Bölüm 2

Tasarım Konseptleri

Bölüm

3

İlişkisel Veri tabanı Modeli

Öğrenme Hedefleri

Bu bölümü tamamladıktan sonra şunları yapabileceksiniz:

- **3-1** **İlişkisel veritabanı modelinin mantıksal yapısını tanımlama**
- **3-2** **İlişkisel modelin temel bileşenlerini tanımlamak ve ilişkisel bir tablonun yapısını, içeriğini ve özelliklerini açıklamak**
- **3-3** **İlişkisel tablo içeriklerini manipüle etmek için ilişkisel veritabanı operatörlerini kullanma**
- **3-4** **Veri sözlüğü ve sistem kataloğunun amacını ve bileşenlerini açıklamak**
- **3-5** **Uygun varlıkları ve ardından ilişkisel veritabanı modelindeki varlıklar arasındaki ilişkileri tanımlama**
- **3-6** **İlişkisel veritabanı modelinde veri fazlalığının nasıl ele alındığını açıklamak**
- **3-7** **İlişkisel bir veritabanında indekslemenin amacını açıklamak**

ÖN İZLEME

Bu bölümde, ilişkisel modelin mantıksal yapısı ve ilişkisel bir veritabanı tasarlamak için varlık ilişki diyagramlarının (ERD'ler) nasıl kullanılabileceği hakkında daha fazla bilgi edineceksiniz. Ayrıca, ilişkisel veritabanının temel veri bileşenlerinin tablo olarak bilinen mantıksal bir yapıya nasıl uyduğunu ve bir veritabanı içindeki tabloların birbirleriyle nasıl ilişkilendirilebileceğini öğreneceksiniz.

Tablolar, bileşenleri ve ilişkileri hakkında bilgi edindikten sonra, temel tablo tasarımı kavramları ve iyi tasarlanmış ve kötü tasarlanmış tabloların özellikleri ile tanışacaksınız. Bu kavramlar, sonraki birkaç bölüme geçiş kapınız olacak.

Veri Dosyaları ve Mevcut Formatlar

	MS Erişim	Oracle	MS SQL	MySQL
Ch03_CollegeTry	Evet	Evet	Evet	Evet
Ch03_CollegeTry2	Evet	Evet	Evet	Evet
Ch03_InsureCo	Evet	Evet	Evet	Evet
Ch03_Museum	Evet	Evet	Evet	Evet
Ch03_SaleCo	Evet	Evet	Evet	Evet
Ch03_TinyCollege	Evet	Evet	Evet	Evet
Ch03_Relational_DB_Operators	Evet	Evet	Evet	Evet
Ch03_AviaCo	Evet	Evet	Evet	Evet
Ch03_BeneCo	Evet	Evet	Evet	Evet
Ch03_CollegeQue	Evet	Evet	Evet	Evet
Ch03_NoComp	Evet	Evet	Evet	Evet
Ch03_StoreCo	Evet	Evet	Evet	Evet
Ch03_Theater	Evet	Evet	Evet	Evet
Ch03_TransCo	Evet	Evet	Evet	Evet
Ch03_VendingCo	Evet	Evet	Evet	Evet

Veri Dosyaları cengage.com adresinde mevcuttur

yüklem mantığı
Matematikte yaygın olarak kullanılan bir çerçeve sağlamak için iddia (gerçek ifadesi) doğru ya da yanlış olarak doğrulanabilir.

Küme teorisi
Matematik biliminin kümeler veya nesne grupları ile ilgilenen ve ilişkisel modelde veri manipülasyonu için temel olarak kullanılan bir bölümü.

Not

E. F. Codd tarafından 1970 yılında tanıtılan ilişkisel model, yüklem mantığı ve küme teorisine dayanmaktadır. Matematikte yaygın olarak kullanılan **yüklem mantığı**, bir iddianın (gerçek ifadesi) doğru veya yanlış olarak doğrulanabileceği bir çerçeve sağlar. Örneğin, öğrenci kimliği 12345678 olan bir öğrencinin adının Melissa Sanduski olduğunu varsayalım. Bu iddianın doğru ya da yanlış olduğu kolaylıkla gösterilebilir. Küme **teorisi**, kümelerle veya nesne gruplarıyla ilgilenen matematiksel bir bilimdir ve ilişkisel modelde veri manipülasyonu için temel olarak kullanılır. Örneğin, A kümesinin üç sayı içerdiğini varsayalım: 16, 24 ve 77. Bu küme A(16, 24, 77) olarak gösterilir. Ayrıca, B kümesi 44, 77, 90 ve 11 olmak üzere dört sayı içerir ve bu nedenle B(44, 77, 90, 11) olarak gösterilir. Bu bilgiler ışığında, A ve B'nin kesişiminin tek bir sayı olan 77'yi içeren bir sonuç kümesi verdiği sonucuna varabilirsiniz. Bu sonuç A>B

77 şeklinde ifade edilebilir. Başka bir deyişle, A ve B ortak bir değer olan 77'yi paylaşmaktadır.

Bu kavramlar temelinde, ilişkisel modelin iyi tanımlanmış üç bileşeni vardır:

1. İlişkilerle temsil edilen mantıksal bir veri yapısı (bkz. Bölüm 3-1, 3-2 ve 3-5)
2. Verilerin tutarlı olmasını ve zaman içinde tutarlı kalmasını sağlamak için bir dizi bütünlük kuralı (bkz. Bölüm 3-3, 3-6, 3-7 ve 3-8)
3. Verilerin nasıl manipüle edildiğini tanımlayan bir dizi işlem (bkz. Bölüm 3-4)

3-1 Verilere Mantıksal Bir Bakış

Bölüm 1, Veritabanı Sistemleri'nde, bir veritabanının hem verileri hem de meta verileri depoladığını ve yönettiğini öğrendiniz. Ayrıca DBMS'nin verilere ve veritabanı yapısına erişimi yönettiğini ve kontrol ettiğini de öğrendiniz. DBMS'yi uygulama ile veritabanı arasına yerleştiren böyle bir düzenleme, dosya sisteminin kendine özgü sınırlamalarının çoğunu ortadan kaldırır. Ancak bu esnekliğin sonucu çok daha karmaşık bir fiziksel yapıdır. Aslında, hem hiyerarşik hem de ağ veritabanı modellerinin gerektirdiği veritabanı yapıları genellikle verimli veritabanı tasarımını azaltacak kadar karmaşık hale gelir. İlişkisel veri modeli, tasarımcının fiziksel depolama ayrıntıları yerine verilerin mantıksal temsiline ve ilişkilerine odaklanmasına olanak tanıyarak tüm bunları değiştirmiştir. Bir otomotiv benzetmesi yapmak gerekirse, ilişkisel veritabanı sizi debriyaj pedalları ve vites değiştirme işlemleriyle uğraşmaktan kurtarmak için otomatik şanzıman kullanır. Kısacası ilişkisel model, verileri *fiziksel olarak* değil *mantıksal olarak* görmenizi sağlar.

Mantıksal bakıŞ açısını benimsemenin pratik önemi, basit dosya veri depolama konseptini hatırlatmasıdır. Bir tablonun kullanımı, bir dosyanınkinden farklı olarak, yapısal ve veri bağımsızlığı avantajlarına sahip olsa da, bir tablo kavramsal açıdan bir dosyaya benzer. Birbiriyle ilişkili kayıtları bağımsız tablolarda saklanıyormuş gibi düşünebildiğiniz için ilişkisel veritabanı modelini anlamak hiyerarşik ve ağ modellerini anlamaktan çok daha kolaydır. Mantıksal basitlik, basit ve etkili veritabanı tasarım metodolojileri ortaya çıkarma eğilimindedir.

Tablo, ilişkisel modelde bu kadar önemli bir rol oynadığı için daha yakından bakılmayı hak etmektedir. Bu nedenle, tartışmamız tablo yapısı ve içeriğinin ayrıntılarını inceleyerek başlayacaktır.

3-1 a Tablolar ve Özellikleri

İlişkisel veritabanının mantıksal görünümü, ilişki olarak bilinen mantıksal bir yapıya dayalı veri ilişkilerinin oluşturulmasıyla kolaylaştırılır. İlişki matematiksel bir yapı olduğundan, son kullanıcılar bir ilişkiyi tablo olarak düşünmeyi çok daha kolay bulurlar. Bir *tablo*, satır ve sütunlardan oluşan iki boyutlu bir yapı olarak algılanır. İlişkisel modelin yaratıcısı E. F. Codd bu iki terimi eşanlamlı olarak kullandığı için tabloya *ilişki* de denir. Bir tabloyu mantıksal bir ilişkinin *kalıcı* bir temsili olarak düşünebilirsiniz; yani, içeriği gelecekte kullanılmak üzere *kalıcı* olarak kaydedilebilen bir ilişki. Tablonun kullanıcısı söz konusu olduğunda, bir tablo *bir grup ilgili varlık oluşumunu*, yani bir varlık kümesini içerir. Örneğin, bir STUDENT tablosu, her biri bir öğrenciyi temsil eden bir varlık oluşumu koleksiyonu içerir. Bu nedenle, *varlık kümesi* ve *tablo* terimleri genellikle birbirlerinin yerine kullanılır.

Not

Microsoft Access'te *veri* kümesi olarak da bilinen *ilişki* kelimesi, Codd'un modelini türettiği matematiksel küme teorisine dayanmaktadır. İlişkisel model, tablolar arasında ilişki kurmak için öznitelik değerlerini kullandığından, birçok veritabanı kullanıcısı yanlışlıkla *ilişki* teriminin bu tür ilişkilere atıfta bulunduğunu varsaymaktadır. Birçoğu da yanlış bir şekilde sadece ilişkisel modelin ilişkilerin kullanımına izin verdiği sonucuna varmaktadır.

Verilerin tablo görünümünün, varlık ilişkilerini tespit etmeyi ve tanımlamayı kolaylaştırdığını ve böylece veritabanı tasarımı görevini büyük ölçüde basitleştirdiğini keşfedeceksiniz. İlişkisel bir tablonun özellikleri Tablo 3.1'de özetlenmiştir.

tuple
İlişkisel modelde, bir tablo satırı.

öznitelik alanı
Veri modellemede, bir özniteliğin olası değerler kümesini düzenlemek ve tanımlamak için kullanılan yapı.

Tablo İlişkisel Tablonun Özellikleri

1	Bir tablo, satır ve sütunlardan oluşan iki boyutlu bir yapı olarak algılanır.
2	Her tablo satırı (**tuple**) varlık kümesi içinde tek bir varlık oluşumunu temsil eder.
3	Her tablo sütunu bir özniteliği temsil eder ve her sütunun ayrı bir adı vardır.
4	Bir satır ve sütunun her kesişimi tek bir veri değerini temsil eder.
5	Bir sütundaki tüm değerler aynı veri formatına uygun olmalıdır.
6	Her sütun, **öznitelik etki alanı** olarak bilinen belirli bir değer aralığına sahiptir.
7	Satır ve sütunların sırası DBMS için önemsizdir.
8	Her tablo, her satırı benzersiz bir şekilde tanımlayan bir özniteliğe veya öznitelik kombinasyonuna sahip olmalıdır.

Not

İlişkisel veritabanı terminolojisi çok kesindir. Ne yazık ki, dosya sistemi terminolojisi bazen veritabanı ortamına sızar. Bu nedenle, satırlar bazen *kayıtlar* olarak adlandırılır ve sütunlar bazen *alanlar* olarak etiketlenir. Bazen tablolar *dosya* olarak etiketlenir. Teknik olarak konuşursak, terimlerin bu şekilde değiştirilmesi her zaman uygun değildir. Veritabanı tablosu fiziksel bir kavramdan ziyade mantıksal bir kavramdır ve *dosya*, *kayıt* ve *alan* terimleri fiziksel kavramları tanımlar. Bununla birlikte, tablonun aslında fiziksel bir yapıdan ziyade mantıksal bir kavram olduğunu kabul ettiğiniz sürece, tablo satırlarını kayıtlar ve tablo sütunlarını alanlar olarak düşünebilirsiniz. Aslında, birçok veritabanı yazılımı satıcısı hala bu tanıdık dosya sistemi terminolojisini kullanmaktadır.

Şekil 3.1'de gösterilen veritabanı tablosu Tablo 3.1'de listelenen özellikleri göstermektedir.

Şekil 3.1 ÖĞRENCİ Tablo Öznitelik Değerleri

Tablo adı: ÖĞRENCİ — Veritabanı adı: Ch03_TinyCollege

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT	STU_DOB	STU_HRS	STU_CLASS	STU_GPA	STU_TRANSFER	DEPT_CODE	STU_PHONE	PROF_NUM
321452	Bowser	William	C	12-Feb-2000	42	So	2.84	No	BIOL	2134	205
324257	Smithson	Anne	K	15-Nov-2001	81	Jr	3.27	Yes	CIS	2256	222
324258	Brewer	Juliette		23-Aug-1999	36	So	2.26	Yes	ACCT	2256	228
324269	Oblonski	Walter	H	16-Sep-1996	66	Jr	3.09	No	CIS	2114	222
324273	Smith	John	D	30-Dec-1988	102	Sr	2.11	Yes	ENGL	2231	199
324274	Katinga	Raphael	P	21-Oct-2000	114	Sr	3.15	No	ACCT	2267	228
324291	Robertson	Gerald	T	08-Apr-1999	120	Sr	3.87	No	EDU	2267	311
324299	Smith	John	B	30-Nov-2000	15	Fr	2.92	No	ACCT	2315	230

STU_NUM = Öğrenci numarası
STU_LNAME = Öğrenci soyadı
STU_FNAME = Öğrenci ilk adı
STU_INIT = Öğrenci orta baş harfi
STU_DOB = Öğrenci doğum tarihi
STU_HRS = Kazanılan kredi
saatleri **STU_CLASS**= Öğrenci sınıflandırması
STU_GPA = Not ortalaması
STU_TRANSFER = Başka bir kurumdan transfer edilen öğrenci
DEPT_CODE = Departman kodu
STU_PHONE = 4 haneli kampüs telefonu dahili numarası
PROF_NUM = Öğrencinin danışmanı olan profesörün numarası

Şekil 3.1'de gösterilen ÖĞRENCİ tablosunu kullanarak, Tablo 3.1'deki noktalara karşılık gelen aşağıdaki sonuçları çıkarabilirsiniz:

1. STUDENT tablosu 8 satır (tuples) ve 12 sütundan (attributes) oluşan iki boyutlu bir yapı olarak algılanmaktadır.
2. ÖĞRENCİ tablosundaki her satır, varlık kümesi içindeki tek bir varlık oluşumunu tanımlar. (Varlık kümesi STUDENT tablosu tarafından temsil edilir.) Örneğin, Şekil 3.1'deki 4. satır Walter H. Oblonski adlı bir öğrenciyi tanımlar. Tablo içeriği göz önüne alındığında, STUDENT varlık kümesi sekiz farklı varlık (satır) veya öğrenci içerir.
3. Her sütun bir özniteliği temsil eder ve her sütunun ayrı bir adı vardır.
4. Bir sütundaki tüm değerler özniteliğin özellikleriyle eşleşir. Örneğin, not ortalaması (STU_GPA) sütunu, tablo satırlarının her biri için yalnızca STU_GPA girdilerini içerir. Veriler biçim ve işlevlerine göre sınıflandırılmalıdır. Çeşitli DBMS'ler farklı veri türlerini destekleyebilse de, çoğu en azından aşağıdakileri destekler:
   a. *Sayısal*. Anlamlı aritmetik prosedürler gerçekleştirmek için sayısal verileri kullanabilirsiniz. Örneğin, Şekil 3.1'de STU_HRS ve STU_GPA sayısal özniteliklerdir.
   b. *Karakter*. Metin verileri veya dize olarak da bilinen karakter verileri, matematiksel manipülasyon için tasarlanmamış herhangi bir karakter veya sembol . Şekil 3.1'de, STU_CLASS ve STU_PHONE karakter niteliklerine örnektir.
   c. *Tarih*. Tarih , Jülyen tarih biçimi olarak bilinen özel bir biçimde saklanan takvim tarihlerini içerir. Şekil 3.1'de STU_DOB bir tarih özniteliğidir.
   d. *Mantıksal*. Mantıksal veriler yalnızca doğru veya yanlış (evet veya hayır) değerlerine sahip olabilir. Şekil 3.1'de, STU_TRANSFER özniteliği mantıksal bir veri biçimi kullanır.
5. Sütunun izin verilen değerler aralığı **etki alanı** olarak bilinir. STU_GPA değerleri 0-4 aralığıyla sınırlı olduğundan, etki alanı [0,4]'tür.
6. Satır ve sütunların sırası kullanıcı için önemsizdir.
7. Her tablonun bir birincil anahtarı olmalıdır. Genel anlamda **birincil anahtar (PK)**, herhangi bir satırı benzersiz bir şekilde tanımlayan bir öznitelik veya öznitelik kombinasyonudur. Bu durumda, STU_NUM (öğrenci numarası) birincil anahtardır. Şekil 3.1'deki verileri kullanarak, bir öğrencinin soyadının (STU_LNAME) iyi bir birincil anahtar olmayacağını gözlemleyin çünkü birkaç öğrenci Smith soyadına sahiptir. Soyadı ve adın birleşimi (STU_FNAME) bile uygun bir birincil anahtar olmayacaktır çünkü birden fazla öğrencinin adı John Smith'tir.

Çevrimiçi İçerik
Aşağıdaki materyalleri göstermek için kullanılan veritabanları bu bölüm (bkz. Veri Bölümün başındaki listesi) www.cengage.com adresinde mevcuttur. Veritabanı adları şekillerde gösterilen adlarıyla eşleşmektedir.

etki alanı
"Öznitelik etki alanı" olarak da bilinir. Bir için izin verilen değerler kümesidir.

birincil anahtar (PK)
İlişkisel modelde, bir satırı benzersiz bir şekilde tanımlayan bir veya daha fazla öznitelikten oluşan bir tanımlayıcı. Ayrıca, benzersiz bir varlık tanımlayıcısı olarak seçilen bir aday anahtar. Ayrıca bkz. *anahtar*.

3-2 Anahtarlar

İlişkisel modelde, anahtarlar önemlidir çünkü bir tablodaki her satırın benzersiz bir şekilde tanımlanabilmesini sağlamak için kullanılırlar. Ayrıca tablolar arasında ilişki kurmak ve verilerin bütünlüğünü sağlamak için de kullanılırlar. Bir **anahtar**, diğer nitelikleri belirleyen bir veya daha fazla nitelikten oluşur. Örneğin, bir fatura numarası, fatura tarihi ve müşteri adı gibi tüm fatura özniteliklerini tanımlar.

Bir anahtar türü olan birincil anahtar daha önce tanıtılmıştı. Şekil 3.1'de gösterilen STUDENT tablosunun yapısı göz önüne alındığında, birincil anahtarın tanımlanması ve açıklanması yeterince basit görünmektedir. Ancak, birincil anahtar ilişkisel ortamda çok önemli bir rol oynadığı için, birincil anahtarın özelliklerini daha dikkatli bir şekilde inceleyeceksiniz. Bu bölümde ayrıca süper anahtarlar, aday anahtarlar ve ikincil anahtarlarla da tanışacaksınız.

anahtar
Diğer öznitelikleri belirleyen bir veya daha fazla öznitelik. Ayrıca bkz. *aday anahtar*, *yabancı anahtar*, *birincil anahtar (PK)*, *ikincil anahtar* ve *üst anahtar*.

3-2a Bağımlılıklar

Bir anahtarın rolü belirleme kavramına dayanır. **Belirleme**, bir niteliğin değerinin bilinmesinin başka bir niteliğin değerinin belirlenmesini mümkün kıldığı durumdur. Belirleme fikri veritabanı ortamına özgü değildir. Şu formüle aşinasınız

kararlılık
Bir anahtarın rolü. Bir veritabanı tablosu bağlamında, "A, B'yi belirler" ifadesi, A özniteliğinin değerinin bilinmesinin, B özniteliğinin değerine bakılabileceği anlamına geldiğini gösterir.

gelir 2 maliyet 5 kar. Bu bir belirleme şeklidir, çünkü size *gelir* ve *maliyet* verilirse, *karı* belirleyebilirsiniz. *Kâr* ve *gelir* verildiğinde, *maliyeti* belirleyebilirsiniz. Herhangi iki değer verildiğinde, üçüncüyü belirleyebilirsiniz. Ancak bir veritabanı ortamında belirleme normalde bir formüle değil, nitelikler arasındaki ilişkilere dayanır.

Şekil 3.1'deki STUDENT tablosunun niteliklerinin gerçekte neyi temsil ettiğini düşünürseniz, nitelikler arasında bir ilişki görürsünüz. Eğer size STU_NUM için bir değer verilirse, STU_LNAME değerini belirleyebilirsiniz çünkü STU_LNAME'in bir ve yalnızca bir değeri STU_NUM'un herhangi bir değeriyle ilişkilidir. Belirlemeye dayalı ilişkileri tanımlamak için belirli bir terminoloji ve gösterim kullanılır. İlişki şu şekilde adlandırılır **fonksiyonel bağımlılık**, yani bir veya daha fazla özniteliğin değerinin bir veya daha fazla diğer özniteliğin değerini belirlediği anlamına gelir. STU_NUM ve STU_LNAME arasındaki ilişkiyi temsil eden standart gösterim aşağıdaki gibidir:

işlevsel bağımlılık Bir R bağıntısı içinde, bir B niteliği bir A niteliğine ancak ve ancak belirli bir değer "B, A'ya bağlıdır" ilişkisi "A, B'yi belirler" ile eşdeğerdir ve A→ B şeklinde yazılır.

STU_NUM→ STU_LNAME

Bu işlevsel bağımlılıkta, değeri bir diğerini belirleyen özniteliğe **belirleyici** veya anahtar denir. Değeri diğer öznitelik tarafından belirlenen özniteliğe ise **bağımlı** denir. Bu terminolojiyi kullanarak, STU_NUM'ın belirleyici ve STU_LNAME'in bağımlı olduğunu söylemek doğru olacaktır. STU_NUM işlevsel olarak STU_LNAME'i belirler ve STU_LNAME işlevsel olarak STU_NUM'a bağımlıdır. Daha önce belirtildiği gibi, işlevsel bağımlılık birden fazla öznitelik ve birden fazla bağımlı öznitelik içeren bir belirleyici içerebilir. Aşağıdaki örnek için STUDENT tablosuna bakın:

determinant
Belirli bir değeri doğrudan o satırdaki diğer değerleri belirleyen herhangi bir nitelik. Ayrıca bakınız *Boyce-Codd normal formu (BCNF)*.

bağımlı
Değeri başka bir öznitelik tarafından belirlenen bir öznitelik.

STU_NUM→ (STU_LNAME, STU_FNAME, STU_GPA)

ve

(STU_FNAME, STU_LNAME, STU_INIT, STU_PHONE)→ (STU_DOB, STU_HRS, STU_GPA)

Birden fazla nitelikten oluşan belirleyiciler özel dikkat gerektirir. Belirleyicinin ilişki için gerekli olmayan nitelikler içerdiği işlevsel bir bağımlılığa sahip olmak mümkündür. Aşağıdaki iki işlevsel bağımlılığı göz önünde bulundurun:

STU_NUM→ STU_GPA

(STU_NUM, STU_LNAME)→ STU_GPA

İkinci işlevsel bağımlılıkta, belirleyici STU_LNAME'i içerir, ancak bu atıf ilişki için gerekli değildir. İşlevsel bağımlılık geçerlidir çünkü STU_NUM ve STU_LNAME için bir çift değer verildiğinde STU_GPA için yalnızca bir değer oluşacaktır. Daha spesifik bir terim olan **tam işlevsel bağımlılık**, belirleyicideki tüm nitelikler koleksiyonunun ilişki için gerekli olduğu işlevsel bağımlılıkları ifade etmek için kullanılır. Bu nedenle, önceki örnekte gösterilen bağımlılık işlevsel bir bağımlılıktır, ancak tam işlevsel bir bağımlılık değildir.

tam işlevsel bağımlılık
Bir niteliğin işlevsel olarak bileşik bir anahtara bağlı olduğu ancak anahtarın herhangi bir alt kümesine bağlı olmadığı durum.

3-2b Anahtar Türleri

Bir anahtarın, diğer niteliklerin değerlerini belirleyebilen bir nitelik veya nitelik grubu olduğunu hatırlayın. Bu nedenle, anahtarlar işlevsel bağımlılıklarda belirleyicidir. İlişkisel modelde çeşitli anahtar türleri kullanılır ve bunlara aşina olmanız gerekir.

Bileşik anahtar, birden fazla öznitelikten oluşan bir anahtardır. Bir anahtarın parçası olan bir özniteliğe **anahtar özniteliği** denir. Örneğin,

kompozit anahtar
Çok özellikli bir anahtar.

anahtar öznitelik
Bir birincil anahtarın parçası olan bir öznitelik. Ayrıca bkz. *asal öznitelik*.

STU_NUM→ STU_GPA

(STU_LNAME, STU_FNAME, STU_INIT, STU_PHONE)→ STU_HRS

İlk işlevsel bağımlılıkta, STU_NUM yalnızca bir anahtar özniteliğinden oluşan bir anahtar örneğidir. İkinci işlevsel bağımlılıkta, (STU_LNAME, STU_FNAME, STU_INIT, STU_PHONE) dört anahtar öznitelikten oluşan bileşik bir anahtardır.

Tablo Öğrenci Sınıflandırması

Tamamlanan Saatler	Sınıflandırma
30'dan az	Fr
30-59	So
60-89	Jr
90 veya daha fazla	Sr

Bir **üst** anahtar, tablodaki herhangi bir satırı benzersiz bir şekilde tanımlayabilen bir anahtardır. Başka bir deyişle, bir üst anahtar işlevsel olarak satırdaki her özniteliği belirler. STUDENT tablosunda, (STU_NUM, STU_LNAME), (STU_NUM, STU_LNAME, STU_INIT) ve STU_LNAME, STU_FNAME, STU_INIT, STU_PHONE) bileşik anahtarları gibi STU_NUM da bir üst anahtardır.

süper anahtar
Bir tablodaki her bir varlığı benzersiz şekilde tanımlayan öznitelik veya öznitelikler. Bkz. *anahtar*.

Aslında, STU_NUM tek başına bir üst anahtar olduğundan, anahtar niteliği olarak STU_NUM içeren herhangi bir bileşik anahtar da bir üst anahtar olacaktır. Ancak dikkatli olun, çünkü tüm anahtarlar süper anahtar değildir. Örneğin, Gigantic State University öğrenci sınıflandırmasını Tablo 3.2'de gösterildiği gibi tamamlanan saatlere göre belirler.

Bu nedenle, STU_HRS→ STU_CLASS yazabilirsiniz.

Ancak, belirli saat sayısı sınıflandırmaya bağlı değildir. Tamamlanmış saat sayısı 62 olan bir genç de bulmak mümkündür, 84 olan bir genç de. Başka bir deyişle, sınıflandırma (STU_CLASS) tamamlanan saatler (STU_HRS) için tek bir değer belirlemez.

Belirli bir süper anahtar türüne aday anahtar denir. **Aday** anahtar minimal bir üst anahtardır, yani gereksiz nitelikleri olmayan bir üst anahtardır. Bir aday anahtar tam bir işlevsel bağımlılığa dayanır. Örneğin, STU_NUM, (STU_LNAME, STU_FNAME, STU_INIT, STU_PHONE) gibi aday bir anahtar olacaktır. Öte yandan, (STU_NUM, STU_LNAME) bir üst anahtardır, ancak aday anahtar değildir çünkü STU_LNAME kaldırılabilir ve anahtar yine de bir üst anahtar olur. Başka bir deyişle, (STU_NUM, STU_LNAME) tablodaki diğer niteliklerle işlevsel bir bağımlılığa sahiptir, ancak tablodaki diğer niteliklerle *tam* bir işlevsel bağımlılığa sahip değildir. Bir tablonun birçok farklı aday anahtarı olabilir. STUDENT tablosu öğrencilerin Sosyal Güvenlik numaralarını da STU_SSN olarak içeriyorsa, bu bir aday anahtar olarak görünecektir. Aday *anahtarlara* aday denir çünkü bunlar tasarımcının birincil anahtarı seçerken tercih edeceği uygun seçeneklerdir. Birincil anahtar, tablonun satırlarının benzersiz bir şekilde tanımlanmasını sağlayan birincil araç olarak seçilen aday anahtardır. Tüm kanişlerin köpek olması, ancak tüm köpeklerin kaniş olmaması gibi, tüm birincil anahtarlar aday anahtarlardır, ancak tüm aday anahtarlar birincil anahtar olarak seçilmez. Tüm aday anahtarlar süper anahtarlardır, ancak tüm süper anahtarlar aday değildir.

aday anahtarı
Minimal bir üst anahtar; yani, kendisi de bir üst anahtar olan bir öznitelik alt kümesi içermeyen bir anahtar. Bkz. *anahtar*.

Not

Null değer ifade etmez. Sıfır veya boşluk anlamına gelmez. Herhangi bir giriş yapmadan bir sonraki girişe geçmek için Enter tuşuna veya Sekme tuşuna basmak null oluşturulur. Boşluk çubuğuna basmak bir boşluk (veya boşluk) oluşturur.

Varlık bütünlüğü, tablodaki her satırın (**varlık** örneği) kendi bilinen, benzersiz kimliğine sahip olması durumudur. Varlık bütünlüğünü sağlamak için birincil anahtarın iki gereksinimi vardır: (1) birincil anahtardaki tüm değerler benzersiz olmalıdır ve (2) birincil anahtardaki hiçbir anahtar özniteliği null içeremez.

varlık bütünlüğü
Her bir varlığın bir birincil anahtarda benzersiz bir değere sahip olmasını ve anahtarda boş değer bulunmamasını garanti eden ilişkisel tablo özelliği.

Null değerler ilişkisel modelde sorunludur. **Null**, herhangi bir veri değerinin olmamasıdır ve birincil anahtarın herhangi bir bölümünde bulunmasına asla izin verilmez. Teorik açıdan bakıldığında, null içeren bir tablonun tam ilişkisel bir tablo olmadığı söylenebilir. Ancak pratik açıdan bakıldığında, bazı boşluklar makul bir şekilde önlenemez. Örneğin, tüm

boş
Bir öznitelik değerinin olmaması. Null değerinin boşluk olmadığını unutmayın.

öğrencilerin orta harfleri vardır. Genel bir kural olarak, boş karakterlerden mümkün olduğunca kaçınılmalıdır. Aslında, çok sayıda null olması genellikle kötü bir tasarımın işaretidir. Ayrıca, anlamları her zaman tanımlanabilir olmadığı için veritabanında null'lardan kaçınılmalıdır. Örneğin, bir null aşağıdakilerden herhangi birini temsil edebilir:

- Bilinmeyen bir öznitelik değeri
- Bilinen ancak eksik olan bir öznitelik değeri
- "Uygulanamaz" koşulu

Uygulama geliştirme yazılımının gelişmişliğine bağlı olarak, null'lar COUNT, AVERAGE ve SUM gibi fonksiyonlar kullanıldığında sorun yaratabilir. Ayrıca, ilişkisel tablolar birbirine bağlandığında null'lar mantıksal sorunlar yaratabilir.

Tablodaki her satıra benzersiz bir kimlik sağlama rolüne ek olarak, birincil anahtar ilişkisel modelin çalışmasını sağlayan kontrollü fazlalıkta ek bir rol oynayabilir. Bölüm 2, Veri Modelleri'nden, ilişkisel modelin ayırt edici özelliklerinden birinin, tablolar arasındaki ilişkilerin bir kontrollü fazlalık biçimi olarak ortak nitelikler aracılığıyla uygulanması hatırlayın. Örneğin, Şekil 3.2'de ortak bir öznitelik olan VEND_CODE aracılığıyla birbirine bağlanan PRODUCT ve VENDOR tabloları gösterilmektedir. VEND_CODE yabancı öznitelik olarak adlandırılır PRODUCT tablosundaki anahtar. **Yabancı anahtar (FK)**, ortak bir nitelik oluşturmak için başka bir tabloya yerleştirilen bir tablonun birincil anahtarıdır. Şekil 3.2'de, VENDOR'ın birincil anahtarı VEND_CODE, PRODUCT tablosuna yerleştirilmiştir; bu nedenle, VEND_CODE, PRODUCT'ta bir yabancı anahtardır. Tablo nitelikleri için uygun bir adlandırma kuralı kullanmanın bir avantajı, yabancı anahtarları daha kolay tanımlayabilmenizdir. Örneğin, Şekil 3.1'deki STUDENT tablosunda uygun bir adlandırma kuralı kullanıldığından, tablodaki iki yabancı anahtarı (DEPT_CODE ve PROF_NUM) tanımlayabilirsiniz; bu da veritabanında STUDENT ile ilgili iki başka tablonun (DEPARTMENT ve PROFESSOR) varlığına işaret eder.

yabancı anahtar (FK)
Ortak bir öznitelik oluşturmak için bir tablodan başka bir yerleştirilen birincil anahtar. İçindeki değerler referans bütünlüğünü sağlamak için yabancı anahtar kısıtlanmalıdır. *Anahtara* bakın.

Şekil 3.2 Basit Bir İlişkisel Veritabanı Örneği

Tablo adı: PRODUCT
Birincil anahtar: PROD_CODE
Yabancı anahtar: VEND_CODE

Veritabanı adı: Ch03_SaleCo

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	18.63	6	231
SRE-657UG	Rat-tail file	2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	6.79	8	235

link

Tablo adı: VENDOR
Birincil anahtar: VEND_CODE
Yabancı anahtar: yok

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

Not

Teknik olarak, bir veritabanındaki null bir değer değildir ve bir giriş de değildir bir değerin ve bir girişin olmamasıdır. Bununla birlikte, konuşma kolaylığı açısından, veritabanı uzmanları ve veritabanı satıcıları bile veritabanındaki bir null'dan genellikle "null değeri" veya "null girişi" olarak bahseder.

Birincil anahtarın veritabanının bütünlüğünü sağlamada bir rolü olduğu gibi yabancı anahtar. Yabancı anahtarlar, bir varlık örneğine başka bir varlık örneği tarafından yapılan her başvurunun geçerli olması koşulu olan **referans bütünlüğünü** sağlamak için kullanılır. Başka bir deyişle, her yabancı anahtar girişi ya null olmalı ya da ilgili tablonun birincil anahtarında geçerli bir değer olmalıdır. PRODUCT tablosundaki VEND_CODE öğesindeki her girişin ya null ya da VENDOR tablosundaki VEND_CODE öğesinde geçerli bir değer olması nedeniyle PRODUCT tablosunun referans bütünlüğüne sahip olduğuna dikkat edin. PRODUCT tablosundaki bir satır tarafından atıfta bulunulan her satıcı geçerli bir satıcıdır.

Son olarak, **ikincil** anahtar sadece veri alma amacıyla kullanılan ve işlevsel bir bağımlılık gerektirmeyen bir anahtar olarak tanımlanır. Açıkçası, ikincil anahtarlar bu bölümde tartışılan diğer anahtarlardan farklıdır çünkü bağımlı olanın benzersiz bir değerini belirlemek için belirleyici gerektirmezler. Ancak, veritabanı ortamında çok önemlidirler. Müşteri verilerinin, müşteri numarasının birincil anahtar olduğu bir CUSTOMER tablosunda saklandığını varsayalım. Çoğu müşterinin numaralarını hatırlayacağını düşünüyor musunuz? Müşterinin soyadı ve telefon numarası kullanıldığında bir müşteri için veri erişimi daha kolaydır. Bu durumda, birincil anahtar müşteri numarasıdır; ikincil anahtar ise müşterinin soyadı ve telefon numarasının birleşimidir. İkincil anahtarın mutlaka benzersiz bir sonuç vermeyeceğini unutmayın. Örneğin, bir müşterinin soyadı ve ev telefon numarası, bir ailenin birlikte yaşadığı ve bir telefon hattını paylaştığı birkaç eşleşmeyi kolayca verebilir. Daha az verimli bir ikincil anahtar, soyadı ve posta kodu kombinasyonu olabilir; bu da düzinelerce eşleşme sağlayabilir ve daha sonra belirli bir eşleşme için taranabilir.

Bir ikincil anahtarın bir aramayı daraltmadaki etkinliği, anahtarın ne kadar kısıtlayıcı olduğuna bağlıdır. Örneğin, CUS_CITY ikincil anahtarı veritabanı açısından meşru olsa da, milyonlarca olası eşleşmeyi incelemek istemediğiniz sürece *New York* veya *Sydney* öznitelik değerlerinin kullanılabilir bir sonuç üretmesi muhtemel değildir. (Elbette, CUS_CITY, CUS_COUNTRY'den daha iyi bir ikincil anahtardır).

Tablo 3.3 çeşitli ilişkisel veritabanı tablo anahtarlarını özetlemektedir.

referans bütünlüğü
Bağımlı bir tablonun yabancı anahtar girişinin ya boş bir ya da eşleşen bir girişe sahip olmasını gerektiren bir koşul İlgili tablonun birincil anahtarında.

ikincil anahtar
İşlevsel bir bağımlılık gerektirmeyen, yalnızca veri alma amacıyla kullanılan bir anahtar. Örneğin, müşterilerin müşteri numaralarını (birincil anahtar) değil, soyadı, adı, adının baş harfi kombinasyonunu bilmeleri muhtemeldir, ve telefon numarası muhtemelen uygun tablo satırıyla eşleşecektir. Ayrıca bkz. *anahtar*.

Tablo 3.3 İlişkisel Veritabanı Anahtarları

Anahtar Tipi	Tanım
Superkey	Bir tablodaki her satırı benzersiz şekilde tanımlayan bir öznitelik veya öznitelik kombinasyonu
Aday anahtarı	Minimal (indirgenemez) bir üst anahtar; kendisi de bir üst anahtar olan bir öznitelik alt kümesi içermeyen bir üst anahtar
Birincil anahtar	Herhangi bir satırdaki diğer tüm öznitelik değerlerini benzersiz bir şekilde tanımlamak için seçilen bir aday anahtar; boş girişler içeremez
Yabancı anahtar	Bir tablodaki, değerleri başka bir tablodaki birincil anahtarla eşleşmesi veya boş olması gereken bir öznitelik veya öznitelik kombinasyonu
İkincil anahtar	Kesinlikle veri alma amacıyla kullanılan bir öznitelik veya öznitelik kombinasyonu

3-3 Dürüstlük Kuralları

İlişkisel veritabanı bütünlük kuralları iyi veritabanı tasarımının temelidir. İlişkisel veritabanı yönetim sistemleri (RDBMS'ler) bütünlük kurallarını otomatik olarak uygulayabilir, ancak uygulama tasarımınızın da bu bölümde bahsedilen varlık ve referans bütünlüğü kurallarına uygun olduğundan emin olmanız önemlidir. Bu kurallar Tablo 3.4'te özetlenmiştir.

Tablo 3.4'te özetlenen bütünlük kuralları Şekil 3.3'te gösterilmiştir.

Tablo 3.4 Dürüstlük Kuralları

Varlık Bütünlüğü	Açıklama
Gereksinim	Tüm birincil anahtar girdileri benzersizdir ve birincil anahtarın hiçbir parçası boş olamaz.
Amaç	Her satırın bilinen, benzersiz bir kimliği olacaktır ve yabancı anahtar değerleri birincil anahtar değerlerine uygun şekilde referans verebilir.
Örnek	Hiçbir fatura mükerrer numaraya sahip olamaz veya boş olamaz; kısacası, tüm faturalar fatura numaralarıyla benzersiz bir şekilde tanımlanır.
Referans Bütünlüğü	**Açıklama**
Gereksinim	Bir yabancı anahtar, kendi tablosunun birincil anahtarının bir parçası olmadığı sürece boş bir girdiye ya da ilişkili olduğu tablodaki birincil anahtar değeriyle eşleşen bir girdiye sahip olabilir (boş olmayan her yabancı anahtar değeri *mevcut* bir birincil anahtar değerine başvurmalıdır).
Amaç	Amaç, bir yabancı anahtar tarafından yapılan her referansın ilgili birincil anahtara geçerli bir referans olmasını sağlamaktır. Bir niteliğin karşılık gelen bir değere sahip *olmaması* mümkündür, ancak geçersiz bir girdiye sahip olmak imkansız olacaktır; referans bütünlüğü kuralının uygulanması, birincil anahtarı başka bir tabloda zorunlu eşleşen yabancı anahtar değerlerine sahip olan bir tablodaki bir satırın silinmesini imkansız hale getirir.
Örnek	Bir müşterinin henüz atanmış bir satış temsilcisi (numarası) olmayabilir, ancak geçersiz bir satış temsilcisine (numarasına) sahip olması imkansız olacaktır.

Şekil 3.3 Dürüstlük Kurallarının Bir Örneği

Tablo adı: CUSTOMER **Veritabanı adı: Ch03_InsureCo**
Birincil anahtar: CUS_CODE
Yabancı anahtar: ACENTE_KODU

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	05-Apr-2024	502
10011	Dunne	Leona	K	16-Jun-2024	501
10012	Smith	Kathy	W	29-Jan-2025	502
10013	Olowski	Paul	F	14-Oct-2024	
10014	Orlando	Myron		28-Dec-2024	501
10015	O'Brian	Amy	B	22-Sep-2024	503
10016	Brown	James	G	25-Mar-2025	502
10017	Williams	George		17-Jul-2024	503
10018	Farriss	Anne	G	03-Dec-2024	501
10019	Smith	Olette	K	14-Mar-2025	503

Tablo adı: AGENT (yalnızca beş seçili alan gösterilir)
Birincil anahtar: AGENT_CODE
Yabancı anahtar: yok

AGENT_CODE	AGENT_AREACODE	AGENT_PHONE	AGENT_LNAME	AGENT_YTD_SLS
501	713	228-1249	Alby	132735.75
502	615	882-1244	Hahn	138967.35
503	615	123-5589	Okon	127093.45

bayraklar
Tasarımcılar tarafından tetiklemek için uygulanan özel kodlar
Gerekli bir yanıt, son kullanıcıları belirtilen koşullara karşı uyarma veya değerleri kodlama. Bayraklar, bir değerin yokluğuna dikkat boş değerleri önlemek için kullanılabilir. bir masa.

Şekil 3.3'ün aşağıdaki özelliklerine dikkat ediniz.

- Varlık bütünlüğü. CUSTOMER tablosunun birincil anahtarı CUS_CODE'dur. CUSTOMER birincil anahtar sütununda boş giriş yoktur ve tüm girişler benzersizdir. Benzer şekilde, AGENT tablosunun birincil anahtarı AGENT_CODE'dur ve bu birincil anahtar sütununda da boş giriş yoktur.

- Referans bütünlüğü. CUSTOMER tablosu, CUSTOMER tablosundaki girdileri AGENT tablosuna bağlayan AGENT_CODE adlı bir yabancı anahtar içerir. (Birincil anahtar) 10013 numarasıyla tanımlanan CUS_CODE satırı, AGENT_CODE yabancı anahtarında boş bir giriş içerir çünkü Paul F. Olowski'henüz kendisine atanmış bir satış temsilcisi yoktur. CUSTOMER tablosundaki kalan AGENT_CODE girişlerinin tümü AGENT tablosundaki AGENT_CODE girişleriyle eşleşir.

Boş değerlerden kaçınmak için, bazı tasarımcılar bazı değerlerin belirtmek üzere bayrak olarak bilinen özel kodlar kullanırlar. Şekil 3.3'ü örnek olarak kullanırsak, -99 kodu AGENT_CODE olarak kullanılabilir

Müşteri Paul Olowski'nin henüz kendisine atanmış bir temsilci olmadığını belirtmek için MÜŞTERİ tablosunun dördüncü satırındaki giriş. Böyle bir bayrak kullanılırsa, ACENTE tablosu -99 AGENT_CODE değerine sahip bir kukla satır içermelidir. Böylece AGENT tablosunun ilk kaydı Tablo 3.5'te gösterilen değerleri içerebilir.

Tablo 3.5 Bayrak Olarak Kullanılan Bir Kukla Değişken Değeri

AGENT_CODE	AGENT_AREACODE	ACENTE_TELEFONU	AGENT_LNAME	AGENT_YTD_SLS
-99	000	000-0000	Hiçbiri	$0.00

Bölüm 4, Varlık İlişkisi (ER) Modelleme, null'ları ele almanın çeşitli yollarını tartışmaktadır.

İlişkisel modelde uygulanabilecek diğer bütünlük kuralları NOT NULL ve UNIQUE kısıtlamaları. NOT NULL kısıtı, tablodaki her satırın o sütun için bir değere sahip olmasını sağlamak için bir sütuna yerleştirilebilir. UNIQUE kısıtı, bir sütun için yinelenen değerlerin bulunmamasını sağlamak üzere sütuna yerleştirilen bir kısıtlamadır.

3-4 İlişkisel Cebir

İlişkisel tablolardaki veriler, yararlı bilgiler üretmek için manipüle edilemediği sürece sınırlı bir değere sahiptir. Bu bölümde ilişkisel modelin temel veri manipülasyon yetenekleri açıklanmaktadır. **İlişkisel cebir,** ilişkisel operatörleri kullanarak tablo içeriklerini manipüle etmenin teorik yolunu tanımlar. Bölüm 7, Yapılandırılmış Sorgu Diline (SQL) Giriş ve Bölüm 8, Gelişmiş SQL'de SQL komutlarının ilişkisel cebir işlemlerini gerçekleştirmek için nasıl kullanılabileceğini öğreneceksiniz.

ilişkisel cebir
İlişkisel tablo içeriklerinin manipüle edilmesi için temel oluşturan bir dizi matematiksel ilke; sekiz ana fonksiyon SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT ve DIVIDE'dır.

Not

İlişkisel tamlık derecesi, ilişkisel cebirin ne ölçüde desteklendiğine göre tanımlanabilir. Asgari düzeyde ilişkisel olarak kabul edilebilmesi için, VTYS'nin temel ilişkisel operatörler olan SELECT, PROJECT ve JOIN'i desteklemesi gerekir.

3-4a Resmi Tanımlar ve Terminoloji

İlişkisel modelin aslında matematiksel ilkelere dayandığını ve veritabanındaki verilerin işlenmesinin matematiksel terimlerle tanımlanabileceğini hatırlayın. İyi haber şu ki, veritabanı uzmanları olarak verilerinizle çalışmak için matematiksel formüller yazmak zorunda değilsiniz. Veriler, veritabanı geliştiricileri ve programcıları tarafından SQL gibi altta yatan matematiği gizleyen güçlü diller kullanılarak manipüle edilir. Bununla birlikte, temel ilkeleri anlamak, gerçekleştirilebilecek işlem türleri hakkında size iyi bir fikir verebilir ve sorgularınızı nasıl daha verimli ve etkili bir şekilde yazacağınızı anlamanıza yardımcı olabilir.

İşlemlerin resmi matematiksel gösterimlerini kullanmanın bir avantajı, matematiksel ifadelerin kesin olmasıdır. Bu ifadeler çok spesifiktir ve veritabanı tasarımcılarının bunları açıklamak için kullanılan dilde spesifik olmalarını gerektirir. Daha önce açıklandığı gibi, *ilişki* ve *tablo* terimlerinin birbirinin yerine kullanılması yaygındır. Ancak, matematiksel terimlerin kesin olması gerektiğinden, bu kitap çeşitli ilişkisel cebir operatörlerinin resmi tanımlarını tartışırken daha spesifik olan ilişki terimini kullanmaktadır.

Belirli ilişkisel cebir operatörlerini ele almadan önce, bir tablo anlayışınızı resmileştirmeniz gerekir.

Özel *ilişki* terimini kullanmanın önemli bir yönü, ilişki ile ilişki değişkeni ya da kısaca *relvar* arasındaki ayrımı kabul etmesidir. İlişki, tablolarınızda gördüğünüz verilerdir. Bir **relvar**, bir ilişkiyi tutan bir değişkendir. Örneğin, şöyle düşünün

relvar
İlişki değişkeninin kısaltması, bir ilişkiyi tutan bir değişken. Bir relvar, ilişkinin kendisini değil, ilişki verilerini tutmak için kullanılan bir kaptır (değişkendir).

Bir program yazıyordunuz ve tamsayı verilerini tutmak için *qty* adında bir değişken oluşturdunuz. *qty* değişkeninin kendisi bir tamsayı değildir; tamsayıları tutmak için bir kaptır. Benzer şekilde, bir tablo oluşturduğunuzda, tablo yapısı tablo verilerini tutar. Yapıya uygun şekilde relvar denir ve yapıdaki veriler bir ilişki olur. Relvar, ilişkinin kendisini değil, ilişki verilerini tutmak için kullanılan bir kaptır (değişkendir). Tablodaki veriler bir ilişkidir.

Bir relvarın iki bölümü vardır: başlık ve gövde. Relvar başlığı niteliklerin içerirken, relvar gövdesi ilişkiyi içerir. Formüllerde bu ayrımı uygun bir şekilde korumak için, belirtilmemiş bir ilişkiye genellikle küçük harf (örneğin, "r") atanırken, relvar'a büyük harf (örneğin, "R") atanır. Bu durumda *r*'nin *R* türünde bir bağıntı ya da *r(R)* olduğunu söyleyebilirsiniz.

3-4b İlişkisel Küme Operatörleri

kapatma
Yeni ilişkiler üretmek için mevcut tablolar (ilişkiler) üzerinde ilişkisel cebir operatörlerinin kullanılmasına izin veren ilişkisel operatörlerin bir özelliği.

İlişkisel operatörler **kapanma** özelliğine sahiptir; yani, ilişkisel cebir operatörlerinin mevcut ilişkiler (tablolar) üzerinde kullanılması yeni ilişkiler üretir. Çok sayıda operatör tanımlanmıştır. Bazı operatörler temeldir, diğerleri ise kullanışlıdır ancak temel operatörler kullanılarak türetilebilir. Bu bölümde SELECT (veya RESTRICT), PROJECT, UNION, INTERSECT, DIFFERENCE, PRODUCT, JOIN ve DIVIDE operatörlerine odaklanılacaktır.
Operatörler.

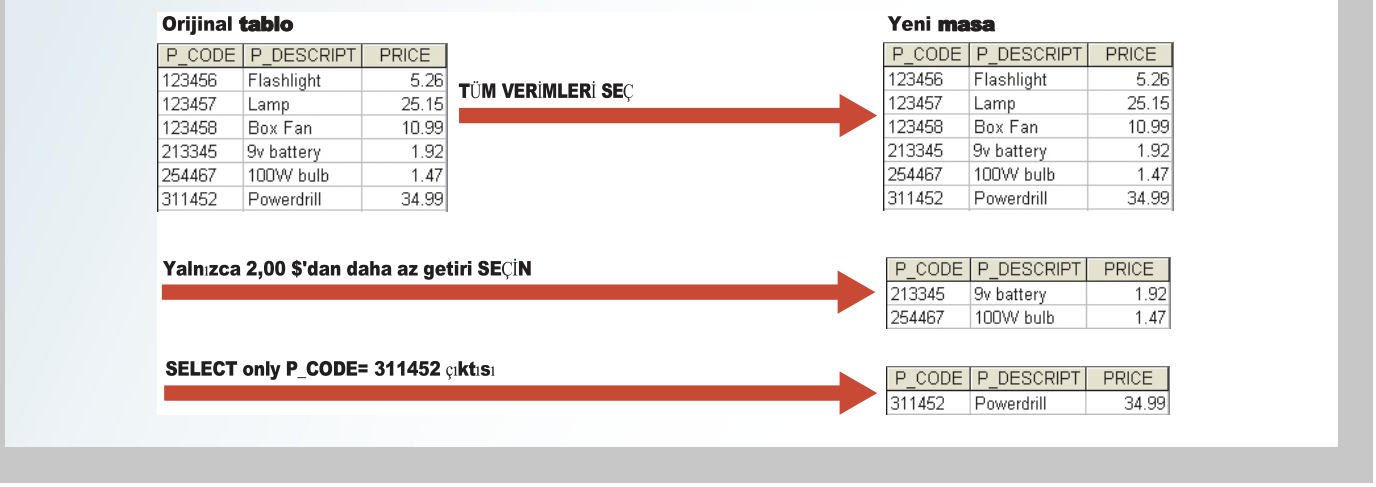
Seç (Kısıtla)

SEÇİNİZ
İlişkisel cebirde, satırların bir alt kümesini seçmek için kullanılan bir operatör. *RESTRICT* olarak da bilinir.

KISITLAMA
SELECT'e bakın.

RESTRICT olarak da bilinen **SELECT**, girdi olarak yalnızca bir tablo kullandığı için unary operatör olarak adlandırılır. Tabloda bulunan ve belirli bir koşulu karşılayan tüm satırlar için değerler verir. SELECT tüm satırları listelemek için kullanılabilir veya yalnızca belirli bir kriterle eşleşen satırları verebilir. Başka bir deyişle, SELECT bir tablonun yatay bir alt kümesini verir. SELECT döndürülen öznitelikleri sınırlamaz, böylece tablonun tüm öznitelikleri sonuca dahil edilir. Bir SELECT işleminin etkisi Şekil 3.4'te gösterilmektedir.

Şekil 3.4 Seçiniz



Not

Biçimsel olarak, SELECT küçük Yunan harfi sigma (σ) ile gösterilir. Sigma'nın alt simge olarak değerlendirilecek koşul (yüklem olarak adlandırılır) gelir ve ardından parantez içinde ilişki listelenir. Örneğin, CUS_CODE özniteliğinde "10010" değerine sahip olan CUSTOMER tüm satırları SEÇMEK için aşağıdakileri yazarsınız:

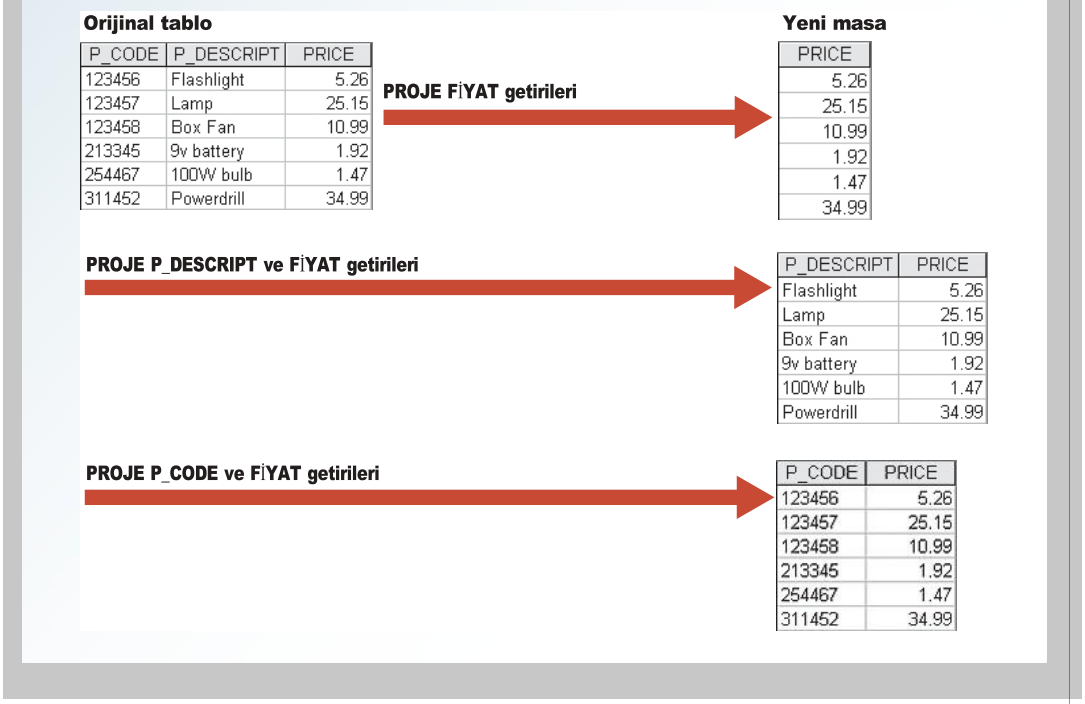
scus_ kodu 5 10010 (müşteri)

Proje

PROJECT seçilen öznitelikler için tüm değerleri verir. Aynı zamanda tek değişkenli bir ve girdi olarak yalnızca bir tablo kabul eder. PROJECT yalnızca istenen öznitelikleri, istendikleri sırada döndürür. Başka bir deyişle, PROJECT bir tablonun dikey bir alt kümesini verir. PROJECT döndürülen satırları sınırlamaz, bu nedenle belirtilen özniteliklerin tüm satırları sonuca dahil edilir. PROJECT işleminin etkisi Şekil 3.5'te gösterilmektedir.

PROJE
İlişkisel cebirde, sütunların bir alt kümesini seçmek için kullanılan bir operatör.

Şekil 3.5 Proje



Not

PROJE resmi olarak Yunanca pi (π) harfi ile gösterilir. Bazı kaynaklar büyük harfi kullanırken, diğer kaynaklar küçük kullanır. Codd, ilişkisel model hakkındaki orijinal makalesinde küçük π harfini kullanmıştır ve biz de burada bunu kullanıyoruz. Pi'nin ardından alt simge olarak döndürülecek niteliklerin listesi ve ardından parantez içinde listelenen ilişki gelir. Örneğin, CUS_FNAME ve CUS_LNAME özniteliklerini CUSTOMER tablosunda PROJECT etmek için aşağıdakileri yazarsınız:

$$\pi_{cus\_fname,\ cus\_lname}(müşteri)$$

İlişkisel operatörler kapanma özelliğine sahip olduklarından, yani ilişkileri girdi olarak kabul edip çıktı olarak ilişkiler ürettiklerinden, operatörleri birleştirmek mümkündür. Örneğin, müşteri kodu 10010 olan müşterinin adını ve soyadını bulmak için önceki iki operatörü birleştirebilirsiniz:

$$\pi_{cus\_fname,\ cus\_lname}(\sigma_{cus\_code\ =\ 10010}(müşteri))$$

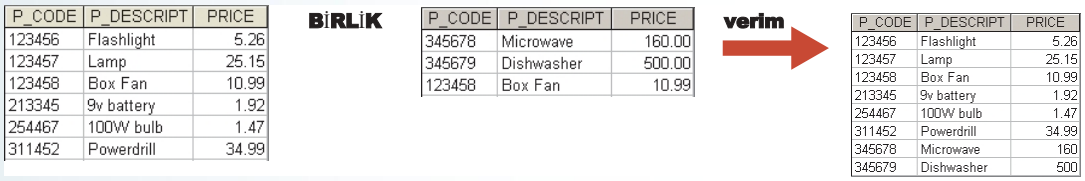
Birlik

UNION, yinelenen satırlar hariç olmak üzere iki tablodaki tüm satırları birleştirir. UNION'da kullanılabilmesi için, tabloların aynı öznitelik özelliklerine sahip olması gerekir; başka bir deyişle, sütunlar ve etki alanları uyumlu olmalıdır. İki veya daha fazla tablo aynı sayıda sütunu paylaştığında ve karşılık gelen sütunları aynı veya uyumlu etki alanlarını paylaştığında, bunların **union uyumlu** olduğu söylenir. Bir UNION işleminin etkisi Şekil 3.6'da gösterilmektedir.

BİRLİK
İlişkisel cebirde, iki tabloyu yeni bir tabloda birleştirmek (eklemek) ve yinelenen satırları atmak için kullanılan bir operatör. Tablolar *birlik uyumlu* olmalıdır.

sendika uyumlu
Aynı sayıda sütuna ve karşılık gelen sütunlara sahip iki veya daha fazla tablo uyumlu etki alanlarına sahiptir.

Şekil 3.6



P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

P_CODE	P_DESCRIPT	PRICE
345678	Microwave	160.00
345679	Dishwasher	500.00
123458	Box Fan	10.99

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99
345678	Microwave	160
345679	Dishwasher	500

Not

UNION< sembolü ile gösterilir. TEDARİKÇİ ve SATICI ilişkileri birlik uyumlu ise, aralarındaki bir BİRLEŞİM aşağıdaki gibi gösterilir:

tedarikçi<satıcı

Bir veritabanında birlik uyumlu iki ilişki bulmak oldukça sıra dışıdır. Tipik olarak, PROJECT operatörleri ilişkilere uygulanarak birlik uyumlu sonuçlar üretilir. Örneğin, SUPPLIER ve VENDOR tablolarının birlik uyumlu olmadığını varsayalım. Tüm satıcı ve tedarikçi adlarının bir listesini üretmek istiyorsanız, her tablodaki adları PROJECT edebilir ve ardından bunlarla bir UNION

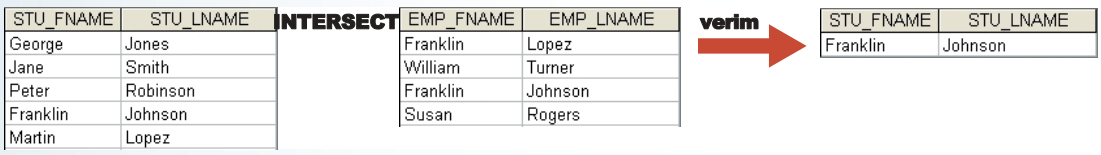
$$_{\text{psupplier\_}}\ (\text{tedarikçi}) <_{\text{pvendor\_name}} (\text{satıcı})$$

Kesişme

INTERSECT
İlişkisel cebirde, yalnızca iki birleştirme uyumlu tabloda ortak olan satırları vermek için kullanılan bir operatör.

INTERSECT yalnızca her iki tabloda da görünen satırları verir. UNION'da olduğu gibi, tabloların geçerli sonuçlar vermesi için union uyumlu olması gerekir. Örneğin, özniteliklerden biri sayısal diğeri karakter tabanlı ise INTERSECT kullanamazsınız. Satırların her iki tabloda da aynı kabul edilmesi ve INTERSECT işleminin sonucunda görünmesi için, satırların tamamının tam kopyalar olması gerekir. Bir INTERSECT işleminin etkisi Şekil 3.7'de gösterilmektedir.

Şekil 3.7 Kesişme



STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

STU_FNAME	STU_LNAME
Franklin	Johnson

Not

INTERSECT> sembolü ile gösterilir. TEDARİKÇİ ve SATICI ilişkileri birlik uyumlu ise, aralarındaki bir KESİŞME aşağıdaki gibi gösterilir:

tedarikçi>satıcı

UNION operatöründe olduğu gibi, bir veritabanında union uyumlu iki ilişki bulmak alışılmadık bir durumdur, bu nedenle PROJECT operatörleri, INTERSECT operatörüyle manipüle edilebilecek sonuçlar üretmek için ilişkilere uygulanır. Örneğin, yine SUPPLIER ve VENDOR tablolarının birlik uyumlu olmadığını varsayalım. Her iki tabloda da aynı olan satıcı ve tedarikçi adlarının bir listesini üretmek istiyorsanız, her tablodaki adları PROJECT edebilir ve ardından bunlarla bir INTERSECT gerçekleştirebilirsiniz.

$$_{\text{psupplier\_ name}}\ (\text{tedarikçi}) >_{\text{pvendor\_name}} (\text{satıcı})$$

Farklar

DIFFERENCE, bir tablodaki diğer bulunmayan tüm satırları verir; , bir tabloyu diğerinden çıkarır. UNION'da olduğu gibi, geçerli sonuçlar elde etmek için tabloların union uyumlu olması gerekir. Bir DIFFERENCE işleminin etkisi Şekil 3.8'de gösterilmektedir. Ancak, ilk tabloyu ikinci tablodan çıkarmanın, ikinci tabloyu ilk tablodan çıkarmakla aynı şey olmadığına dikkat edin.

FARKLILIK
İlişkisel cebirde, bir tablodaki tüm satırların başka bir birleşim uyumlu tabloda bulunmamasını sağlamak için kullanılan bir operatör.

Şekil 3.8 Farklar

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

FARKLILIK

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

verim

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Martin	Lopez

Not

FARK, eksi sembolü - ile gösterilir. Eğer TEDARİKÇİ ve SATICI ilişkileri birleşik uyumlu ise,
TEDARİKÇİ eksi SATICI FARKI aşağıdaki gibi yazılır:

tedarikçi 2 satıcı

SUPPLIER ve VENDOR tablolarının birlik uyumlu olmadığını ve satıcı adı olarak görünmeyen tedarikçi adlarının bir listesini varsayarsak, bir FARK operatörü kullanabilirsiniz.

$$\Pi_{\text{tedarikçi\_ismi}}(\text{tedarikçi}) - \Pi_{\text{satıcı\_ismi}}(\text{satıcı})$$

Ürün

PRODUCT, Kartezyen ürün olarak da bilinen iki tablodaki tüm olası satır çiftlerini verir. Bu nedenle, bir tabloda 6 satır ve diğer tabloda 3 satır varsa, PRODUCT işlemi 6 3 3 5 18 satırdan oluşan liste verir. Bir PRODUCT işleminin etkisi Şekil 3.9'da gösterilmiştir.

ÜRÜN
İlişkisel cebirde, iki tablodan tüm olası satır çiftlerini elde etmek için kullanılan bir operatör. *Kartezyen çarpım* olarak da bilinir.

Şekil 3.9 Ürün

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

ÜRÜN

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

verim

P_CODE	P_DESCRIPT	PRICE	STORE	AISLE	SHELF
123456	Flashlight	5.26	23	W	5
123456	Flashlight	5.26	24	K	9
123456	Flashlight	5.26	25	Z	6
123457	Lamp	25.15	23	W	5
123457	Lamp	25.15	24	K	9
123457	Lamp	25.15	25	Z	6
123458	Box Fan	10.99	23	W	5
123458	Box Fan	10.99	24	K	9
123458	Box Fan	10.99	25	Z	6
213345	9v battery	1.92	23	W	5
213345	9v battery	1.92	24	K	9
213345	9v battery	1.92	25	Z	6
311452	Powerdrill	34.99	23	W	5
311452	Powerdrill	34.99	24	K	9
311452	Powerdrill	34.99	25	Z	6
254467	100W bulb	1.47	23	W	5
254467	100W bulb	1.47	24	K	9
254467	100W bulb	1.47	25	Z	6

BİRLEŞİN
İlişkisel cebirde, kriterlere dayalı iki tablodan satır elde etmek için kullanılan bir operatör türü. Doğal birleştirme, teta birleştirme, eşit birleştirme ve dış gibi birçok birleştirme türü vardır.

Doğal birleştirme
Yeni bir tablo ilişkisel bir işlem yalnızca ortak öznitelik(ler)inde ortak değerlere sahip satırlardan oluşur.

Not

ÜRÜN çarpma sembolü X ile gösterilir. MÜŞTERİ ve ACENTE ilişkilerinin ÜRÜNÜ aşağıdaki gibi yazılacaktır:

müşteri X temsilcisi

Kartezyen çarpım, bir kümenin her üyesinin başka bir kümenin her üyesiyle eşleştirildiği bir dizi üretir. Bağıntılar açısından bu, bir bağıntıdaki her ikilinin ikinci bağıntıdaki her ikiliyle eşleştirildiği anlamına gelir.

Katılın

JOIN, bilgilerin iki veya daha fazla tablodan akıllıca birleştirilmesini sağlar. JOIN, ilişkisel veritabanının arkasındaki gerçek güçtür ve ortak niteliklerle birbirine bağlanmış bağımsız tabloların kullanılmasına izin verir. Şekil 3.10'da gösterilen CUSTOMER ve AGENT tabloları çeşitli birleştirme türlerini göstermek için kullanılacaktır.

Şekil 3.10 JOIN Çizimlerinde Kullanılacak İki Tablo

Tablo adı: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Tablo adı: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

Doğal birleştirme, yalnızca ortak öznitelik(ler)inde ortak değerlere sahip satırları seçerek tabloları birbirine bağlar. Doğal birleştirme üç aşamalı bir sürecin sonucudur:

1. İlk olarak, tabloların bir ÜRETİMİ oluşturulur ve Şekil 3.11'de gösterilen sonuçlar elde edilir.

Şekil 3.11 Doğal Birleştirme, Adım 1: ÜRÜN

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1132445	Walker	32145	231	125	6152439887
1132445	Walker	32145	231	167	6153426778
1132445	Walker	32145	231	231	6152431124
1132445	Walker	32145	231	333	9041234445
1217782	Adares	32145	125	125	6152439887
1217782	Adares	32145	125	167	6153426778
1217782	Adares	32145	125	231	6152431124
1217782	Adares	32145	125	333	9041234445
1312243	Rakowski	34129	167	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1312243	Rakowski	34129	167	231	6152431124
1312243	Rakowski	34129	167	333	9041234445
1321242	Rodriguez	37134	125	125	6152439887
1321242	Rodriguez	37134	125	167	6153426778
1321242	Rodriguez	37134	125	231	6152431124
1321242	Rodriguez	37134	125	333	9041234445
1542311	Smithson	37134	421	125	6152439887
1542311	Smithson	37134	421	167	6153426778
1542311	Smithson	37134	421	231	6152431124
1542311	Smithson	37134	421	333	9041234445
1657399	Vanloo	32145	231	125	6152439887
1657399	Vanloo	32145	231	167	6153426778
1657399	Vanloo	32145	231	231	6152431124
1657399	Vanloo	32145	231	333	9041234445

2. İkinci olarak, yalnızca aşağıdaki satırları elde etmek için Adım 1'in çıktısı üzerinde bir SELECT gerçekleştirilir AGENT_CODE değerleri eşittir. Ortak sütunlar **birleştirme sütunları** olarak adlandırılır. Adım 2, Şekil 3.12'de gösterilen sonuçları verir.

sütunlara katıl
Birleştirme işlemlerinin ölçütlerinde kullanılan sütunlar. Birleştirme sütunları genellikle benzer değerleri paylaşır.

Şekil 3.12 Doğal Birleştirme, Adım 2: SELECT

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124

3. Her bir özniteliğin tek bir kopyasını elde etmek için Adım 2'nin sonuçları üzerinde bir PROJECT gerçekleştirilir ve böylece yinelenen sütunlar ortadan kaldırılır. Adım 3, Şekil 3.13 'te gösterilen çıktıyı verir.

Şekil 3.13 Doğal Birleştirme, Adım 3: PROJE

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124

Doğal bir birleştirmenin nihai sonucu, eşleşmeyen çiftleri içermeyen ve yalnızca eşleşenlerin kopyalarını sağlayan bir tablo verir.

Doğal birleştirme işleminin birkaç önemli özelliğine dikkat edin:

- Tablo satırları arasında eşleşme yapılmazsa, yeni tablo eşleşmeyen satırı içermez. Bu durumda, ne AGENT_CODE 421 ne de soyadı Smithson olan müşteri dahil edilir. Smithson'ın AGENT_CODE 421'i AGENT tablosundaki hiçbir girişle eşleşmez.
- Birleştirmenin yapıldığı sütun (AGENT_CODE) yeni tabloda yalnızca bir kez yer alır.
- Aynı AGENT_CODE'un AGENT tablosunda birkaç kez geçmesi durumunda, her eşleşme için bir müşteri listelenecektir. Örneğin, AGENT_CODE 167 AGENT tablosunda üç kez geçerse, Rakowski adlı müşteri de ortaya çıkan tabloda üç kez geçecektir çünkü Rakowski AGENT_CODE 167 ile ilişkilidir. (Elbette, iyi bir AGENT tablosu benzersiz birincil anahtar değerleri içereceğinden böyle bir sonuç veremez).

Not

Doğal birleştirme normalde resmi işlemlerde sadece JOIN olarak adlandırılır. JOIN, ⋈ sembolü ile gösterilir. CUSTOMER ve AGENT ilişkilerinin JOIN'i aşağıdaki gibi yazılabilir:

müşteri ⋈ temsilcisi

İki ilişkinin JOIN'inin, ortak özniteliğin yalnızca bir kopyasının döndürülmesi dışında, her iki ilişkinin tüm özniteliklerini döndürdüğüne dikkat edin. Biçimsel olarak bu, relvar başlıklarının UNION'ı olarak tanımlanır. Bu nedenle, ilişkilerin JOIN'i (c⋈ a) relvarların UNION'ını (C<A) içerir. Ayrıca, daha önce açıklandığı gibi, JOIN'in temel bir ilişkisel cebir operatörü olmadığını . Diğer operatörlerden aşağıdaki şekilde türetilebilir:

pcus_ kodu , cus _adı , cus _adı , cus _ilk, cus _yenileme _tarihi, acente_ kodu , acente _areakodu, acente_telefonu, acente_adı, acente_ ytd_ sls

$(_{\text{scustomer.agent\_code 5 agent.agent\_code}}(\text{customer } 3))$

equijoin
'nin belirtilen sütunlarını karşılaştıran bir eşitlik koşuluna dayalı olarak tabloları bağlayan bir birleştirme işleci masaları.

teta birleştirme
Eşitsizlik karşılaştırma işleci kullanarak tabloları birbirine bağlayan bir birleştirme işleci
(<, >, <5, >5) birleştirme koşulunda.

Eşit birleştirme olarak bilinen başka bir birleştirme biçimi, tabloları her tablonun belirtilen sütunlarını karşılaştıran bir eşitlik koşulu temelinde birbirine bağlar. Equijoin'in sonucu yinelenen sütunları ortadan kaldırmaz ve tabloları birleştirmek için kullanılan koşul veya ölçüt açıkça tanımlanmalıdır. Aslında, bir equijoin'in sonucu Şekil 3.12'de doğal bir birleştirmenin 2. Adımı için gösterilen sonuca benzer. Equijoin adını koşulda kullanılan eşitlik karşılaştırma operatöründen (5) alır. Başka bir karşılaştırma operatörü kullanılırsa, birleştirme **teta** birleştirme olarak adlandırılır.

Not

Biçimsel açıdan, teta doğal birleştirmenin bir uzantısı olarak kabul edilir. Teta birleştirme, JOIN sembolünden sonra bir teta alt simgesi eklenerek gösterilir: ⋈ . Equijoin, teta birleştirmenin özel bir türüdür.

iç birleştirme
Yalnızca belirli bir kriteri karşılayan satırların seçildiği bir birleştirme işlemi. Ölçüt bir eşitlik koşulu (doğal birleştirme veya equijoin) veya bir eşitsizlik koşulu (theta join) olabilir. En yaygın kullanılan türüdür.

dış birleştirme
Tüm eşleşmeyen çiftlerin tutulduğu bir tablo üreten bir birleştirme işlemi; ilgili tablodaki eşleşmeyen değerler
Sol boş.

sol dış birleştirme
Diğer tabloda eşleşen değerlere sahip da dahil olmak üzere soldaki tablodaki tüm satırları veren bir birleştirme işlemi.

sağ dış birleştirme
Diğer tabloda eşleşen değerlere sahip olmayanlar da dahil olmak üzere doğru tüm satırları veren bir işlemi.

Önceki her biri genellikle bir iç birleştirme olarak sınıflandırılır. Bir **iç birleştirme** yalnızca birleştirilen tablolardan eşleşen kayıtları döndürür. Bir **dış birleştirmede**, eşleşen çiftler korunur ve diğer tablodaki eşleşmeyen değerler boş bırakılır. Dış birleştirmenin iç birleştirmenin tersi olduğunu düşünmek kolay bir hatadır. Ancak, dış birleştirmeyi "iç birleştirme artı" olarak düşünmek daha doğrudur. Dış birleştirme, iç birleştirmenin döndürdüğü tüm eşleşen kayıtları döndürmeye devam eder, ayrıca tablolardan birinden eşleşmeyen kayıtları döndürür. Daha spesifik olarak, CUSTOMER ve AGENT tabloları için bir dış birleştirme oluşturulursa, iki senaryo mümkündür:

- **Sol dış** birleştirme, AGENT tablosunda eşleşen bir değere sahip olmayanlar da dahil olmak üzere CUSTOMER tablosundaki tüm satırları verir. Böyle bir birleştirme örneği Şekil 3.14 'te gösterilmektedir.

Şekil 3.14 Sol Dış Birleştirme

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124
1542311	Smithson	37134	421		

- **Sağ dış** birleştirme, CUSTOMER tablosunda eşleşen değerlere sahip olmayanlar da dahil olmak üzere, AGENT tablosundaki tüm satırları verir. Böyle bir birleştirme örneği Şekil 3.15 'te gösterilmektedir.

Şekil 3.1 Sağ Dış Birleştirme

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124
				333	9041234445

Dış birleştirmeler özellikle ilgili tablolardaki hangi değerlerin referans bütünlüğü sorunlarına neden olduğunu belirlemeye çalıştığınızda kullanışlıdır. Bu tür sorunlar, yabancı anahtar değerleri ilgili tablo(lar)daki birincil anahtar değerleriyle eşleşmediğinde ortaya çıkar. Aslında, büyük elektronik tabloları veya diğer "veritabanı olmayan" verileri ilişkisel veritabanı tablolarına dönüştürmeniz istenirse, dönüştürmelerden sonra referans bütünlüğü hatalarıyla karşılaştığınızda dış birleştirmelerin size büyük miktarda zaman ve sayısız baş ağrısı kazandırdığını keşfedeceksiniz.

Dış birleştirmelerin neden "sol" ve "sağ" olarak etiketlendiğini merak edebilirsiniz. Etiketler, tabloların SQL komutunda listelendiği sırayı ifade eder. Bölüm 7'de bu tür birleştirmeler daha ayrıntılı olarak incelenmektedir.

Not

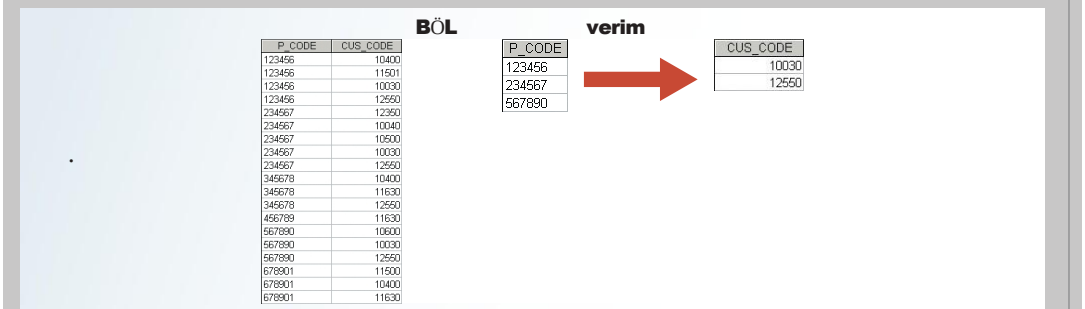
Outer join aynı zamanda JOIN'in bir uzantısıdır. Outer join'ler JOIN, DIFFERENCE, UNION ve PRODUCT'ın uygulamasıdır. Bir JOIN eşleşen ikilileri döndürür, DIFFERENCE bir tablodaki ortak öznitelikte diğer ilişkinin özniteliğinde görünmeyen değerlere sahip ikilileri bulur, bu eşleşmeyen ikililer bir PRODUCT aracılığıyla NULL değerlerle birleştirilir ve ardından bir UNION bu sonuçları tek bir ilişkide birleştirir. Açıkçası, tanımlanmış bir dış birleştirme büyük bir basitleştirmedir! Sol ve sağ dış birleşimler sırasıyla ⟕ ve ⟖ sembolleriyle gösterilir.

Bölmek

DIVIDE operatörü, bir veri kümesinin başka bir veri kümesindeki tüm veri değerleriyle ilişkilendirilmesi hakkındaki soruları yanıtlamak için kullanılır. DIVIDE işlemi, bir çift sütunlu tabloyu (Tablo 1) kar payı olarak ve bir tek sütunlu tabloyu (Tablo 2) bölen olarak kullanır. Örneğin, Şekil 3.16da soldaki Tablo 1'de bir müşteri listesi ve satın alınan ürünler gösterilmektedir. Ortadaki Tablo 2, kullanıcıların ilgisini çeken bir dizi ürün içermektedir. Varsa hangi müşterilerin Tablo 2'de gösterilen her ürünü satın aldığını belirlemek için bir DIVIDE işlemi kullanılabilir. Şekilde, bölen P_CODE ve CUS_CODE sütunlarını içerir. Bölen, P_CODE sütununu içerir. Tablolar ortak bir sütuna sahip olmalıdır - bu durumda P_CODE sütunu. Sağdaki DIVIDE işleminin çıktısı, bölenin ikinci sütunundaki (CUS_CODE) bölenin her satırıyla ilişkili tüm değerleri içeren tek bir sütundur.

BÖL
İlişkisel cebirde, bir veri kümesinin başka bir kümesindeki tüm veri değerleriyle ilişkilendirilmesi hakkındaki sorguları yanıtlayan bir operatör.

Şekil 3.16'da gösterilen örneği kullanarak aşağıdakilere dikkat edin:



Tablo 1, Tablo 2 tarafından "bölünerek" Tablo 3 oluşturulur. Tablo 1 ve 2'nin her ikisi de P_CODE sütununu içerir ancak CUS_CODE sütununu paylaşmaz