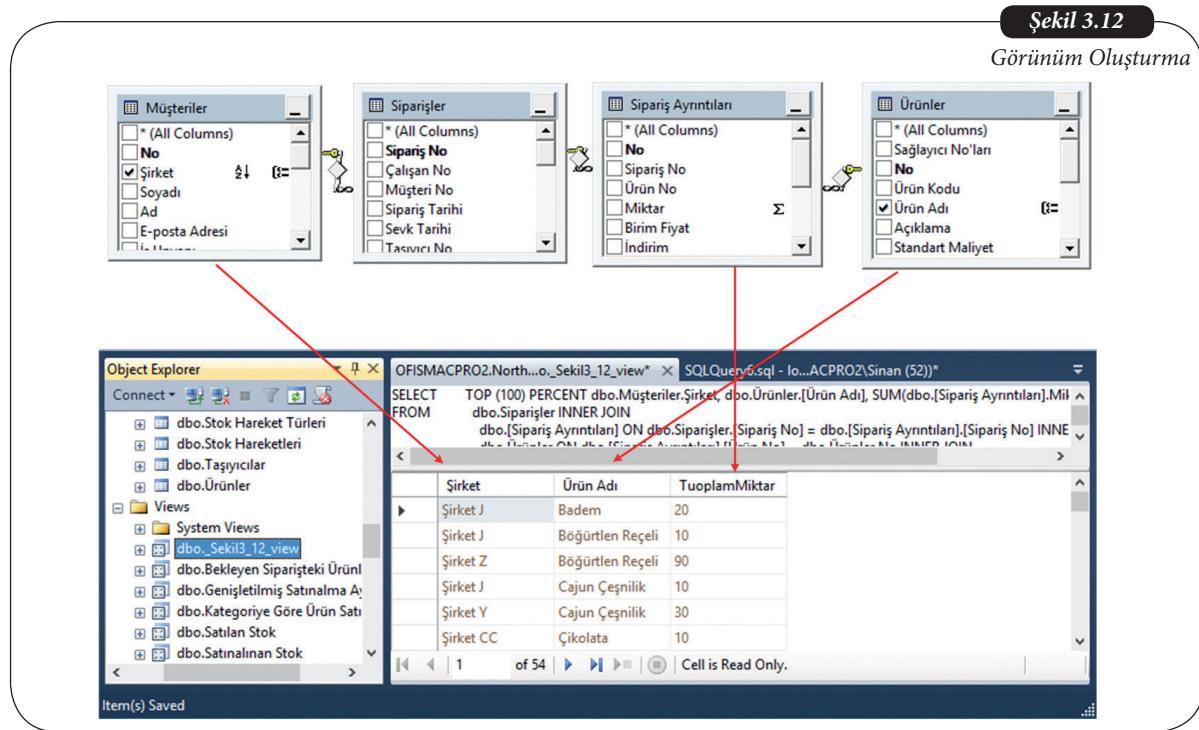
Görünümler (Views)

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



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 alımı yapmış 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 kitabındaki indeks bölümünden nasıl doğrudan ilgili sayfa numarasını bulup 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 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 komutlarının 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://goo.gl/K839el> adresinde yer alan Northwind.rar dosyasına ulaşabilirsiniz.



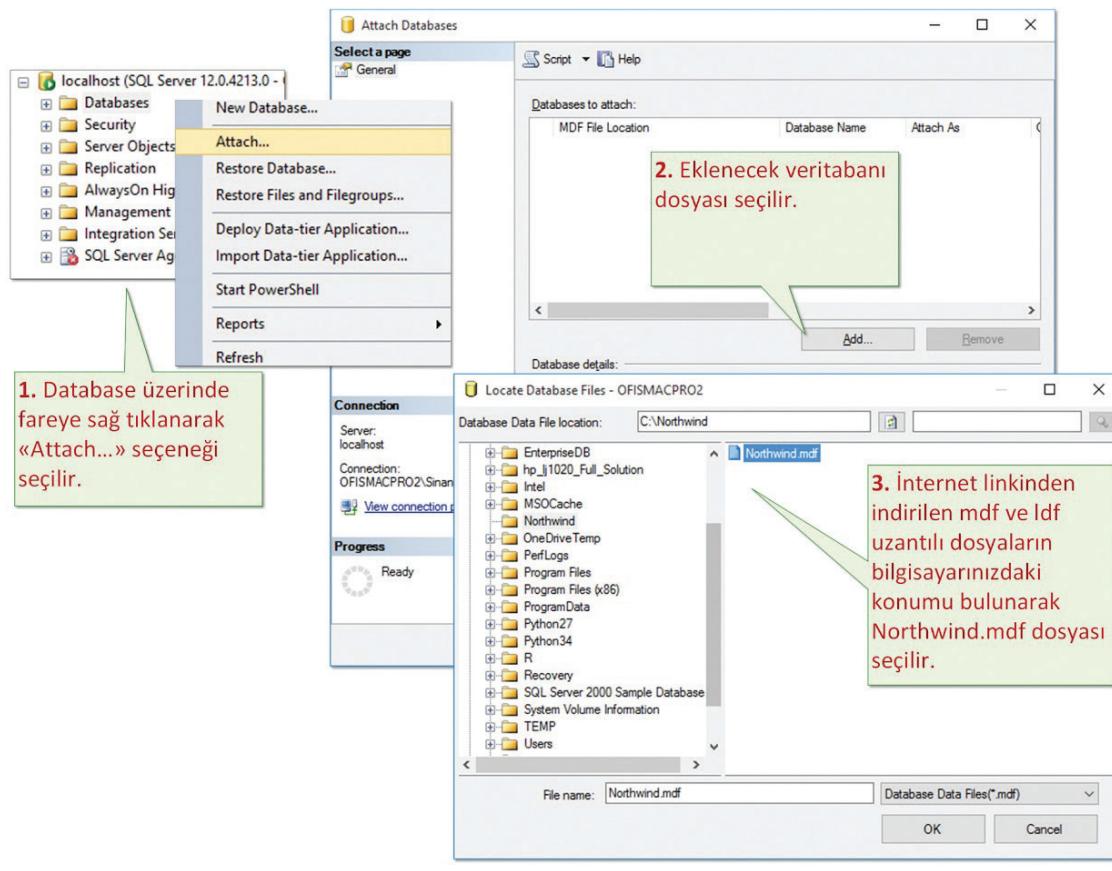
INTERNET

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

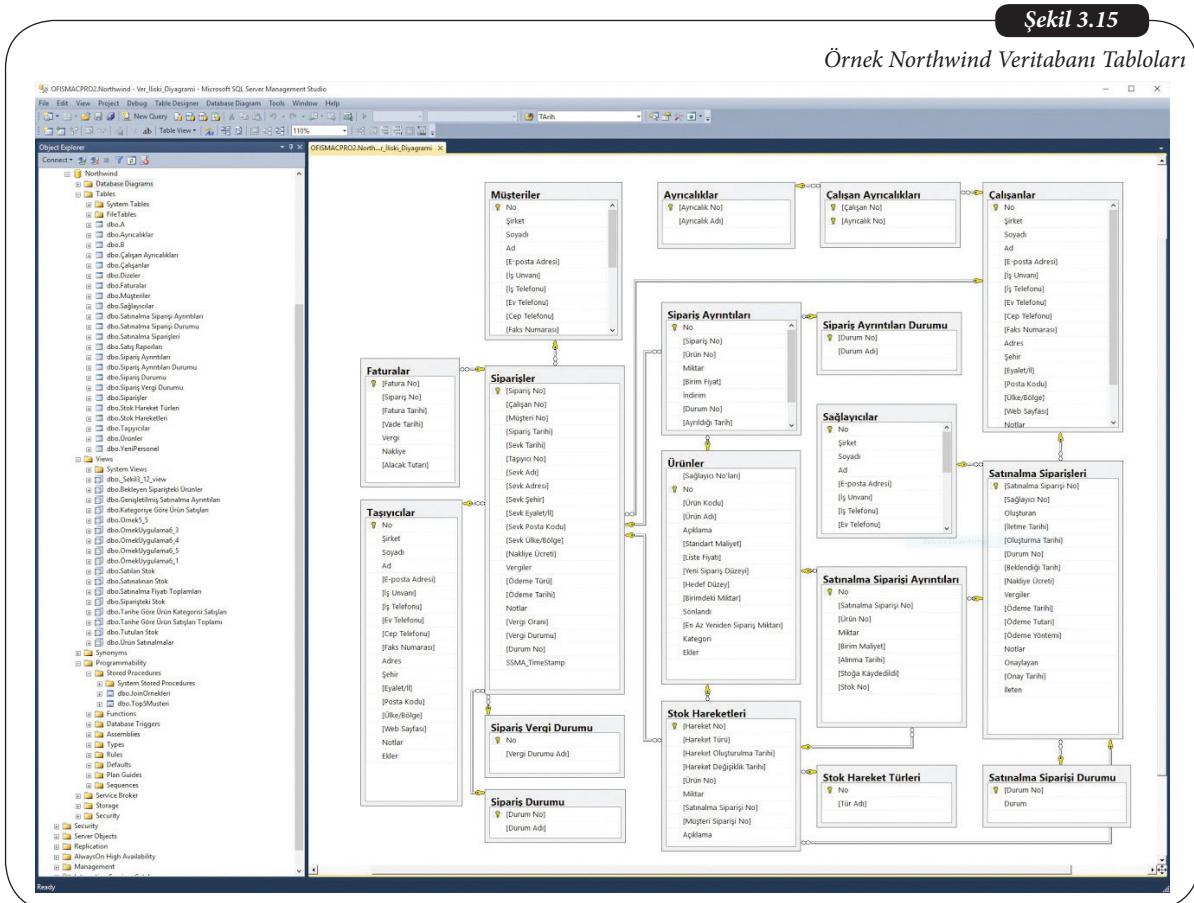
MS SQL Server Management Studio'da SQL Betik ile Örnek Northwind Veritabanı Kurulumu

```

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
)

```

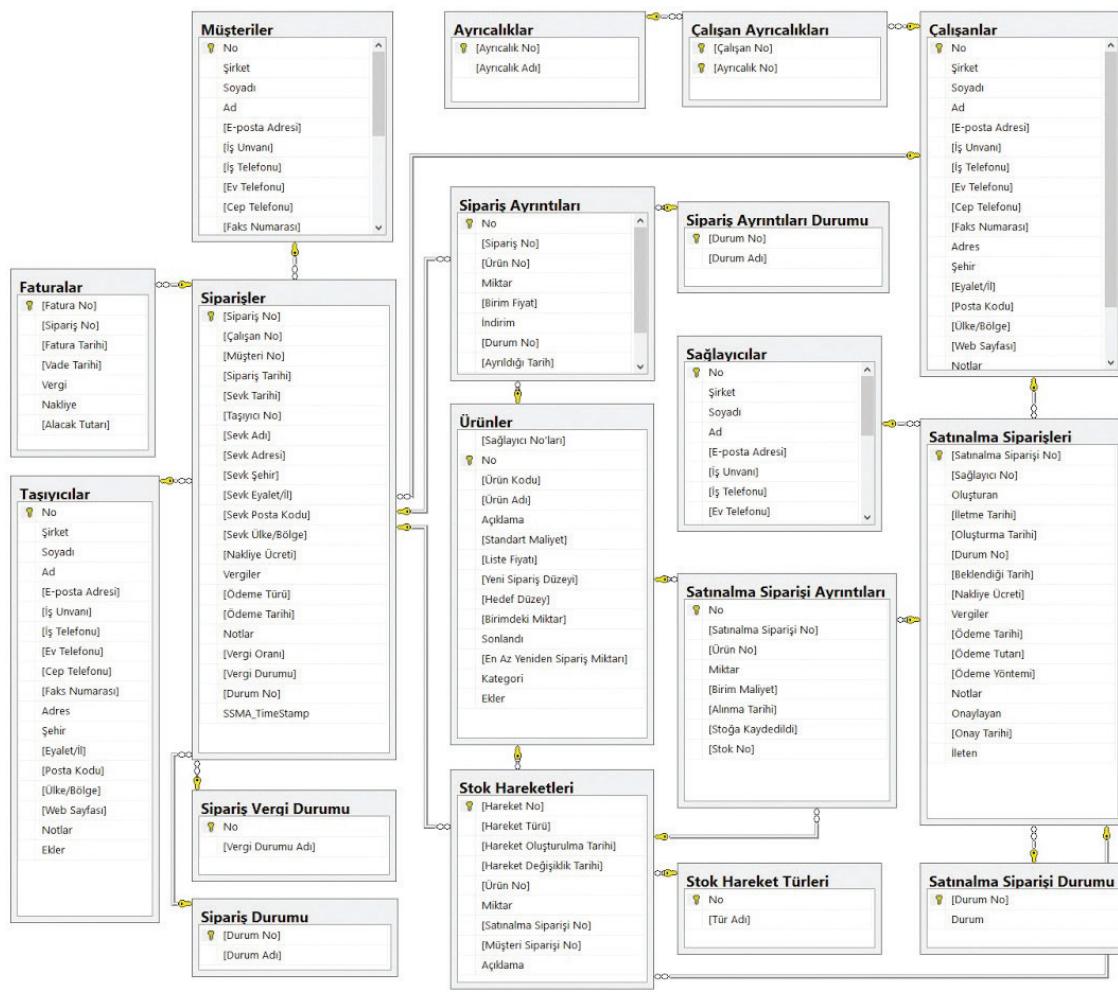
Northwind veritabanı eklendikten sonra MS SQL Server Management Studio uygulamasında veritabanının nesnelerinin (tablolar, görünümler, diyagramlar vb.) görünümü Şekil 3.15'te verilmiştir.



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

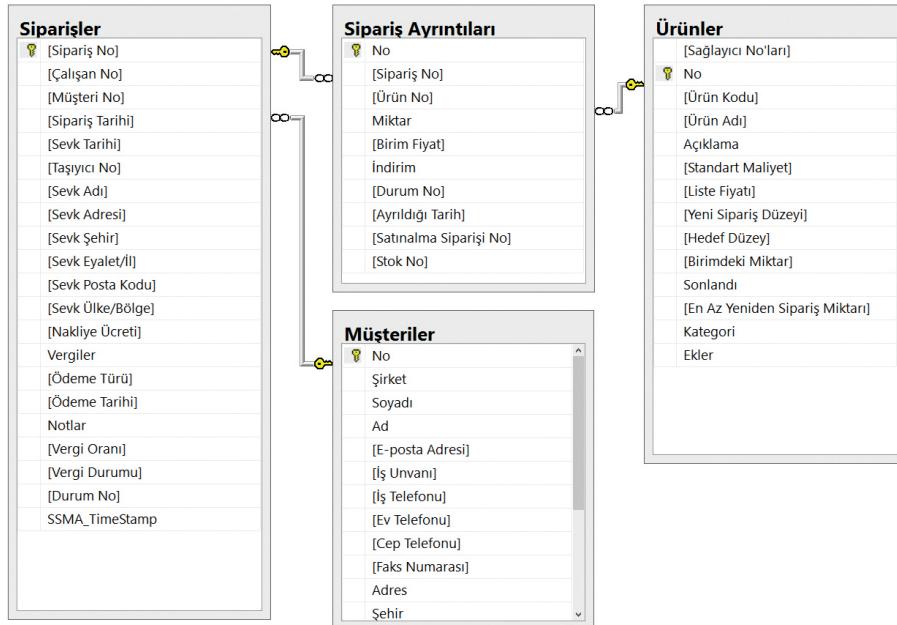
İlişkisel veritabanı işleminde yukarıda verilen ifadelerin sonucu aşağıdakilerden hangisidir?

- 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 Yanıt Anahtarları

Sıra Sizde 1

Hangi ürünlerin hangi şehre sevk edildiğine ulaşmak için *[Ürün Adı]* alanının bulunduğu *[Ürünler]* tablosu ile *[Sevk Şehir]* alanın bulunduğu *[Siparişler]* tablosunun ilişkisi bulunmalıdır. Bu iki tablo arasında ilişki *[Sipariş Ayrıntı]* tablosudur. *[Sipariş Ayrıntı]* tablosu her bir siparişte hangi ürün den 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 tablo dan A=1 ile seçim yapılrsa aşağıdaki kayıtlar gelir.

Sıra Sizde 3

Tablonun çok parçalı *[PostakoduSehir]* alanı *[Postakodu]* ve *[Sehir]* olmak üzere iki ayrı alan hâlinde bölünerek tekrar tasarılanmalıdır. 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
<i>[Satınalma Siparişleri]</i>	<i>[Satınalma Siparişi No]</i>	<i>[Oluşturan], [Sağlayıcı No]</i>
<i>[Çalışanlar]</i>	<i>[No]</i>	-

Sıra Sizde 5

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

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.

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

