* Ortaya çıkan Tablo 3'e dahil edilmek için, paylaşılmayan sütundaki (CUS\_CODE) bir

değer Tablo 2'deki her değerle ilişkilendirilmelidir.

* 123456, 234567 ve 567890 numaralı ürünlerin tamamıyla ilişkili tek müşteriler 10030 ve 12550

numaralı müşterilerdir.

**Not**

DIVIDE operatörü bölme sembolü 4 ile gösterilir. R ve S olmak üzere iki bağıntı verildiğinde, bunların BÖLÜNMESİ r 4 s olarak yazılacaktır.

#### veri sözlüğü

Veriler hakkındaki meta verileri depolayan bir DBMS bileşenidir. Böylece, veri sözlüğü veri tanımını da içerir

özellikleri ve ilişkileri gibi. Bir veri

Sözlük, VTYSnin dışında bulunan verileri de içerebilir. *Bilgi kaynağı sözlüğü* olarak da bilinir. Ayrıca bkz. *aktif veri sözlüğü*, *meta veri* ve *pasif veri sözlüğü*.

**Not**

Tablo 3.6'daki veri sözlüğü, varlıkların, özniteliklerin ve ilişkilerin *insan* bakış açısına bir örnektir. Bu veri sözlüğünün amacı, veritabanı tasarım ve uygulama ekiplerinin tüm üyelerinin aynı tablo ve öznitelik adlarını ve özelliklerini kullanmasını sağlamaktır. VTYS'nin dahili olarak depolanan veri sözlüğü, ilişki türleri, varlık ve referans bütünlüğü kontrolleri ve uygulamaları ile dizin türleri ve bileşenleri hakkında ek bilgiler içerir. Bu ek bilgiler veritabanı uygulama aşamasında oluşturulur.

# 3-5 Veri Sözlüğü ve Sistem Kataloğu

**Veri sözlüğü**, kullanıcı ve tasarımcı tarafından oluşturulan veritabanındaki tüm tabloların ayrıntılı bir tanımını sağlar. Böylece veri sözlüğü, sistemdeki her tablo için en azından tüm öznitelik adlarını ve özelliklerini içerir. Kısacası, veri sözlüğü metadata- veri hakkında veri içerir. Şekil 3.3'te sunulan küçük veritabanını kullanarak, veri sözlüğünü Tablo 3.6'da gösterildiği gibi hayal edebilirsiniz.

#### si̇stem kataloğu

Bir veritabanındaki tüm nesneleri tanımlayan ayrıntılı bir sistem veri sözlüğü.

#### eşsesli

Farklı nitelikleri etiketlemek için aynı adın kullanılması. Eşsesli isimlerden genellikle kaçınılmalıdır. Ayrıca bkz. *eşanlamlı*.

Veri sözlüğü bazen "veritabanı tasarımcısının veritabanı" olarak tanımlanır çünkü tablolar ve

yapıları hakkındaki tasarım kararlarını kaydeder.

Veri sözlüğü gibi, sistem kataloğu da meta veriler içerir. **Sistem kataloğu**, tablo adları, bir tablonun oluşturucusu ve oluşturulma tarihi, her tablodaki sütun sayısı, her sütuna karşılık gelen veri türü, dizin dosya adları, dizin oluşturucuları, yetkili kullanıcılar ve erişim ayrıcalıkları hakkındaki veriler dahil olmak üzere veritabanındaki tüm nesneleri tanımlayan ayrıntılı bir sistem veri sözlüğü olarak tanımlanabilir. Sistem kataloğu gerekli tüm veri sözlüğü bilgilerini içerdiğinden, *sistem kataloğu* ve *veri sözlüğü* terimleri genellikle birbirlerinin yerine kullanılır. Aslında, mevcut ilişkisel veritabanı yazılımı genellikle yalnızca tasarımcının veri sözlüğü bilgilerinin türetilebileceği bir sistem kataloğu sağlar. Sistem kataloğu aslında tabloları kullanıcı/tasarımcı tarafından oluşturulan veritabanı özelliklerini ve içeriklerini saklayan sistem tarafından oluşturulmuş bir veritabanıdır. Bu nedenle, sistem kataloğu tabloları tıpkı kullanıcı/tasarımcı tarafından oluşturulan herhangi bir tablo gibi sorgulanabilir.

Gerçekte, sistem kataloğu otomatik olarak veritabanı dokümantasyonu üretir. Veritabanına yeni tablolar eklendikçe, bu dokümantasyon RDBMS'nin eşanlamlıları ve eşanlamlıları kontrol etmesini ve ortadan kaldırmasını da sağlar. Genel anlamda **eĢ** anlamlı sözcükler, *boar* ve *bore* gibi farklı anlamlara sahip benzer sesli sözcükler ya da *fair* gibi farklı anlamlara sahip sözcüklerdir (bazı bağlamlarda "adil", bazılarında ise "festival" anlamına gelir). Veritabanı bağlamında, eşsesli kelimesi farklı öznitelikleri etiketlemek için aynı adın kullanıldığını gösterir. Örneğin, CUSTOMER tablosundaki bir müşteri adı niteliğini etiketlemek için C\_NAME kullanabilir ve CONSULTANT tablosundaki bir danışman adı niteliğini etiketlemek için C\_NAME kullanabilirsiniz. Karışıklığı azaltmak için, veritabanı eşanlamlılarından kaçınmalısınız; veri sözlüğü bu konuda çok yararlıdır.

**Bölüm 3: İlişkisel Veritabanı Modeli**

**85**

Telif Hakkı 2023 Cengage Learning. Tüm hakları saklıdır. Tamamen veya kısmen kopyalanamaz, taranamaz veya çoğaltılamaz. Elektronik haklar nedeniyle, bazı üçüncü taraf içerikleri eKitap ve/veya eBölüm(ler)den çıkarılmış olabilir.

Editoryal inceleme, bastırılan herhangi bir içeriğin genel öğrenme deneyimini önemli ölçüde etkilemediğine karar vermiştir. Cengage Learning, sonraki hak kısıtlamalarının gerektirmesi halinde ek içeriği istediği zaman kaldırma hakkını saklı tutar.

**Örnek Bir Veri Sözlüğü**

**Tablo 3.6**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tablo Adı** | **Öznitelik Adı** | **İçindekiler** | **Tip** | **Biçim** | **Menzil** | **Gerekli** | **PK VEYA FK** | **FK Referanslı Tablo** |
| MÜŞTERİ | CUS\_CODE | Müşteri hesap kodu | CHAR(5) | 99999 | 10000-99999 | Y | PK |  |
|  | CUS\_LNAME | Müşteri soyadı | VARCHAR(20) | Xxxxxxxx |  | Y |  |  |
|  | CUS\_FNAME | Müşteri adı | VARCHAR(20) | Xxxxxxxx |  | Y |  |  |
|  | CUS\_INITIAL | İlk müşteri | CHAR(1) | X |  |  |  |  |
|  | CUS\_RENEW\_DATE | Müşteri sigortası yenileme tarihi | TARİH | dd-mmm-yyyy |  |  |  |  |
|  | AGENT\_CODE | Temsilci kodu | CHAR(3) | 999 |  |  | FK | AJAN |
| AJAN | AGENT\_CODE | Temsilci kodu | CHAR(3) | 999 |  | Y | PK |  |
|  | AGENT\_AREACODE | Acente alan kodu | CHAR(3) | 999 |  | Y |  |  |
|  | ACENTE\_TELEFONU | Acente telefon numarası | CHAR(8) | 999-9999 |  | Y |  |  |
|  | AGENT\_LNAME | Temsilci soyadı | VARCHAR(20) | Xxxxxxxx |  | Y |  |  |
|  | AGENT\_YTD\_SLS | Acente yılbaşından bugüne satışları | SAYI(9,2) | 9,999,999.99 |  |  |  |  |
|  |  | | | | | | | |
| FK | 5 Yabancı anahtar | | | | | | | |
| PK | 5 Birincil anahtar | | | | | | | |
| CHAR | 5 Sabit karakter uzunluğunda veri (1- 255 karakter) | | | | | | | |
| VARCHAR | 5 Değişken karakter uzunluğunda veri (1- 2.000 karakter) | | | | | | | |
| NUMARA | 5 Sayısal veriler. NUMBER (9,2), ondalık basamağın sağındaki iki basamak da dahil olmak üzere en fazla dokuz basamaklı sayıları belirtmek için kullanılır. Bazı RDBMS'ler yerine NUMERIC veya DECIMAL  kullanır ve diğerleri MONEY veya CURRENCY veri türünün kullanımına izin verir. | | | | | | | |

**Not**

Telefon alan kodları her zaman 0-9 rakamlarından oluşur, ancak alan kodları aritmetik olarak kullanılmadığından, en verimli şekilde karakter verileri olarak saklanırlar. Ayrıca, alan kodları her zaman üç basamaktan oluşur. Bu nedenle, alan kodu veri tipi CHAR(3) olarak tanımlanır. Öte yandan, isimler standart bir uzunluğa uymaz. Bu nedenle, müşteri ilk adları VARCHAR(20) olarak tanımlanır ve adları saklamak için 20 karaktere kadar kullanılabileceğini gösterir. Karakter verileri sola hizalı olarak gösterilir.

#### eşanlamlı

Bir varlık, bir nitelik veya bir ilişki gibi aynı nesneyi tanımlamak için farklı adların kullanılması; eşanlamlılardan genellikle kaçınılmalıdır. Ayrıca bkz. *eşsesli*.

Bir veritabanı bağlamında, **eşanlamlı**, eşseslinin tersidir ve aşağıdakilerin kullanımını gösterir Aynı niteliği tanımlamak için farklı isimler. Örneğin, *car* ve *auto* aynı nesneye atıfta bulunur. Mümkün olduğunca eş anlamlı kelimelerden kaçınılmalıdır.

# 3-6 İlişkisel Veritabanındaki İlişkiler

İlişkisel veritabanlarının tablolar arasındaki ilişkileri uygulamak için ortak nitelikler kullandığını zaten biliyorsunuz. Bu ortak nitelikler yabancı anahtarlar kullanılarak oluşturulur; yani bir tablonun birincil anahtarının başka bir tabloya yerleştirilmesi Sorun, hangi birincil anahtarın yabancı anahtar olarak kullanılacağıdır. Şekil 3.2'deki PRODUCT ve VENDOR arasındaki ilişkide, VENDOR'ın birincil anahtarı PRODUCT'a yabancı anahtar olarak yerleştirilmiştir. Ancak, bunun yerine PRODUCT'un birincil anahtarı VENDOR'a yerleştirilebilir miydi? İlişkisel modelde ortak bir nitelik oluşturmak için yabancı anahtar olarak hangi birincil anahtarın kullanılacağını belirlemek ilişkinin sınıflandırılmasına dayanır. İlişkilerin bire-bir (1:1), bire-çok (1:M) ve çoka-çok (M:N veya M:M) olarak sınıflandırıldığını hatırlayın. Bu bölüm, veritabanı tasarımları geliştirmeye başladığınızda bunları doğru şekilde uygulamanıza yardımcı olmak için bu ilişkileri daha ayrıntılı olarak incelemekte ve doğru yabancı anahtar yerleşimini nasıl belirleyeceğinizi göstermektedir. Bu bölüm aşağıdaki noktalara odaklanmaktadır:

* 1:M ilişkisi ilişkisel modelleme idealidir. Bu nedenle, bu ilişki türü herhangi bir ilişkisel veritabanı tasarımında norm olmalıdır.
* 1:1 ilişkisi herhangi bir ilişkisel veritabanı tasarımında nadiren görülmelidir.
* M:N ilişkileri ilişkisel modelde bu şekilde uygulanamaz. Bu bölümün ilerleyen kısımlarında,

herhangi bir M:N ilişkisinin nasıl iki 1:M dönüştürülebileceğini göreceksiniz.

## 3-6a 1:M İlişkisi

1:M ilişkisi ilişkisel veritabanları için normdur. Böyle bir ilişkinin nasıl modellendiğini ve uygulandığını görmek için Şekil 3.17'de gösterilen PAINTER ve PAINTING örneğini ele alalım.

**Şekil 3.17**

BOYA

boyala r

BOYAMA

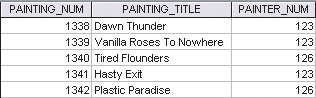


Şekil 3.17'deki veri modeli ile Şekil 3.18'deki uygulamasını karşılaştırın. Şekil 3.18'deki PAINTER ve PAINTING tablo içeriklerini incelerken aşağıdaki özelliklere dikkat edin:

* Her resim bir ve yalnızca bir ressam tarafından yapılmıştır, ancak her ressam birçok resim yapmış olabilir. Ressam 123'ün (Georgette P. Ross) PAINTING tablosunda kayıtlı üç eseri olduğuna dikkat edin.
* PAINTER tablosundaki herhangi bir satır için PAINTING tablosunda yalnızca bir satır vardır, ancak PAINTER tablosundaki herhangi bir satır için PAINTING tablosunda birçok satır olabilir.

Yabancı anahtarın yerleşimi 1:M ilişkisinin etkili bir şekilde uygulanması için kritik öneme sahiptir. Bu bölümün başından itibaren, ilişkisel bir tablonun özelliklerinden birinin her hücrenin yalnızca tek bir değer içerebilmesi olduğunu hatırlayın. Her tablo yalnızca bir değerle ilişkilendirildiğinden

bir ressam, PAINTER\_NUM'u PAINTING'e yerleştirmek, her resmin saklanacak bir ressam numarasına ve bir numaranın saklanacağı bir yere sahip olduğu bir senaryo ortaya çıkarır. Mükemmel! PAINTING\_NUM öğesini PAINTER içine yerleştirmiş olsaydınız, 123 numaralı ressamın üç resim numarasını (1338, 1339 ve 1341) saklaması gerekirdi, ancak bunları saklamak için yalnızca bir hücre olurdu. Bu, ilişkisel tablolar için kuralları ihlal eder ve uygulanabilir bir çözüm olmazdı.



**Not**

Bir-çok (1:M) ilişkisi, ilişkisel modelde *"1" tarafının birincil anahtarının "çok" tarafının tablosuna*

*yabancı anahtar olarak* yerleştirilmesiyle kolayca uygulanabilir.

### Şekil 3.18 PAINTER ve PAINTING arasında Uygulanan 1:M İlişkisi

Tablo adı: PAINTER

Birincil anahtar: PAINTER\_NUM Veritabanı adı: Ch03\_Museum Yabancı anahtar: yok

Tablo adı: PAINTING Birincil anahtar: PAINTING\_NUM Yabancı anahtar: PAINTER\_NUM

1:M ilişkisi herhangi bir veritabanı ortamında bulunur. Tipik bir kolej veya üniversitedeki öğrenciler, her KURS'un birçok SINIF oluşturabileceğini, ancak her SINIF'ın yalnızca bir KURS'a atıfta bulunduğunu keşfedeceklerdir. Örneğin, bir Muhasebe II dersi iki sınıf oluşturabilir: biri Pazartesi, Çarşamba ve Cuma günleri (MWF) sabah 10:00 ile 10:50 arasında ve diğeri Perşembe günleri (Th) akşam 18:00 ile 20:40 arasında:

* Her KURS'un birçok SINIFI olabilir, ancak her SINIF yalnızca bir KURS'a referans verir.
* CLASS herhangi bir satır için COURSE tablosunda yalnızca bir satır olacaktır, ancak COURSE tablosundaki herhangi bir satır için CLASS tablosunda birçok satır olabilir.

Şekil 3.19, COURSE ve CLASS arasındaki 1:M ilişkisi için varlık ilişki modelini (ERM)

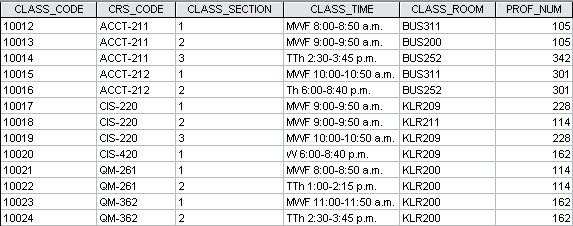
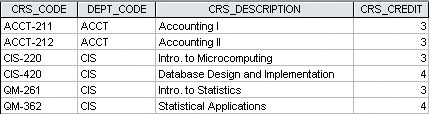
eşler.

**Şekil 3.19 KURS ve SINIF arasındaki 1:M İlişkisi**

KURS var SINIF

**DERS ve SINIF arasında Uygulanan 1:M İlişkisi**

**Şekil 3.20**



Tablo adı: COURSE Birincil anahtar: CRS\_CODE Yabancı anahtar: yok

Veritabanı adı: Ch03\_TinyCollege

Tablo adı: CLASS Birincil anahtar: CLASS\_CODE Yabancı anahtar: CRS\_CODE

KURS ve SINIF arasındaki 1:M ilişkisi Şekil 3.20'de daha ayrıntılı olarak gösterilmiştir. Şekil 3.20'yi kullanarak, bazı önemli terminolojiyi gözden geçirmek için bir dakikanızı ayırın. CLASS tablosundaki CLASS\_ CODE'un her satırı benzersiz bir şekilde tanımladığına dikkat edin. Bu nedenle, CLASS\_CODE birincil anahtar olarak seçilmiştir. Ancak, CRS\_CODE ve CLASS\_SECTION kombinasyonu da sınıf tablosundaki her satırı benzersiz bir şekilde tanımlayacaktır. Başka bir deyişle, CRS\_CODE ve CLASS\_SECTION'dan oluşan *bileşik* bir aday anahtardır. Herhangi bir *aday anahtarın* not-null ve unique kısıtlamalarına sahip olması gerekir.

(Bunun nasıl yapıldığını öğrendiğinizde göreceksiniz

SQL Bölüm 8'de).

Örneğin, Şekil 3.18'de PAINTER tablosunun birincil anahtarı PAINTER\_NUM'un yabancı olarak PAINTING tablosuna dahil edildiğine dikkat edin. Benzer şekilde, Şekil 3.20'de, COURSE tablosunun birincil anahtarı CRS\_CODE, yabancı anahtar olarak CLASS tablosuna dahil edilmiştir.

## b 1:1 İlişki

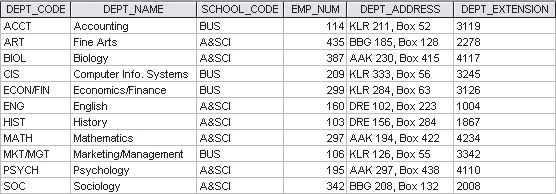
1:1 etiketinden de anlaşılacağı üzere, 1:1 ilişkisindeki bir varlık yalnızca bir başka varlıkla ilişkili olabilir ve bunun tersi de geçerlidir. Örneğin, bir bölüm başkanı -bir profesör- sadece bir departmana başkanlık edebilir ve bir departmanın sadece bir bölüm başkanı olabilir. PROFESÖR ve DEPARTMENT varlıkları bu nedenle 1:1 ilişki sergiler. (Tüm profesörlerin bir bölüme başkanlık etmediğini ve profesörlerden bir bölüme başkanlık etmelerinin *istenemeyeceğini* iddia edebilirsiniz. Yani, iki varlık arasındaki ilişki isteğe bağlıdır. Ancak, tartışmanın bu aşamasında, dikkatinizi temel 1:1 ilişkisine odaklamalısınız. İsteğe bağlı ilişkiler Bölüm 4'te ele alınacaktır). Temel 1:1 ilişkisi Şekil 3.21'de modellenmiştir ve uygulaması Şekil 3.22'de gösterilmiştir.

**Şekil 3.21 PROFESÖR ve BÖLÜM arasındaki 1:1 İlişki**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PROFESÖR | sandal | | | | DEPARTMAN |
|  |  |
|  |  |  |  |
|  |  | yeler |  |
|  | | | |

### Şekil 3.22 PROFESÖR ve BÖLÜM arasında Uygulanan 1:1 İlişki



Tablo adı: PROFESSOR Birincil anahtar: EMP\_NUM Yabancı anahtar: DEPT\_CODE

Veritabanı adı: Ch03\_TinyCollege

PROFESÖR ilişkisini kullanan 1:M DEPARTMENT, PROFESSOR tablosuna DEPT\_CODE yabancı anahtarının yerleştirilmesi yoluyla uygulanır.

Tablo adı: DEPARTMENT Birincil anahtar: DEPT\_CODE Yabancı anahtar: EMP\_NUM

1:1 PROFESÖR sandalye DEPARTMENT ilişkisi, DEPARTMENT tablosuna EMP\_NUM yabancı anahtarının yerleştirilmesi yoluyla uygulanır.

Şekil 3.22'deki tabloları incelerken birkaç önemli özelliğe dikkat edin:

* + Her profesör bir Tiny College çalışanıdır. Bu nedenle, profesör tanımlaması EMP\_NUM aracılığıyla yapılır. (Ancak, tüm çalışanların profesör olmadığını unutmayın - başka bir isteğe bağlı ilişki daha vardır).
  + 1:1 "PROFESSOR chairs DEPARTMENT" ilişkisi, DEPARTMENT tablosunda EMP\_NUM yabancı anahtarına sahip olarak uygulanır. 1:1 ilişkisinin, "çok" tarafının tek bir oluşumla sınırlandırıldığı 1:M ilişkisinin özel bir durumu olarak ele alındığını unutmayın. Bu durumda, DEPARTMENT bir yabancı anahtar olarak EMP\_NUM'u içerir ve bu da bir kürsüye sahip olan *departman* olduğunu gösterir.
  + Ayrıca PROFESSOR tablosunun 1:M "DEPARTMENT employs PROFESSOR" ilişkisini uygulamak için DEPT\_CODE yabancı anahtarını içerdiğine dikkat edin. Bu, iki varlığın aynı anda iki (hatta daha fazla) ilişkiye nasıl katılabileceğine dair iyi bir örnektir.

**Çevrimiçi İçerik**

[www.cengage.com ,](http://www.cengage.com/) adresinde Ch03\_ TinyCollege veritabanını açarsanızSTUDENT ve CLASS varlıklarının yabancı anahtar olarak hala PROF\_NUM kullandığını görürsünüz. PROF\_NUM ve EMP\_NUM aynı özniteliğin etiketleridir, bu da eşanlamlıların kullanımına örnektir - yani, aynı öznitelik için farklı adlar.

Bu eş anlamlı kelimeler gelecekte ortadan kaldırılacaktır

Tiny College veri tabanı geliştirilmeye devam ederken bölümler.

**Çevrimiçi İçerik**

[www.](http://www/) cengage.com adresindeki Ch03\_ AviaCo veritabanına bakarsanız, 1:1 PILOT to

EMPLOYEE ilişkisi. Bu ilişki, Bölüm 5'te öğreneceğiniz bir genelleme hiyerarşisine dayanmaktadır.

1:1 ilişki ile yabancı anahtar yerleşimi açısından, teoride her iki varlıktan birindeki birincil anahtar diğer varlıkta yabancı anahtar olarak kullanılabilir. Pratikte, bazı durumlar bize yabancı anahtarı bir yönde veya diğer yönde yerleştirme tercihi verecektir. Bu örnekte, aynı varlıklar arasında 1:M ilişkisinin bulunması, eşanlamlılıktan kaçınmak için yabancı anahtarın 1:1 ilişkisine yerleştirilmesini tercih etmemize neden olmuştur. İsteğe bağlı ilişkiler gibi diğer durumlar Bölüm 4'te ele alınmaktadır. Önceki "PROFES- SOR sandalyeleri DEPARTMAN" örneği uygun bir 1:1 ilişkisini göstermektedir. *Aslında, 1:1 ilişkisinin kullanılması, iki varlık kümesinin olmaması gerektiği halde aynı tabloya yerleştirilmemesini sağlar*. Ancak, 1:1 ilişkisinin varlığı bazen varlık bileşenlerinin doğru tanımlanmadığı anlamına gelir. Bu, iki varlığın aslında aynı tabloya ait olduğunu gösterebilir!

1:1 ilişkilerin nadiren kullanılması gerekse de, belirli koşullar kesinlikle kullanılmalarını gerektirir. Bölüm 5, Gelişmiş Veri Modelleme'de, genelleme hiyerarşisi olarak adlandırılan ve belirli koşullar altında veritabanı tasarımlarını geliştirerek boşların çoğalmasını önlemek için güçlü bir araç olan bir kavramı keşfedeceksiniz. Genelleme hiyerarşilerinin bir özelliği, 1:1 ilişkiler olarak uygulanmalarıdır.

## 3-6c M:N İlişkisi

Çoktan çoka (M:N) ilişki ilişkisel ortamda doğrudan desteklenmez. Ancak, M:N ilişkileri, orijinal

varlıklarla 1:M ilişkilerinde yeni bir varlık oluşturularak uygulanabilir.

M:N ilişkisini keşfetmek için tipik bir üniversite ortamını düşünün. Şekil 3.23'teki ER modeli bu M:N ilişkisini göstermektedir.

Şekil 3.23'te ERM'nin özelliklerine dikkat edin.

* + Her SINIFIN birçok ÖĞRENCİSİ olabilir ve her ÖĞRENCİ birçok SINIF alabilir.
  + STUDENT tablosundaki herhangi bir satır için CLASS tablosunda birçok satır olabilir ve CLASS tablosundaki herhangi bir satır için STUDENT tablosunda birçok satır olabilir.

### Şekil 3.23 ERM'nin ÖĞRENCİ ve SINIF arasındaki M:N İlişkisi

ÖĞRENCİ



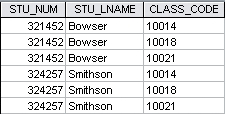
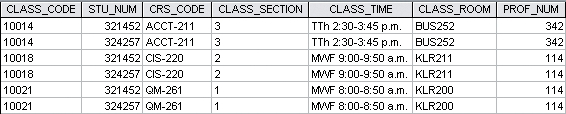
var

SINIF

|  |  |  |
| --- | --- | --- |
| **Tablo 3.7** | **Örnek Öğrenci Kayıt Verileri** | |
| **Öğrencinin Soyadı** | | **Seçilmiş Sınıflar** |
| Bowser | | Muhasebe 1, ACCT-211, kod 10014 Mikrobilgisayara Giriş, CIS-220, kod 10018 İstatistiğe Giriş, QM-261, kod 10021 |
| Smithson | | Muhasebe 1, ACCT-211, kod 10014 Mikrobilgisayara Giriş, CIS-220, kod 10018 İstatistiğe Giriş, QM-261, kod 10021 |

M:N ilişkisini daha yakından incelemek için, her biri üç ders alan iki öğrencisi olan küçük bir üniversite düşünün. Tablo 3.7 iki öğrenci için kayıt verilerini göstermektedir.

Böyle bir veri ilişkisi ve Tablo 3.7'örnek veriler göz önüne alındığında, Şekil 3.24'te gösterildiği gibi, ilişkinin "çok" tarafına ilgili tablonun birincil anahtarına işaret eden bir yabancı anahtar ekleyerek bu M:N ilişkisini uygulayabileceğinizi yanlış bir şekilde varsayabilirsiniz.



**Şekil 3.24 ÖĞRENCİ ve SINIF arasındaki M:N İlişkisinin Yanlış Uygulanması**

Tablo adı: STUDENT Birincil anahtar: STU\_NUM Yabancı anahtar: yok

Veritabanı adı: Ch03\_CollegeTry

Tablo adı: CLASS Birincil anahtar: CLASS\_CODE Yabancı anahtar: STU\_NUM

Ancak, M:N ilişkisi iki iyi nedenden dolayı Şekil 3.24'te gösterildiği gibi

uygulanmamalıdır:

* Tablolar birçok fazlalık yaratır. Örneğin, STU\_NUM değerlerinin STUDENT tablosunda birçok kez yer aldığına dikkat edin. Gerçek dünyadaki bir durumda, adres, sınıflandırma, ana dal ve ev telefonu gibi ek öğrenci öznitelikleri de STU- DENT tablosunda yer alacak ve bu öznitelik değerlerinin her biri burada gösterilen kayıtların her birinde tekrarlanacaktır. Benzer şekilde, CLASS tablosu da çok sayıda yineleme içerir: dersi alan her öğrenci bir CLASS kaydı oluşturur. Eğer CLASS tablosu kredi saatleri ve kurs açıklaması gibi nitelikleri de içeriyorsa sorun daha da kötüleşecektir. Bu fazlalıklar Bölüm 1'de tartışılan anomalilere yol açar.
* İki tablonun yapısı ve içeriği göz önüne alındığında, ilişkisel işlemler çok karmaşık hale gelir ve muhtemelen sistem verimliliği hatalarına ve çıktı hatalarına yol açar.

#### bileşik varlık

Bir M:N ilişkisini iki 1:M ilişkisine için tasarlanmış bir varlık.

Bileşik varlığın birincil anahtarı, en azından bağladığı varlıkların birincil anahtarlarını içerir.

*Köprü varlık* veya *ilişkisel varlık* olarak da bilinir. Ayrıca bkz. *bağlantı tablosu*.

#### köprü kuruluşu

Bkz. *bileşik varlık*.

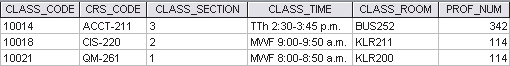
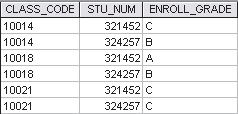
#### ilişkisel varlık

Bkz. *bileşik varlık*.

Neyse ki, M:N ilişkisinin doğasında bulunan sorunlar, bir **bileşik varlık** (**köprü varlık** veya **ilişkisel varlık** olarak da adlandırılır) oluşturularak kolayca önlenebilir. Böyle bir tablo başlangıçta bir M:N ilişkisinde ilişkili olan tabloları birbirine bağlamak için kullanıldığından, bileşik varlık yapısı yabancı anahtarlar olarak en *azından* bağlanacak tabloların birincil anahtarlarını içerir. Veritabanı tasarımcısının bileşik tablonun birincil anahtarını tanımlarken iki ana seçeneği vardır: bu yabancı anahtarların kombinasyonunu kullanmak veya yeni bir birincil anahtar oluşturmak.

ERM'deki her bir varlığın bir tablo tarafından temsil edildiğini unutmayın. Bu nedenle, CLASS ve STU- DENT tablolarını birbirine bağlamak için Şekil 3.25'te gösterilen bileşik ENROLL tablosunu oluşturabilirsiniz. Bu örnekte, ENROLL tablosunun birincil anahtarı, CLASS\_CODE ve STU\_NUM yabancı anahtarlarının birleşimidir. Ancak tasarımcı, her ENROLL tablo satırını benzersiz bir şekilde tanımlamak için farklı bir satır değeri kullanarak ENROLL\_LINE gibi tek özellikli yeni bir birincil anahtar oluşturmaya karar verebilirdi. (Microsoft Access kullanıcıları bu tür satır değerlerini otomatik olarak oluşturmak için Autonumber veri türünü kullanabilir).

**Şekil 3.25 M:N İlişkisini İki 1:M İlişkisine Dönüştürme**



Tablo adı: STUDENT Birincil anahtar: STU\_NUM Yabancı anahtar: yok

Veritabanı adı: Ch03\_CollegeTry2

Tablo adı: ENROLL

Birincil anahtar: CLASS\_CODE + STU\_NUM Yabancı anahtar: CLASS\_CODE, STU\_NUM

Tablo adı: CLASS Birincil anahtar: CLASS\_CODE Yabancı anahtar: CRS\_CODE

#### bağlantı tablosu

İlişkisel modelde,

M:N ilişkisi uygulayan bir tablo. Ayrıca bkz. *bileşik varlık*.

**Not**

Bağlantı özniteliklerine ek olarak, bileşik ENROLL tablosu kursta kazanılan not gibi ilgili öznitelikleri de içerebilir. Aslında, bileşik bir tablo, tasarımcının izlemek istediği herhangi bir sayıda öznitelik içerebilir. Bileşik varlığın, *gerçek bir tablo olarak uygulanmasına rağmen*, *kavramsal* olarak bir amaç için yaratılmış mantıksal bir varlık olduğunu unutmayın: orijinal M:N ilişkisindeki çoklu potansiyelini ortadan kaldırmak.

Şekil 3.25'teki ENROLL tablosu STUDENT ve CLASS adlı iki tabloyu birbirine bağladığından, bu tablo aynı zamanda bir **bağlantı tablosu** olarak da adlandırılır. Başka bir deyişle, bir bağlantı tablosu bir bileşik varlığın uygulanmasıdır.

Şekil 3.25'te gösterilen ENROLL tablosu gerekli M:N - 1:M dönüşümünü sağlar. ENROLL tablosu tarafından temsil edilen bileşik varlığın, bağlayıcı olarak hizmet ettiği CLASS ve STUDENT tablolarının en azından birincil anahtarlarını (sırasıyla CLASS\_CODE ve STU\_NUM) içermesi gerektiğini gözlemleyin. Ayrıca STUDENT ve CLASS tablolarının artık varlık başına yalnızca bir satır içerdiğini unutmayın. ENROLL tablosu, yabancı anahtar değerlerinin birden fazla oluşumunu içerir, ancak bu kontrollü fazlalıklar, referans bütünlüğü uygulandığı sürece anomalilere neden olamaz. Gerektiğinde ek nitelikler atanabilir. Bu durumda, ENROLL\_GRADE bir raporlama gereksinimini karşılamak için seçilmiştir. Ayrıca ENROLL\_GRADE'in bileşik birincil anahtara tamamen bağımlı olduğunu unutmayın. Doğal olarak, dönüşüm ERM'ye de yansıtılır. Revize edilmiş ilişki Şekil 3.26'da gösterilmektedir.

Şekil 3.26'yı incelerken, ENROLL adlı bileşik varlığın STUDENT ve CLASS arasındaki bağlantı tablosunu temsil ettiğine dikkat edin.

### Şekil 3.26 M:N İlişkilerinin İki 1:M İlişkisine Değiştirilmesi

KURS ve SINIF arasındaki 1:M ilişkisi ilk olarak Şekil 3.19 ve Şekil 3.20'de gösterilmiştir. Veritabanının fazlalıklarını kontrol ederken bile mevcut bilgi miktarını artırabilirsiniz. Böylece Şekil 3.27, Şekil 3.19'da gösterilen COURSE ve CLASS arasındaki 1:M ilişkisini de içeren genişletilmiş ERM'yi göstermektedir. Modelin, her bir SINIF için ortak olan tüm KURS verilerinin KURS tablosunda tutulmasını sağlayarak fazlalıkları kontrol ederken bir SINIF'ın birden fazla bölümünü işleyebileceğine dikkat edin.

**Şekil 3.27 Genişletilmiş ER Modeli**

KURS

var

ÖĞRENCİ



ÖĞRENCİ

SINIF

var

ÖĞRENCİ

KAYIT OL

SINIF

kayıtlar içinde

gösterir

kayıtlar

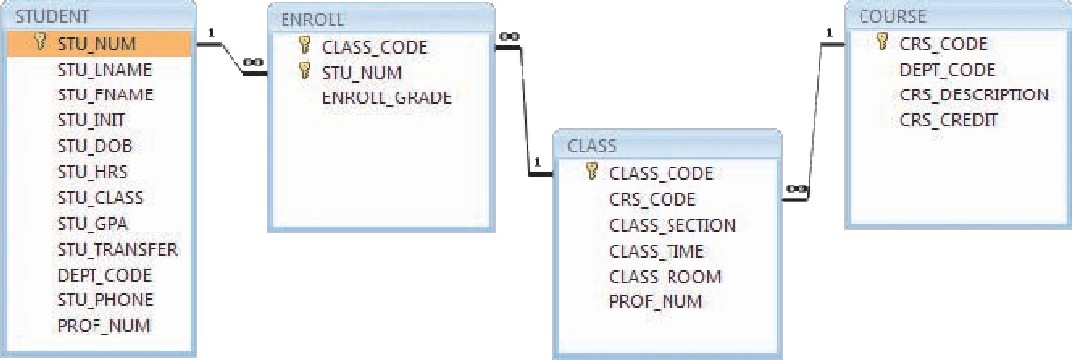
KAYIT OL

içinde gösterir

SINIF

**Ch03\_TinyCollege Veritabanı için İlişkisel Diyagram**

**Şekil 3.28**



Şekil 3.27'deki ERM'ye karşılık gelen ilişkisel diyagram Şekil 3.28'de gösterilmektedir. ERM, daha karmaşık veritabanları tasarlamak için nasıl kullanıldığını göstermek amacıyla Bölüm 4'te daha ayrıntılı olarak incelenecektir. ERM ayrıca Ek B ve C'de bir üniversite bilgisayar laboratuarının gerçekçi bir veritabanı tasarımını geliştirmek ve uygulamak için temel olarak

kullanılacaktır.

Bu eklere [www.cengage.com](http://www.cengage.com/) adresinden ulaşılabilir.

# Veri Yedekliliği Yeniden Değerlendirildi

Bölüm 1'de, veri fazlalığının veri anormalliklerine yol açtığını ve bunun da veritabanının etkinliğini yok edebileceğini öğrendiniz. Ayrıca ilişkisel veritabanının, yabancı anahtarlar olarak adlandırılan tablolar tarafından paylaşılan ortak nitelikleri kullanarak veri fazlalıklarını kontrol etmeyi mümkün kıldığını da öğrendiniz.

Yabancı anahtarların doğru kullanımı veri fazlalığını kontrol etmek için çok önemlidir, ancak yabancı anahtar değerleri birçok kez tekrarlanabileceğinden sorunu tamamen ortadan kaldırmazlar. Ancak, yabancı anahtarların doğru kullanımı veri fazlalıklarını ve yıkıcı veri anormalliklerinin ortaya çıkma olasılığını *en aza indirir*.

**Not**

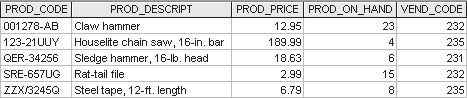
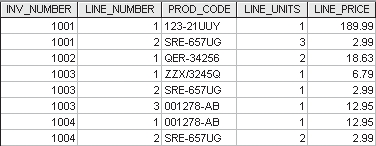
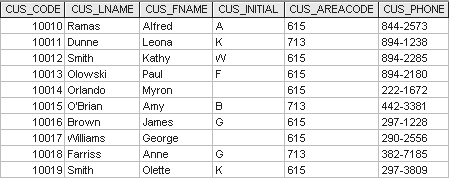
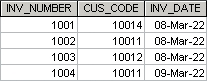
Gereksizliğin gerçek testi, belirli bir özniteliğin kaç kopyasının saklandığı *değil, bir özniteliğin ortadan kaldırılmasının bilgiyi ortadan kaldırıp kaldırmayacağıdır*. Bu nedenle, bir özniteliği silerseniz ve orijinal bilgi ilişkisel cebir yoluyla hala üretilebiliyorsa, bu özniteliğin dahil edilmesi gereksiz olacaktır. Bu görüşü göz önüne alındığında, uygun yabancı anahtarlar bir tabloda birden fazla bulunmalarına rağmen açıkça gereksiz değildir. Ancak, bu daha az kısıtlayıcı fazlalık görüşünü kullandığınızda bile, *kontrollü* fazlalıkların genellikle işlem hızını ve/veya bilgi gereksinimlerini sağlamak için sistemin bir parçası olarak tasarlandığını unutmayın.

Bölüm 4'te veritabanı tasarımcılarının genellikle birbiriyle çelişen üç gereksinimi uzlaştırması gerektiğini öğreneceksiniz: tasarım zarafeti, işlem hızı ve bilgi gereksinimleri. Ayrıca, Bölüm 13, İş Zekası ve Veri Ambarları'nda doğru veri ambarı tasarımının düzgün çalışması için dikkatlice tanımlanmış ve kontrol edilmiş veri fazlalıkları gerektirdiğini öğreneceksiniz. Veri fazlalıklarını nasıl tanımladığınızdan bağımsız olarak, zarar verme potansiyeli doğru uygulama ve dikkatli kontrol ile sınırlandırılır.

Veri fazlalığını kontrol etmek ne kadar önemli olsa da, bazen veritabanının önemli bilgi amaçlarına hizmet etmesi için veri seviyesinin gerçekten artırılması gerekir. Bu tür fazlalıklar hakkında Bölüm 13'te bilgi edineceksiniz. Ayrıca, veri fazlalıkları bazen verilerin tarihsel doğruluğunu korumak için varmış *gibi görünür*. Örneğin, küçük bir faturalama sistemi düşünün. Sistem, bir veya daha fazla ÜRÜN satın alabilen ve böylece bir FATURA oluşturan MÜŞTERİ'yi içerir. Bir müşteri aynı anda birden fazla ürün satın alabileceğinden, bir fatura, her biri satın alınan ürün hakkında ayrıntılar sağlayan birkaç fatura SATIRI içerebilir. Faturada görünen her ürün için tutarlı bir fiyatlandırma girdisi sağlamak amacıyla ÜRÜN tablosu ürün fiyatını içermelidir. Böyle bir sistemin parçası olan tablolar Şekil 3.29'da gösterilmektedir. Sistemin ilişkisel diyagramı Şekil 3.30'da gösterilmektedir.

**Küçük Bir Faturalama Sistemi**

**Şekil 3.29**



Tablo adı: CUSTOMER Birincil anahtar:

CUS\_CODE Yabancı anahtar: yok

Veritabanı adı: Ch03\_SaleCo

Tablo adı: INVOICE Birincil anahtar: INV\_NUMBER Yabancı anahtar: CUS\_CODE

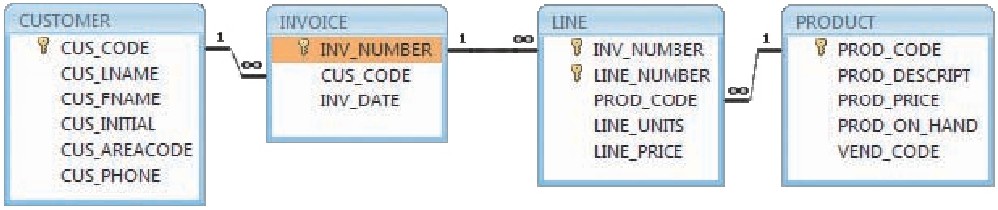
Tablo adı: LINE

Birincil anahtar: INV\_NUMBER + LINE\_NUMBER Yabancı anahtar: INV\_NUMBER, PROD\_CODE

Tablo adı: PRODUCT Birincil anahtar: PROD\_CODE Yabancı anahtar: yok

**Faturalama Sistemi için İlişkisel Diyagram**

**Şekil 3.30**



İki şekildeki tabloları ve ilişkileri incelerken, aşağıdakileri takip edebileceğinizi unutmayın

tipik satış bilgilerinden oluşur. Örneğin, dört tablo arasındaki ilişkileri izleyerek, 10014 numaralı müşterinin (Myron Orlando) 8 Mart 2022 tarihinde 1001 numaralı faturaya yazılan iki ürünü satın aldığını keşfedersiniz: bir adet 16 inç çubuklu Houselite zincirli testere ve üç adet fare kuyruğu dosyası. Başka bir deyişle, CUSTOMER tablosundaki 10014 numaralı CUS\_CODE değerini INVOICE tablosundaki eşleşen CUS\_CODE değerine kadar izleyin. Ardından, INV\_NUMBER 1001'i LINE tablosundaki ilk iki satıra kadar izleyin. Son olarak, LINE'daki iki PROD\_CODE değerini PRODUCT'taki PROD\_CODE değerleriyle eşleştirin. Uygulama yazılımı, her bir fatura satır kaleminin SATIR\_BİRİMLERİ ile SATIR\_FİYATI çarpılarak, sonuçlar toplanarak ve uygun vergiler uygulanarak doğru faturayı yazmak için kullanılacaktır. Daha sonra, diğer uygulama yazılımları aynı tekniği kullanarak satışları hafta, ay veya yıl bazında takip eden ve karşılaştıran satış raporları yazabilir. Şekil 3.29'daki satış işlemlerini incelerken, müşteriye fatura edilen ürün fiyatının PRODUCT tablosundan türetildiğini düşünebilirsiniz çünkü ürün verileri burada saklanmaktadır.

*Ancak aynı ürün fiyatı neden LINE tablosunda tekrar ortaya çıkıyor? Bu bir veri fazlalığı değil midir?* Kesinlikle öyle görünüyor, ancak bu kez, görünürdeki fazlalık sistemin başarısı için çok önemlidir. Ürün fiyatının PRODUCT tablosundan LINE tablosuna kopyalanması *işlemlerin tarihsel doğruluğunu* korur. Örneğin, LINE tablosuna LINE\_PRICE değerini yazmadığınızı ve satış gelirini hesaplamak için PRODUCT tablosundaki PROD\_PRICE değerini kullandığınızı varsayalım. Şimdi, fiyatların sıklıkla yaptığı gibi PRODUCT tablosunun PROD\_PRICE değerinin değiştiğini varsayalım. Bu fiyat değişikliği, sonraki tüm satış geliri hesaplamalarına uygun şekilde yansıtılacaktır. Ancak, geçmiş satış gelirlerinin hesaplamaları da işlem gerçekleştiğinde yürürlükte olmayan yeni ürün fiyatını yansıtacaktır! Sonuç , tüm geçmiş işlemler için gelir hesaplamaları hatalı olacak ve böylece zaman içinde uygun satış karşılaştırmaları yapma olasılığı ortadan kalkacaktır.

Öte yandan, fiyat verileri PRODUCT tablosundan kopyalanır ve LINE tablosunda işlemle birlikte

saklanırsa, bu fiyat *her* zaman *o sırada* gerçekleşen işlemi doğru bir şekilde yansıtacaktır.

Keşfedeceksiniz

Bu tür planlı "fazlalıklar" iyi veritabanı tasarımında yaygındır.

Son olarak, Şekil 3.29'daki LINE tablosunda neden LINE\_NUMBER özniteliğinin kullanıldığını merak edebilirsiniz. INV\_NUMBER ve PROD\_CODE kombinasyonu yeterli bir bileşik birincil anahtar olmaz mıydı ve bu nedenle LINE\_NUMBER gereksiz değil mi? Evet, öyle, ancak bu fazlalık, genellikle bu tür satır numaralarını otomatik olarak oluşturan faturalama yazılımlarında yaygın bir uygulamadır. Bu durumda, fazlalık gerekli değildir, ancak otomatik olarak oluşturulduğu göz önüne alındığında, fazlalık bir anormallik kaynağı değildir. LINE\_NUMBER'ın dahil edilmesi başka bir fayda da sağlar: alınan faturalama verilerinin sırası her zaman verilerin girildiği sırayla eşleşecektir. Birincil anahtarın bir parçası olarak ürün kodları kullanılıyorsa, fatura tamamlanır tamamlanmaz ve veriler depolanır depolanmaz indeksleme bu ürün kodlarını düzenleyecektir. Bir müşteri arayıp "Faturamdaki ikinci kalemin fiyatı yanlış" dediğinde ve siz de satırları müşterinin kopyasındakilerden farklı bir sıra gösteren bir faturaya baktığınızda yaşanabilecek olası karışıklığı tahmin edebilirsiniz!

# İndeksler

#### indeks

Dizin anahtar değerleri ve satır kimliği değerlerinden (işaretçiler) oluşan sıralı bir dizi. Dizinler genellikle veri alımını hızlandırmak ve kolaylaştırmak için kullanılır. *Dizin anahtarı* olarak da bilinir.

Bir kütüphanede bir kitabı bulmak istediğinizi varsayalım. İstediğiniz kitabı bulana kadar her kitaba bakmak mantıklı mı? Elbette hayır; başlık, konu ve yazara göre indekslenen kütüphane kataloğunu kullanırsınız. Dizin (manuel ya da bilgisayarlı kütüphane kataloğunda) sizi kitabın bulunduğu yere yönlendirir ve böylece kitaba hızlı ve basit bir şekilde ulaşmanızı sağlar. **Dizin**, bir tablodaki satırlara mantıksal olarak erişmek için kullanılan düzenli bir düzenlemedir.

Ya da bu kitapta *ER modeli* gibi bir konu bulmak istediğinizi varsayalım. Konuya rastlayana kadar her sayfayı okumak mantıklı mı? Elbette hayır; kitabın dizinine gitmek, *ER modeli* ifadesini aramak ve sizi uygun sayfa(lar)a yönlendiren referansları okumak çok daha basittir. Her durumda, ihtiyaç duyulan bir öğeyi hızlı bir şekilde bulmak için bir dizin kullanılır.

**Şekil 3.31 Bir Endeksin Bileşenleri**



BOYAMA tablo dizini

BOYAMA masası

123 1, 2, 4

126

3, 5

PAINTER\_NUM

(dizin anahtarı)

BOYAMA için işaretçiler tablo satırları

İlişkisel veritabanı ortamındaki indeksler, önceki paragraflarda açıklanan indeksler gibi çalışır. Kavramsal açıdan bakıldığında, bir dizin bir dizin anahtarı ve bir dizi işaretçiden oluşur. **İndeks anahtarı** aslında indeksin referans noktasıdır. Daha resmi olarak, bir dizin anahtarların ve sıralı bir düzenlemesidir. Her anahtar, anahtar tarafından tanımlanan verinin konumuna işaret eder.

Örneğin, Şekil 3.18'deki Ch03\_Museum veritabanında belirli bir ressam tarafından oluşturulan tüm tabloları aramak istediğinizi varsayalım. Bir dizin olmadan, PAINTING tablosundaki her satırı okumanız ve PAINTER\_NUM'un istenen ressamla eşleşip eşleşmediğine bakmanız gerekir. Ancak, PAINTER tablosunu indeksler ve PAINTER\_NUM indeks anahtarını kullanırsanız, sadece indekste uygun PAINTER\_NUM'a bakmanız ve eşleşen işaretçileri bulmanız gerekir. Kavramsal olarak konuşmak gerekirse, dizin Şekil 3.31'deki sunuma benzeyecektir.

Şekil 3.31'i incelerken, ilk PAINTER\_NUM dizin anahtarı değerinin (123) PAINTING tablosunun 1, 2 ve 4 numaralı kayıtlarında bulunduğuna dikkat edin. İkinci PAINTER\_NUM dizin anahtarı değeri (126) PAINTING tablosunun 3 ve 5 numaralı kayıtlarında bulunur.

DBMS'ler indeksleri birçok amaç için kullanır. Az önce bir dizinin verileri daha verimli bir şekilde almak için kullanılabileceğini öğrendiniz, ancak dizinler bir DBMS tarafından belirli bir özniteliğe veya özniteliklere göre sıralanmış verileri almak için de kullanılabilir. Örneğin, bir müşterinin soyadı üzerinde bir dizin oluşturmak, müşteri verilerini müşterinin soyadına göre alfabetik olarak almanızı sağlayacaktır. Ayrıca, bir dizin anahtarı bir veya daha fazla öznitelikten oluşabilir. Örneğin, Şekil 3.29'da, VEND\_CODE ve PROD\_CODE üzerinde bir dizin oluşturarak PRODUCT tablosundaki satıcıya göre sıralanmış ve satıcı içinde ürüne göre sıralanmış tüm satırları alabilirsiniz.

Dizinler, birincil anahtarların uygulanması için DBMS'lerde önemli bir rol oynar. Bir tablonun birincil anahtarını tanımladığınızda, DBMS otomatik olarak birincil anahtar sütun(lar)ı üzerinde benzersiz bir dizin oluşturur. Örneğin, Şekil 3.29'da CUS\_CODE'u CUSTOMER tablosunun birincil anahtarı olarak bildirdiğinizde, VTYS otomatik olarak bu nitelik üzerinde benzersiz bir dizin oluşturur. **Benzersiz** bir **dizinde**, adından da anlaşılacağı gibi, dizin anahtarının kendisiyle ilişkili yalnızca bir işaretçi değeri (satırı) olabilir. (Şekil 3.31'deki dizin benzersiz bir dizin değildir çünkü PAINTER\_NUM ile ilişkili birden fazla işaretçi değeri vardır. Örneğin, ressam numarası 123, PAINTING tablosunda üç satıra (1, 2 ve 4) işaret eder).

Bir tablonun birçok dizini olabilir, ancak her dizin yalnızca bir tabloyla ilişkilendirilir. Dizin anahtarı birden fazla özniteliğe sahip olabilir (bileşik dizin). Bir dizin oluşturmak kolaydır. Bölüm 8'de basit bir SQL komutunun gerekli herhangi bir dizini oluşturduğunu öğreneceksiniz.

#### dizin anahtarı

*Dizine* bakınız.

#### benzersiz dizin

Dizin anahtarının yalnızca bir ilişkili işaretçi değerine (satır) sahip olabileceği bir dizin.

# Codd'un İlişkisel Veritabanı Kuralları

1985 yılında Dr. E. F. Codd ilişkisel bir veritabanı sistemini tanımlamak için 12 kuraldan oluşan bir liste yayınlamıştır.1 Codd bu listeyi, birçok satıcının minimum ilişkisel standartları karşılamamasına rağmen ürünlerini "ilişkisel" olarak pazarlamasından duyduğu endişe nedeniyle yayınlamıştır. Tablo

3.8'de gösterilen Dr. Codd'un listesi, gerçekten ilişkisel bir veritabanının ne olması gerektiğine dair bir referans çerçevesidir. Baskın veritabanı satıcılarının bile 12 kuralın tümünü tam olarak desteklemediğini unutmayın.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tablo 13.8** | | **Dr. Codd'un 12 İlişkisel Veritabanı Kuralı** | |
| **Kural** | **Kural Adı** | | **Açıklama** |
| **1** | Bilgi | | İlişkisel bir veritabanındaki tüm bilgiler mantıksal olarak tablolardaki satırlarda sütun değerleri olarak temsil edilmelidir. |
| **2** | Garantili erişim | | Bir tablodaki her değere tablo adı, birincil anahtar değeri ve sütun adı kombinasyonuyla erişilebileceği garanti edilir. |
| **3** | Boşların sistematik olarak ele  alınması | | Null'lar, veri türünden bağımsız olarak sistematik bir şekilde temsil edilmeli ve ele alınmalıdır. |
| **4** | İlişkisel modele dinamik çevrimiçi katalog | | Meta veriler sıradan veriler gibi, yani veritabanı içindeki tablolarda saklanmalı ve yönetilmelidir; bu veriler standart  veritabanı ilişkisel dilini kullanan yetkili kullanıcılar tarafından kullanılabilir olmalıdır. |
| **5** | Kapsamlı veri alt dili | | İlişkisel veritabanı birçok dili destekleyebilir; ancak, iyi tanımlanmış, bildirimsel bir dilin yanı sıra veri , görünüm tanımı, veri manipülasyonu (etkileşimli ve program tarafından), bütünlük kısıtlamaları, yetkilendirme ve işlem yönetimini (başlatma, işleme ve geri alma) desteklemelidir. |
| **6** | Güncellemeyi görüntüle | | Teorik olarak güncellenebilir olan herhangi bir görünüm, sistem aracılığıyla güncellenebilir olmalıdır. |
| **7** | Üst düzey ekleme, güncelleme ve silme | | Veritabanı set düzeyinde ekleme, güncelleme ve silme işlemlerini desteklemelidir. |
| **8** | Fiziksel veri bağımsızlığı | | Uygulama programları ve geçici tesisler, fiziksel erişim yöntemleri veya depolama yapıları değiştirildiğinde mantıksal  olarak etkilenmez. |
| **9** | Mantıksal veri bağımsızlığı | | Orijinal tablo değerlerini koruyan tablo yapılarında değişiklik yapıldığında (sütunların sırasını değiştirme veya sütun ekleme) uygulama programları ve geçici tesisler mantıksal olarak etkilenmez. |
| **10** | Dürüstlük bağımsızlık | | Tüm ilişkisel bütünlük kısıtlamaları ilişkisel dilde tanımlanabilir olmalı ve uygulama düzeyinde değil sistem  kataloğunda saklanmalıdır. |
| **11** | Dağıtım bağımsızlığı | | Son kullanıcılar ve uygulama programları veri konumundan (dağıtık yerel veritabanları) habersizdir ve bundan etkilenmez. |
| **12** | Dönüşümsüzlük | | Sistem verilere düşük seviyeli erişimi destekliyorsa, kullanıcıların veritabanının bütünlük kurallarını atlamasına izin verilmemelidir. |
| **13** | Kural sıfır | | Önceki tüm kurallar, bir veritabanının ilişkisel olarak kabul edilebilmesi için ilişkisel olanaklarını yalnızca yönetim için  kullanması gerektiği fikrine dayanmaktadır. |

1Codd, E., "DBMS'niz Gerçekten İlişkisel mi?" ve "DBMS'niz Kurallara Göre mi Çalışıyor?" *Computerworld*, 14 ve 21 Ekim 1985.

# Özet

* Tablolar, ilişkisel bir veri tabanının temel yapı taşlarıdır. Varlık kümesi olarak bilinen ilgili varlıkların bir gruplaması bir tabloda saklanır. Kavramsal olarak, ilişkisel tablo kesişen satırlardan (tuples) ve sütunlardan oluşur. Her satır tek bir varlığı temsil eder ve her sütun varlıkların özelliklerini (niteliklerini) temsil eder.
* Anahtarlar, ilişkisel tabloların kullanımının merkezinde yer alır. Anahtarlar işlevsel bağımlılıkları tanımlar; , diğer nitelikler

anahtara bağlıdır ve bu nedenle anahtar değeri biliniyorsa bulunabilir. Bir anahtar bir üst anahtar, aday anahtar, birincil anahtar, ikincil anahtar veya yabancı anahtar olarak sınıflandırılabilir.

* Her tablo satırının bir birincil anahtarı olmalıdır. Birincil anahtar, herhangi bir satırda bulunan diğer tüm öznitelikleri benzersiz bir şekilde tanımlayan bir öznitelik veya öznitelik kombinasyonudur. Bir birincil anahtarın benzersiz olması gerektiğinden, varlık bütünlüğü korunacaksa null değerlere izin verilmez.
  + Tablolar bağımsız olmalarına rağmen, ortak niteliklerle birbirlerine bağlanabilirler. Böylece, bir tablonun birincil anahtarı, bağlı olduğu başka bir tabloda yabancı anahtar olarak görünebilir. Referans bütünlüğü, yabancı anahtarın ilgili tablodaki birincil anahtarla eşleşen değerler içermesi veya boş değerler içermesi gerektiğini belirtir.
  + İlişkisel model, SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT

dahil olmak üzere çeşitli ilişkisel birleştirme işlevlerini

destekler,

ve DIVIDE. Bu fonksiyonların temel matematiksel formlarının anlaşılması, veri manipülasyon seçeneklerinin daha geniş bir şekilde anlaşılmasını sağlar.

* + İlişkisel bir veritabanı, veri işleme işinin çoğunu perde

arkasında gerçekleştirir. Örneğin, siz

bir veritabanı oluşturduğunuzda, RDBMS otomatik olarak için bir veri sözlüğü barındıracak bir yapı . Veritabanında her yeni tablo oluşturduğunuzda, RDBMS veri sözlüğünü günceller ve böylece veritabanı belgelerini sağlar.

* İlişkisel veritabanlarının temellerini öğrendikten sonra tasarıma odaklanabilirsiniz. İyi tasarım, uygun varlıkları ve bunların niteliklerini ve ardından varlıklar arasındaki ilişkileri belirleyerek başlar. Bu ilişkiler (1:1, 1:M ve M:N) ERD'ler kullanılarak gösterilebilir. ERD'lerin kullanımı basit mantıksal tasarımlar oluşturmanızı ve değerlendirmenizi sağlar. 1:M ilişkisi iyi bir tasarıma en kolay şekilde dahil edilebilir; sadece "1 "in birincil anahtarının "çok" tablosuna dahil edildiğinden emin olun.

# Anahtar Terimler

associative entity attribute domain bridge entity candidate key closure composite entity composite key data dictionary dependent determinant determination DIFFERENCE DIVIDE

etki alanı

varlık bütünlüğü equijoin bayraklar

yabancı anahtar (FK)

tam fonksiyonel bağımlılık fonksiyonel bağımlılık eşanlamlısı

dizin dizin anahtarı iç birleştirme INTERSECT JOIN

sütunlara katıl anahtar anahtar öznitelik sol dış birleştirme bağlantı tablosu doğal birleştirme null

outer join yüklem mantığı birincil anahtar (PK)

ÜRÜN PROJESI

referans bütünlüğü ilişkisel cebir relvar KISITLAMA

sağ dış birleştirme ikincil anahtar SELECT

küme teorisi süper anahtar eşanlamlı sistem kataloğu teta join tuple BİRLİK

sendika uyumlu benzersiz dizin

# İnceleme Soruları

1. Bir veritabanı ile bir tablo arasındaki fark nedir?

**Çevrimiçi İçerik**

Sorularda ve problemlerde kullanılan tüm veritabanları [www.cengage.com](http://www.cengage.com/) adresinde mevcuttur.

Veritabanı adları şekillerde gösterilen veritabanı adlarıyla eşleşmektedir.

1. Bir veritabanının hem varlık bütünlüğü hem de referans

bütünlüğü gösterdiğini söylemek ne anlama gelir?

1. Bir veritabanında varlık bütünlüğü ve referans neden

önemlidir?

1. İki ilişkinin sendika uyumlu olarak kabul edilmesi için karşılaması gereken gereklilikler nelerdir?
2. Birlik uyumlu olmayan bir tablo çiftine hangi ilişkisel
3. π ) (öğrenci) ile sonuçlanacak tabloyu oluşturun.

cebir operatörleri uygulanabilir?

(stu\_code

1. π'den kaynaklanacak tabloyu oluşturun

stu\_code, dept\_code

1. Veri sözlüğüne neden bazen şu adların verildiğini açıklayın (öğrenci⋈ profesör).

"veritabanı tasarımcısının veritabanı."

1. Bir veritabanı kullanıcısı manuel olarak "Dosya iki yüz kayıt içeriyor, her kayıt dokuz alan içeriyor." şeklinde not düşer. Bu ifadeyi "çevirmek" için uygun ilişkisel veritabanı terminolojisini kullanın.

Soru 8-12'yi yanıtlamak için Şekil Q3.8'i kullanın.

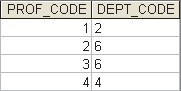
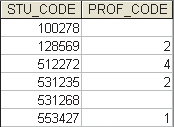
1. STUDENT ve PROFESSOR tablolarını kullanarak, doğal birleştirme, eşit birleştirme ve dış birleştirme arasındaki farkı gösterin.
2. Şekil Q3.8'de gösterilen veritabanı için temel ERD'yi

oluşturun.

1. Şekil Q3.8'de gösterilen veritabanı için ilişkisel diyagramı oluşturun.

**Ch03\_CollegeQue Veritabanı Tabloları**

**Şekil Q3.8**



Veritabanı adı: Ch03\_CollegeQue

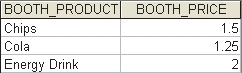
Tablo adı: ÖĞRENCİ

Tablo adı: PROFESÖR

Soru 13-17'yi yanıtlamak için Şekil Q3.13'ü kullanın.

**Ch03\_VendingCo Veritabanı Tabloları**

**Şekil Q3.13**



Masa adı: BOOTH

Veritabanı adı: Ch03\_VendingCo

Tablo adı: MACHINE

1. Şekil Q3.13'te gösterilen tablolara UNION ilişkisel operatörünü uygulamak için ilişkisel cebir formülünü yazınız.
2. Şekil Q3.13'te gösterilen tablolara UNION ilişkisel operatörü uygulandığında ortaya çıkan tabloyu oluşturun.
3. Şekil Q3.13'te gösterilen tablolara bir INTERSECT ilişkisel işleci uygulamak için ilişkisel cebir formülünü yazınız.
4. Şekil Q3.13'te gösterilen tablolara bir INTER- SECT ilişkisel operatörü uygulandığında ortaya çıkan tabloyu oluşturun.
5. Şekil S3.13'teki tabloları kullanarak, MAKİNE FARKI TABLOSU'ndan kaynaklanan tabloyu oluşturun.

Soru 18'i yanıtlamak için Şekil Q3.18'i kullanın.

**Şekil Q3.18 DRIVER ve TRUCK için Karga Ayağı ERD**



SÜRÜC

Ü

sürücül er

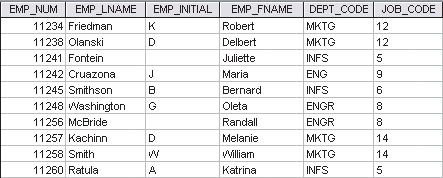
KAMYO N

Belirli bir zaman aralığında, bir SÜRÜCÜ birçok KAMYON'u sürebilir ve herhangi bir KAMYON birçok SÜRÜCÜ tarafından sürülebilir

1. Şekil Q3.18'de gösterilen ERD'ye sahip olduğunuzu varsayalım. Bu modeli yalnızca 1:M ilişkilerini gösteren bir ERM'ye nasıl dönüştürürsünüz? (Revize edilmiş ERD'yi oluşturduğunuzdan emin olun).
2. Eşanlamlılar ve eşanlamlılar nedir ve veritabanı tasarımında neden bunlardan kaçınılmalıdır?
3. İki tablodan oluşan bir veri tabanında l:M ilişkisini nasıl uygularsınız? Bir örnek veriniz.

Soru 21'i yanıtlamak için Şekil Q3.21'i kullanın.

**Şekil Q3.21 Ch03\_NoComp Veritabanı EMPLOYEE Tablosu**



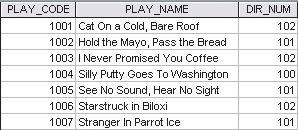
Tablo adı: EMPLOYEE Veritabanı adı: Ch03\_NoComp

1. Şekil Q3.21'de gösterilen tablonun bileşenlerini doğru terminolojiyi kullanarak tanımlayın ve açıklayın. Tablonun olası yabancı anahtar(lar)ını tanımlamak için adlandırma kuralları bilginizi kullanınız.

Soru 22-27'yi yanıtlamak için Şekil Q3.22'de gösterilen

veritabanını kullanın.

**Şekil Q3.22 Ch03\_Theater Veritabanı Tabloları**



Veritabanı adı: Ch03\_Theater

Tablo adı: DİREKTÖR

Tablo adı: PLAY

1. Birincil anahtarları tanımlayın.
2. Yabancı anahtarları tanımlayın.
3. ERM'yi oluşturun.
4. DIRECTOR ve PLAY arasındaki ilişkiyi göstermek için ilişkisel diyagram oluşturun.
5. Belirli bir yönetmen tarafından yönetilen tüm oyunların bir listesini almak için hızlı arama özelliği istediğinizi varsayalım. Hangi

tablosu INDEX tablosunun temeli olur ve indeks anahtarı

ne olur?

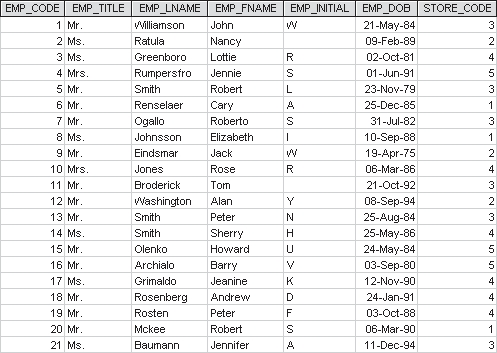
1. Soru 26'da açıklanan INDEX tablosunun kavramsal görünümü ne olurdu? Kavramsal INDEX tablosunun içeriğini tasvir .

# Problemler

Problem 1-9'u yanıtlamak için Şekil P3.1'de gösterilen veritabanını kullanın.

**Ch03\_StoreCo Veritabanı Tabloları**

**Şekil P3.1**



Tablo adı: EMPLOYEE

Veritabanı adı: Ch03\_StoreCo

Tablo adı: STORE

Tablo adı: BÖLGE

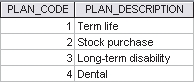
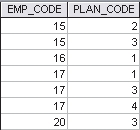
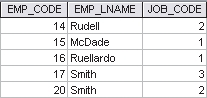
* 1. Her tablo için birincil anahtarı ve yabancı (lar)ı tanımlayın. Bir tablonun yabancı anahtarı yoksa, *Yok* yazın.
  2. Tablolar varlık bütünlüğü sergiliyor mu? Evet veya hayır olarak yanıtlayın ve ardından açıklayın.
  3. Tablolar referans bütünlüğü sergiliyor mu? Evet veya hayır olarak yanıtlayın ve ardından yanıtınızı açıklayın. Tabloda yabancı

anahtar yoksa *NA* (Uygulanamaz) yazın.

* 1. MAĞAZA ve BÖLGE arasındaki ilişki(ler)in tür(ler)ini tanımlayın.
  2. STORE ve REGION arasındaki ilişkiyi göstermek için ERD'yi oluşturun.
  3. STORE ve REGION arasındaki ilişkiyi göstermek için ilişkisel diyagramı oluşturun.
  4. ÇALIŞAN ile MAĞAZA arasındaki ilişki(ler)in tür(ler)ini tanımlayın. (*İpucu:* Her mağaza, biri mağazayı yöneten çok sayıda çalışan istihdam etmektedir).
  5. EMPLOYEE, STORE ve REGION arasındaki ilişkileri göstermek için ERD'yi oluşturun.
  6. ÇALIŞAN, MAĞAZA ve BÖLGE arasındaki ilişkileri göstermek için ilişkisel diyagramı oluşturun.

**Ch03\_BeneCo Veritabanı Tabloları**

**Şekil P3.10**



Veritabanı adı: Ch03\_BeneCo

Tablo adı: EMPLOYEE

Tablo adı: BENEFIT

Tablo adı: JOB

Tablo adı: PLAN

Problem 10-16 üzerinde çalışmak için Şekil P3.10'da gösterilen veritabanını kullanın. Veritabanının bu ilişkileri yansıtan dört tablodan oluştuğunu unutmayın

* Bir ÇALIŞAN yalnızca bir İŞ\_KODU'na sahiptir, ancak bir İŞ\_KODU birçok ÇALIŞAN tarafından tutulabilir.
* Bir ÇALIŞAN birçok PLAN'a katılabilir ve herhangi bir PLAN birçok ÇALIŞAN'a atanabilir.

Ayrıca, M:N ilişkisinin, BENEFIT tablosunun bileşik veya köprü varlık olarak hizmet verdiği iki 1:M ilişkisine bölündüğüne

dikkat edin.

* 1. Veritabanındaki her tablo için birincil anahtarı ve yabancı (lar)ı tanımlayın. Eğer bir tablonun yabancı anahtarı yoksa,

*None* yazın.

* 1. EMPLOYEE ve JOB arasındaki ilişkiyi göstermek için ERD'yi oluşturun.
  2. EMPLOYEE ve JOB arasındaki ilişkiyi göstermek için ilişkisel diyagram oluşturun.
  3. Tablolar varlık bütünlüğü sergiliyor mu? Evet veya hayır olarak yanıtlayın ve ardından açıklayın.
  4. Tablolar referans bütünlüğü sergiliyor mu? Evet veya hayır olarak yanıtlayın ve ardından yanıtınızı açıklayın. Tabloda yabancı

anahtar yoksa *NA* (Uygulanamaz) yazın.

* 1. EMPLOYEE, BENEFIT, JOB ve PLAN arasındaki ilişkileri göstermek için ERD'yi oluşturun.
  2. EMPLOYEE, BENEFIT, JOB ve PLAN arasındaki ilişkileri göstermek için ilişkisel diyagramı oluşturun.