Tablo 7.6 Yabancı Anahtarlar Aracılığıyla Bağlantı Oluşturma

Tablo	Gösterilecek Özellikler	Bağlantı Niteliği
ÜRÜN	P_DESCRIPT, P_PRICE	V_CODE
SATICI	V_NAME, V_CONTACT, V_AREACODE, V_PHONE	V_CODE

Şekil 7.24'te gösterilen çıktıyı üreten PRODUCT ve VENDOR tablolarının birleştirilmesi birden fazla yolla gerçekleştirilebilir.

Şekil 7.24 Birleştirmenin Sonuçları

P_DESCRIPT	P_PRICE	V_NAME	V_CONTACT	V_AREACODE	V_PHONE
Claw hammer	9.95	Bryson, Inc.	Smithson	615	223-3234
1.25-in. metal screw, 25	6.99	Bryson, Inc.	Smithson	615	223-3234
2.5-in. wd. screw, 50	8.45	D&E Supply	Singh	615	228-3245
7.25-in. pwr. saw blade	14.99	Gomez Bros.	Ortega	615	889-2546
9.00-in. pwr. saw blade	17.49	Gomez Bros.	Ortega	615	889-2546
Rat-tail file, 1/8-in. fine	4.99	Gomez Bros.	Ortega	615	889-2546
Hrd. cloth, 1/4-in., 2x50	39.95	Randsets Ltd.	Anderson	901	678-3998
Hrd. cloth, 1/2-in., 3x50	43.99	Randsets Ltd.	Anderson	901	678-3998
B&D jigsaw, 12-in. blade	109.92	ORDVA, Inc.	Hakford	615	898-1234
B&D jigsaw, 8-in. blade	99.87	ORDVA, Inc.	Hakford	615	898-1234
Hicut chain saw, 16 in.	256.99	ORDVA, Inc.	Hakford	615	898-1234
Power painter, 15 psi., 3-nozzle	109.99	Rubicon Systems	Orton	904	456-0092
B&D cordless drill, 1/2-in.	38.95	Rubicon Systems	Orton	904	456-0092
Steel matting, 4'x8'x1/6", .5" mesh	119.95	Rubicon Systems	Orton	904	456-0092

7-7a Doğal Katılım

Bölüm 3'ten doğal birleştirmenin eşleşen sütunlarda eşleşen değerlere sahip tüm satırları döndürdüğünü ve yinelenen sütunları ortadan kaldırdığını hatırlayın. Bu sorgu tarzı, tablolar ortak adlara sahip bir veya daha fazla ortak özniteliği paylaştığında kullanılır. Doğal birleştirme sözdizimi şöyledir:

SELECT *column-list* FROM *table1* NATURAL JOIN *table2*

Doğal birleştirme aşağıdaki görevleri yerine getirir:

- Aynı adlara ve uyumlu veri türlerine sahip öznitelikleri arayarak ortak öznitelik(ler)i belirler.
- Yalnızca ortak öznitelik(ler)de ortak değerlere sahip satırları seçer.
- Ortak öznitelik yoksa, iki tablonun ilişkisel çarpımını döndürür.

Aşağıdaki örnek, CUSTOMER ve INVOICE tablolarının doğal birleştirmesini gerçekleştirir ve yalnızca seçilen öznitelikleri döndürür:

```
SEÇİNİZ      CUS_CODE, CUS_LNAME, INV_NUMBER, INV_DATE
FROM         MÜŞTERI DOĞAL KATILIM FATURASI;
```

Bu sorgunun sonuçları Şekil 7.25'te gösterilmektedir.

Doğal gerçekleştirirken iki tabloyla sınırlı değilsinizdir. Örneğin, INVOICE, LINE ve PRODUCT tabloları arasında doğal bir birleştirme gerçekleştirebilir ve aşağıdakileri yazarak yalnızca seçilen öznitelikleri yansıtabilirsiniz:

```
SEÇİNİZ      INV_NUMBER, P_CODE, P_DESCRIPT, LINE_UNITS, LINE_PRICE
FROM         FATURA DOĞAL BIRLEŞTIRME HATTI DOĞAL BIRLEŞTIRME
ÜRÜNÜ;
```

Bu SQL kodunun sonuçları Şekil 7.26'da gösterilmektedir.

Şekil 7.25 MÜŞTERİ DOĞAL ORTAK FATURA Sonuçlar

CUS_CODE	CUS_LNAME	INV_NUMBER	INV_DATE
10011	Dunne	1002	16-Jan-22
10011	Dunne	1004	17-Jan-22
10011	Dunne	1008	17-Jan-22
10012	Smith	1003	16-Jan-22
10014	Orlando	1001	16-Jan-22
10014	Orlando	1006	17-Jan-22
10015	O'Brian	1007	17-Jan-22
10018	Farriss	1005	17-Jan-22

Şekil 7.26 Üç Tablo ile DOĞAL BİRLEŞİM

INV_NUMBER	P_CODE	P_DESCRIPT	LINE_UNITS	LINE_PRICE
1001	13-Q2/P2	7.25-in. pwr. saw blade	1	14.99
1001	23109-HB	Claw hammer	1	9.95
1002	54778-2T	Rat-tail file, 1/8-in. fine	2	4.99
1003	2238/QPD	B&D cordless drill, 1/2-in.	1	38.95
1003	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	1	39.95
1003	13-Q2/P2	7.25-in. pwr. saw blade	5	14.99
1004	54778-2T	Rat-tail file, 1/8-in. fine	3	4.99
1004	23109-HB	Claw hammer	2	9.95
1005	PVC23DRT	PVC pipe, 3.5-in., 8-ft	12	5.87
1006	SM-18277	1.25-in. metal screw, 25	3	6.99
1006	2232/QTY	B&D jigsaw, 12-in. blade	1	109.92
1006	23109-HB	Claw hammer	1	9.95
1006	89-WRE-Q	Hicut chain saw, 16 in.	1	256.99
1007	13-Q2/P2	7.25-in. pwr. saw blade	2	14.99
1007	54778-2T	Rat-tail file, 1/8-in. fine	1	4.99
1008	PVC23DRT	PVC pipe, 3.5-in., 8-ft	5	5.87
1008	WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	3	119.95
1008	23109-HB	Claw hammer	1	9.95

Not

Bazı DBMS'ler NATURAL JOIN operatörünü içerse de, uygulamada genellikle tavsiye edilmez çünkü programcı ve kod üzerinde bakım yapan diğer kişiler için DBMS'nin tam olarak hangi özniteliği veya öznitelikleri birleştirmeyi gerçekleştirmek için ortak atribüt olarak kullandığı belirsiz olabilir. Kod ilk yazıldığında DBMS tabloları doğru bir şekilde birleştiriyor olsa bile, kullanılan veritabanı tablolarının yapısında sonradan yapılan değişiklikler DBMS'nin tabloları daha sonra yanlış bir şekilde birleştirmesine neden olabilir.

7-7b JOIN USING Sözdizimi

Birleştirmeyi ifade etmenin ikinci bir yolu USING anahtar sözcüğüdür. Sorgu yalnızca USING cümlesinde belirtilen sütunda eşleşen değerlere sahip satırları döndürür ve bu sütun her iki tabloda da . Sözdizimi şöyledir:

SELECT *column-list* FROM *table1* JOIN *table2* USING (*common-column*)

JOIN USING sorgusunu çalışırken görmek için, aşağıdakileri yazarak INVOICE ve LINE tabloları arasında bir birleştirme gerçekleştirin:

```
SEÇİNİZ     P_CODE, P_DESCRIPT, V_CODE, V_NAME, V_AREACODE,
            V_PHONE
FROM        ÜRÜN (V_CODE) KULLANARAK SATICIYA KATILIR;
```

SQL ifadesi Şekil 7.27'de gösterilen sonuçları üretir.

Şekil 7.27 Sonuçları KULLANARAK KATILIN

P_CODE	P_DESCRIPT	V_CODE	V_NAME	V_AREACODE	V_PHONE
23109-HB	Claw hammer	21225	Bryson, Inc.	615	223-3234
SM-18277	1.25-in. metal screw, 25	21225	Bryson, Inc.	615	223-3234
SW-23116	2.5-in. wd. screw, 50	21231	D&E Supply	615	228-3245
13-Q2/P2	7.25-in. pwr. saw blade	21344	Gomez Bros.	615	889-2546
14-Q1/L3	9.00-in. pwr. saw blade	21344	Gomez Bros.	615	889-2546
54778-2T	Rat-tail file, 1/8-in. fine	21344	Gomez Bros.	615	889-2546
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	23119	Randsets Ltd.	901	678-3998
1558-QW1	Hrd. cloth, 1/2-in., 3x50	23119	Randsets Ltd.	901	678-3998
2232/QTY	B&D jigsaw, 12-in. blade	24288	ORDVA, Inc.	615	898-1234
2232/QWE	B&D jigsaw, 8-in. blade	24288	ORDVA, Inc.	615	898-1234
89-WRE-Q	Hicut chain saw, 16 in.	24288	ORDVA, Inc.	615	898-1234
11QER/31	Power painter, 15 psi., 3-nozzle	25595	Rubicon Systems	904	456-0092
2238/QPD	B&D cordless drill, 1/2-in.	25595	Rubicon Systems	904	456-0092
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	25595	Rubicon Systems	904	456-0092

Yukarıdaki SQL komut dizisi, PRODUCT tablosundaki bir satırı VENDOR tablosundaki bir satırla birleştirir; USING cümlesinde belirtildiği gibi, bu satırların V_CODE değerleri aynıdır. Herhangi bir satıcı sipariş edilen herhangi bir sayıda ürünü teslim edebileceğinden, PRODUCT tablosu VENDOR tablosundaki her bir V_CODE girişi için birden fazla V_CODE girişi içerebilir. Başka bir deyişle, VENDOR'daki her V_CODE, PRODUCT'taki birçok V_CODE satırıyla eşleştirilebilir.

NATURAL JOIN komutunda olduğu gibi, JOIN USING işleci tablo niteleyicileri gerektirmez ve ortak özniteliğin yalnızca bir kopyasını döndürür.

Not

Oracle ve MySQL JOIN USING sözdizimini destekler. MS SQL Server ve Access desteklemez. Oracleda JOIN USING kullanılırsa, tablo niteleyicileri sorgunun herhangi bir yerinde ortak nitelik ile kullanılamaz. MySQL, USING cümlesinin kendisi dışında herhangi bir yerde ortak nitelik üzerinde tablo niteleyicilerine izin verir.

7-7c JOIN ON Sözdizimi

Önceki iki birleştirme stili, birleştirilen tablolarda ortak öznitelik adlarını kullanır. Tabloların ortak öznitelik adları olmadığında bir birleştirmeyi ifade etmenin başka bir yolu da JOIN ON işlemini kullanmaktır. Sorgu yalnızca belirtilen birleştirme koşulunu karşılayan satırları döndürür. Birleştirme koşulu tipik olarak iki sütunun eşitlik karşılaştırma ifadesini içerir. (Sütunlar aynı adı paylaşabilir veya paylaşmayabilir, ancak açıkça karşılaştırılabilir veri türlerine sahip olmalıdırlar). Sözdizimi şöyledir:

SELECT *column-list* FROM *table1* JOIN *table2* ON *join-condition*

Aşağıdaki örnek, ON cümlesini kullanarak INVOICE ve LINE tablolarının birleştirilmesini gerçekleştirir. Sonuç Şekil 7.28'de gösterilmektedir.

```
SEÇİNİZ     INVOICE.INV_NUMBER, PRODUCT.P_CODE, P_DESCRIPT,
            LINE_UNITS, LINE_PRICE
FROM        INVOICE JOIN LINE ON INVOICE.INV_NUMBER 5
            LINE.INV_NUMBER JOIN PRODUCT ON LINE.P_CODE 5
            PRODUCT.P_CODE;
```

Şekil 7.28 JOIN ON Sonuçlar

INV_NUMBER	P_CODE	P_DESCRIPT	LINE_UNITS	LINE_PRICE
1001	13-Q2/P2	7.25-in. pwr. saw blade	1	14.99
1001	23109-HB	Claw hammer	1	9.95
1002	54778-2T	Rat-tail file, 1/8-in. fine	2	4.99
1003	2238/QPD	B&D cordless drill, 1/2-in.	1	38.95
1003	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	1	39.95
1003	13-Q2/P2	7.25-in. pwr. saw blade	5	14.99
1004	54778-2T	Rat-tail file, 1/8-in. fine	3	4.99
1004	23109-HB	Claw hammer	2	9.95
1005	PVC23DRT	PVC pipe, 3.5-in., 8-ft	12	5.87
1006	SM-18277	1.25-in. metal screw, 25	3	6.99
1006	2232/QTY	B&D jigsaw, 12-in. blade	1	109.92
1006	23109-HB	Claw hammer	1	9.95
1006	89-WRE-Q	Hicut chain saw, 16 in.	1	256.99
1007	13-Q2/P2	7.25-in. pwr. saw blade	2	14.99
1007	54778-2T	Rat-tail file, 1/8-in. fine	1	4.99
1008	PVC23DRT	PVC pipe, 3.5-in., 8-ft	5	5.87
1008	WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	3	119.95
1008	23109-HB	Claw hammer	1	9.95

Not

SQL programlama için en iyi uygulamalar, bu bölümün ilerleyen kısımlarında ele alınan NATURAL JOIN veya eski tarz birleştirmeler yerine JOIN ON veya JOIN USING kullanılmasını önermektedir. JOIN USING sözdizimi DBMS satıcıları arasında yaygın olarak desteklenmez ve ortak niteliklerin birleştirilen tablolarda tam olarak aynı ada sahip olmasını gerektirir. Sonuç olarak, kullanım olanakları, yaygın olarak desteklenen ve ortak nitelikler üzerinde herhangi bir sınırlaması olmayan JOIN ON'a göre daha sınırlıdır. Bu nedenle, pratikte JOIN ON genellikle tercih edilen birleştirme sözdizimi olarak kabul edilir.

7-7d Ortak Öznitelik Adları

Bölüm 3'te sunulan ilişkisel tablo özelliklerinden biri, bir tablodaki hiçbir iki sütunun tam olarak aynı ada sahip olamayacağıdır. Tabloların birleştirilmesi, yeni ve tek bir tablo oluşturmak için belirtilen birleştirme ölçütlerini kullanarak tablolardaki satırları birleştirir. Bu tabloların birleştirilmesi sürecinde sadece satırlar birleştirilmez, aynı zamanda tabloların sütunları da yeni tabloda bir araya getirilir. Sonuç olarak, orijinal tabloların her biri benzersiz sütun adlarına sahip olsa bile, tablolar arasında yinelenen sütun adları olması muhtemeldir. Bu sütunların tümü birleştirme işlemi ile aynı tabloya yerleştirildiğinde, sonuçta ortaya çıkan tabloda yinelenen sütun adları olması mümkündür. Bir benzersiz sütun adlarının ilişkisel gerekliliğini uygulamak için, RDBMS tablo adlarını sütun adlarının önüne ekleyecektir. Bu tam nitelikli adlar genellikle sorgu sonuçlarında tablo adı niteleyicisini göstermez, ancak sorgu kodu tam nitelikli adları kullanmalıdır. Yinelenen sütun adlarının en yaygın nedeni bir yabancı anahtarın varlığıdır. Aslında, çoğu sorgu, birleştirme kriterleri için ortak nitelik olarak PK/FK kombinasyonlarını kullanarak tabloları birleştirir. NATURAL JOIN ve JOIN USING işleçleri, yinelenen sütun adları sorununu önlemek için ortak öznitelik için yinelenen sütunları otomatik olarak ortadan kaldırır. JOIN ON cümlesi ortak niteliğin bir kopyasını otomatik olarak kaldırmaz, bu nedenle sorgu ortak niteliğe her başvurduğunda bir tablo niteleyicisi gerektirir. Aşağıdaki koddaki farka dikkat edin:

```
SEÇİNİZ     P_CODE, VENDOR.V_CODE, V_NAME
FROM        PRODUCT JOIN VENDOR ON PRODUCT.V_CODE 5
            SATICI.V_CODE;
```

aşağıdaki kodla aynı sonucu üretir (bkz. Şekil 7.29): SEÇİNİZ

```
            P_CODE, V_CODE, V_NAME
FROM        ÜRÜN (V_CODE) KULLANARAK SATICIYA KATILIR;
```

P_CODE	V_CODE	V_NAME
23109-HB	21225	Bryson, Inc.
SM-18277	21225	Bryson, Inc.
SW-23116	21231	D&E Supply
13-Q2/P2	21344	Gomez Bros.
14-Q1/L3	21344	Gomez Bros.
54778-2T	21344	Gomez Bros.
1546-QQ2	23119	Randsets Ltd.
1558-QW1	23119	Randsets Ltd.
2232/QTY	24288	ORDVA, Inc.
2232/QWE	24288	ORDVA, Inc.
89-WRE-Q	24288	ORDVA, Inc.
11QER/31	25595	Rubicon Systems
2238/QPD	25595	Rubicon Systems
WR3/TT3	25595	Rubicon Systems

7-7e Eski Stil Birleştirmeler

Bölüm 3'te, doğal birleştirmenin kavramsal olarak üç adımlı bir süreç olarak düşünülebileceğini öğrendiniz: (1) tablolar arasında bir çarpım oluşturun, (2) yalnızca ortak öznitelik için eşleşen değerlere sahip satırlarla sınırlamak için ilişkisel seçim işlemini kullanın ve (3)

ortak özniteliğin bir kopyasını bırakmak için ilişkisel projeksiyon. Daha sonra bir equijoin'in bu üç adımdan sadece ilk ikisinin gerçekleştirilmesinin sonucu olduğu gösterilmiştir. En iyi uygulamalar bu gerçek adımları kullanarak birleştirme yapmayı cesaretlendirmese de, bunu yapmak hala mümkündür. Örneğin, aşağıdakileri yazarak PRODUCT ve VENDOR tablolarını ortak V_CODE'ları aracılığıyla birleştirebilirsiniz:

```
SEÇİNİZ      P_CODE, P_DESCRIPT, P_PRICE, V_NAME
FROM         ÜRÜN, SATICI
NEREDE       PRODUCT.V_CODE 5 VENDOR.V_CODE;
```

Önceki SQL birleştirme sözdizimi bazen "eski tarz" birleştirme olarak adlandırılır. FROM cümlesi birleştirilen tabloları içerir ve WHERE cümlesi tabloları birleştirmek için kullanılan koşul(lar)ı içerir.

Önceki sorguyla ilgili aşağıdaki noktalara dikkat edin:

- FROM cümlesi hangi tabloların birleştirileceğini belirtir. Üç veya daha fazla tablo dahil edilirse, birleştirme işlemi her seferinde iki tablo olacak şekilde soldan sağa doğru gerçekleştirilir. Örneğin, T1, T2 ve T3 tablolarını birleştiriyorsanız, ilk birleştirme T2 ile T1 tablosudur; bu birleştirmenin sonuçları daha sonra T3 tablosuna birleştirilir.
- WHERE cümlesindeki birleştirme koşulu, SELECT deyimine hangi satırların döndürüleceğini söyler. Bu durumda SELECT deyimi, PRODUCT ve VENDOR tablolarındaki V_CODE değerlerinin eşit olduğu tüm satırları döndürür.
- Birleştirme koşullarının sayısı her zaman birleştirilen tabloların sayısı eksi bire eşittir. Örneğin, üç tabloyu (T1, T2 ve T3) birleştirirseniz, iki birleştirme koşulunuz (j1 ve j2) *olur*. Tüm birleştirme koşulları bir AND mantıksal operatörü aracılığıyla bağlanır. İlk birleştirme koşulu (j1), T1 ve T2 için birleştirme kriterlerini tanımlar. İkinci birleştirme koşulu (j2), ilk birleştirme ve T3'ün çıktısı için birleştirme kriterlerini tanımlar.
- Genel olarak, birleştirme koşulu bir tablodaki birincil anahtar ile ikinci tablodaki ilgili yabancı anahtarın eşitlik karşılaştırması olacaktır.

Eski tarz birleştirmeler genellikle iki potansiyel sorun nedeniyle önerilmez. İlk olarak, tabloları birleştirme görevi hem FROM hem de WHERE cümlelerine bölünür ve bu da karmaşık sorguların bakımını daha zor hale getirir. SELECT sorgu cümleleri arasında sorumlulukların net bir şekilde ayrılması kod bakımını kolaylaştırır. JOIN ON veya JOIN USING sözdizimi ile, tabloları birleştirmek için gerekli tüm kod FROM bulunur. Verileri iş gereksinimlerine göre kısıtlamak için gerekli kodun tamamı WHERE cümlesinde yer alır. Eski tarz bir ile, birleştirmeyi tamamlama kriterleri, verileri iş gereksinimlerine göre kısıtlama kriterleri ile karıştırılır. İkinci olarak, eski tarz birleştirme, diğer birleştirmelerde olmayan tespit edilmemiş hatalara açıktır. Örneğin, aşağıdaki sorgu, TN'den satıcılar tarafından tedarik edilen ürünleri satın alan müşterileri listelemek için birden fazla tabloya katılmaya çalışır, ancak bir hata içerir. LINE tablosunu ve PRODUCT tablosunu bağlamak için birleştirme koşulu eksiktir. Sonuç olarak, sorgu bir hata oluşturur.

```
SEÇİNİZ      CUS_FNAME, CUS_LNAME, V_NAME
FROM         CUSTOMER JOIN INVOICE ON CUSTOMER.CUS_CODE 5
             INVOICE.CUS_CODE JOIN LINE ON INVOICE.INV_NUMBER 5
             LINE.INV_NUMBER JOIN PRODUCT JOIN VENDOR ON
             PRODUCT.V_CODE 5 .V_CODE
NEREDE       V_STATE 5 'TN';
```

Önceki sorguda, DBMS eksik bir birleştirme koşulunu tespit edebilir çünkü her JOIN bir birleştirme koşuluna sahip olmalıdır.

Eski tarz birleştirmelerin kullanıldığı aşağıdaki sorgu da aynı hatayı içerir. Ancak, DBMS'den bir hata üretmez - sadece kullanıcılara yanlış veri sağlar! DBMS, WHERE cümlesindeki kriterlerle amaçlanan birleştirmeleri ilişkilendiremez, bu nedenle eksik birleştirme koşulunu tespit edemez.

```
SEÇİNİZ     CUS_FNAME, CUS_LNAME, V_NAME
FROM        MÜŞTERI, FATURA, HAT, ÜRÜN, SATICI NEREDE
            V_STATE 5 'TN' VE CUSTOMER.CUS_CODE 5
            INVOICE.CUS_CODE VE INVOICE.INV_NUMBER 5
            LINE.INV_NUMBER VE PRODUCT.V_CODE 5 VENDOR.V_CODE;
```

7-7f Dış Birleştirmeler

Bir dış birleştirme yalnızca birleştirme koşuluyla eşleşen satırları (yani, ortak sütunlarda eşleşen değerlere sahip satırlar) değil, aynı zamanda eşleşmeyen değerlere sahip satırları da döndürür. ANSI standardı üç tür dış birleştirme tanımlar: sol, sağ ve tam. Sol ve sağ tanımlamaları, tabloların DBMS tarafından işlenme sırasını yansıtır. Birleştirme işlemlerinin her seferinde iki tabloda gerçekleştiğini unutmayın. FROM cümlesinde adı geçen ilk tablo sol taraf, adı geçen ikinci tablo ise sağ taraf olacaktır. Üç veya daha fazla tablo , ilk iki tablonun birleştirilmesinin sonucu sol taraf olur ve üçüncü tablo sağ taraf olur.

Sol dış birleştirme yalnızca birleştirme koşuluyla eşleşen satırları (yani, ortak sütunda eşleşen değerlere sahip satırlar) değil, aynı zamanda sağ tabloda eşleşmeyen değerlere sahip sol tablodaki satırları da döndürür. Sözdizimi şöyledir:

```
SEÇİNİZ     sütun-listesi
FROM        table1 LEFT [OUTER] JOIN table2 ON join-condition
```

Örneğin, aşağıdaki sorgu tüm ürünler için ürün kodunu, satıcı kodunu ve satıcı adını listeler ve eşleşen ürünü olmayan satıcıları içerir:

```
SEÇİNİZ     P_CODE, VENDOR.V_CODE, V_NAME
FROM        VENDOR LEFT JOIN PRODUCT ON VENDOR. V_CODE 5
            PRODUCT.V_CODE;
```

Önceki SQL kodu ve sonuçları Şekil 7.30'da gösterilmektedir.

Sağ dış birleştirme yalnızca birleştirme koşuluyla eşleşen satırları (, ortak sütunda eşleşen değerlere sahip satırlar) değil, aynı zamanda sol tabloda eşleşmeyen değerlere sahip sağ tablodaki satırları da döndürür. Sözdizimi şöyledir:

```
SEÇİNİZ     sütun-listesi
FROM        table1 RIGHT [OUTER] JOIN table2 ON join-condition
```

Örneğin, aşağıdaki sorgu tüm ürünler için ürün kodunu, satıcı kodunu ve satıcı adını listeler ve eşleşen bir satıcı koduna sahip olmayan ürünleri içerir:

```
SEÇİNİZ     P_CODE, VENDOR.V_CODE, V_NAME
FROM        SATICI SAĞ SATICIDAKI ÜRÜNE KATILIR. V_CODE 5
            PRODUCT.V_CODE;
```

SQL kodu ve çıktısı Şekil 7.31'de gösterilmektedir.

Şekil 7.30 LEFT JOIN Sonuçlar

P_CODE	V_CODE	V_NAME
23109-HB	21225	Bryson, Inc.
SM-18277	21225	Bryson, Inc.
	21226	SuperLoo, Inc.
SW-23116	21231	D&E Supply
13-Q2/P2	21344	Gomez Bros.
14-Q1/L3	21344	Gomez Bros.
54778-2T	21344	Gomez Bros.
	22567	Dome Supply
1546-QQ2	23119	Randsets Ltd.
1558-QW1	23119	Randsets Ltd.
	24004	Brackman Bros.
2232/QTY	24288	ORDVA, Inc.
2232/QWE	24288	ORDVA, Inc.
89-WRE-Q	24288	ORDVA, Inc.
	25443	B&K, Inc.
	25501	Damal Supplies
11QER/31	25595	Rubicon Systems
2238/QPD	25595	Rubicon Systems
WR3/TT3	25595	Rubicon Systems

Şekil 7.31 RIGHT JOIN Sonuçlar

P_CODE	V_CODE	V_NAME
23114-AA		
PVC23DRT		
23109-HB	21225	Bryson, Inc.
SM-18277	21225	Bryson, Inc.
SW-23116	21231	D&E Supply
13-Q2/P2	21344	Gomez Bros.
14-Q1/L3	21344	Gomez Bros.
54778-2T	21344	Gomez Bros.
1546-QQ2	23119	Randsets Ltd.
1558-QW1	23119	Randsets Ltd.
2232/QTY	24288	ORDVA, Inc.
2232/QWE	24288	ORDVA, Inc.
89-WRE-Q	24288	ORDVA, Inc.
11QER/31	25595	Rubicon Systems
2238/QPD	25595	Rubicon Systems
WR3/TT3	25595	Rubicon Systems

Not

Bazı DBMS satıcıları hibrit birleştirmeleri destekler. Bir hibrit birleştirme, iki veya daha fazla birleştirme türünün özelliklerini içerir. Örneğin, JOIN USING doğal birleştirme ve equijoin'in bir melezi olarak düşünülebilir. Doğal birleştirme gibi, ortak özniteliğin bir kopyasını bırakır. Bir equijoin gibi, birleştirme koşulu için kullanılan ortak özniteliği belirtmenize olanak tanır. Oracle ve MySQL gibi satıcılar karma doğal birleştirmeleri ve dış destekler. Bu, FROM PRODUCT NATURAL LEFT JOIN VENDOR gibi birleşimlere yol açar. Doğal gibi, ortak özniteliği belirtemezsiniz ve ortak özniteliğin bir kopyası sonuçlardan çıkarılır. Dış gibi, belirtilen tablodaki (sol veya sağ) eşleşmeyen satırlar dahil edilir. Normal doğal birleştirmelerde olduğu gibi, bu tür birleştirmeler de aynı nedenden dolayı pratikte önerilmez - DBMS tabloların nasıl birleştirilmesi gerektiği konusunda tahminde bulunmamalıdır.

Outer joins birçok durumda kullanışlıdır. Örneğin, sonuçları tablolar arasında yalnızca eşleşmeyen satırlarla sınırlandırmaya yardımcı olmak için kullanılabilirler. Daha önce Şekil 7.23'te gösterildiği gibi, tablodaki yabancı anahtar null içerdiğinden, 1:M ilişkisinin "çok" tarafındaki eşleşmeyen satırları bulmak basitleştirilmiştir. Ancak, 1:M ilişkisinin "bir" tarafında eşleşmeyen satırları bulmak daha karmaşıktır ve Şekil 7.32'de gösterildiği gibi herhangi bir ürünle ilişkili olmayan satıcıları bulmak gibi bir dış birleştirme ile IS NULL işleci kullanılarak yapılabilir.

```
SEÇİNİZ      V_CODE, V_NAME, P_CODE
FROM         PRODUCT RIGHT JOIN VENDOR ON PRODUCT.V_CODE 5
               SATICI.V_CODE
NEREDE       P_CODE NULL'DUR;
```

Şekil 7.32 Herhangi Bir Ürünle İlişkisi Olmayan Satıcılar

V_CODE	V_NAME	P_CODE
21226	SuperLoo, Inc.	
22567	Dome Supply	
24004	Brackman Bros.	
25443	B&K, Inc.	
25501	Damal Supplies	

Tam dış birleştirme yalnızca birleştirme koşuluyla eşleşen satırları (yani, ortak sütunda eşleşen değerlere sahip satırlar) değil, aynı zamanda her iki taraftaki tabloda eşleşmeyen değerlere sahip tüm satırları da döndürür. Sözdizimi şöyledir:

```
SEÇİNİZ      sütun-listesi
FROM         table1 FULL [OUTER] JOIN table2 ON join-condition
```

Örneğin, aşağıdaki sorgu tüm ürünler için ürün kodunu, satıcı kodunu ve satıcı adını listeler ve tüm ürün satırlarının (eşleşen satıcıları olmayan ürünler) yanı tüm satıcı satırlarını (eşleşen ürünleri olmayan satıcılar) içerir:

```
SEÇİNİZ      P_CODE, VENDOR.V_CODE, V_NAME
FROM         SATICI SATICIDAKI ÜRÜNE TAM OLARAK KATILIR. V_CODE 5
             PRODUCT.V_CODE;
```

SQL kodu ve sonuçları Şekil 7.33'te gösterilmektedir.

Şekil 7.33 FULL OUTER JOIN Sonuçları

P_CODE	V_CODE	V_NAME
	21226	SuperLoo, Inc.
	22567	Dome Supply
	24004	Brackman Bros.
	25443	B&K, Inc.
	25501	Damal Supplies
11QER/31	25595	Rubicon Systems
13-Q2/P2	21344	Gomez Bros.
14-Q1/L3	21344	Gomez Bros.
1546-QQ2	23119	Randsets Ltd.
1558-QW1	23119	Randsets Ltd.
2232/QTY	24288	ORDVA, Inc.
2232/QWE	24288	ORDVA, Inc.
2238/QPD	25595	Rubicon Systems
23109-HB	21225	Bryson, Inc.
23114-AA		
54778-2T	21344	Gomez Bros.
89-WRE-Q	24288	ORDVA, Inc.
PVC23DRT		
SM-18277	21225	Bryson, Inc.
SW-23116	21231	D&E Supply
WR3/TT3	25595	Rubicon Systems

Not

Oracle ve MS SQL Server FULL JOIN sözdizimini destekler. MySQL ve Access desteklemez.

7-7 g Çapraz Birleştirme

çapraz birleştirme
İki tablonun ilişkisel çarpımını (veya Kartezyen çarpımını) gerçekleştiren bir birleştirme.

Çapraz birleştirme, iki tablonun ilişkisel *çarpımını* (*Kartezyen çarpım* olarak da bilinir) gerçekleştirir. Çapraz birleştirme sözdizimi şöyledir:

SELECT *column-list* FROM *table1* CROSS JOIN *table2*

Örneğin, aşağıdaki komut:

SELECT * FROM INVOICE CROSS JOIN LINE;

INVOICE ve LINE tablolarının 144 satır üreten bir çapraz birleştirmesini gerçekleştirir. (8 fatura satırı ve 18 satır satırı vardır, böylece 8 3 18 5 144 satır elde edilir).

Yalnızca belirtilen öznitelikleri veren bir çapraz birleştirme de gerçekleştirebilirsiniz. Örneğin, şunu belirtebilirsiniz:

```
SEÇİNİZ     INVOICE.INV_NUMBER, CUS_CODE, INV_DATE, P_CODE
FROM        FATURA ÇAPRAZ BIRLEŞTIRME HATTI;
```

Bu SQL deyimi aracılığıyla oluşturulan sonuçlar aşağıdaki sözdizimi kullanılarak da oluşturulabilir:

```
SEÇİNİZ     INVOICE.INV_NUMBER, CUS_CODE, INV_DATE, P_CODE
FROM        FATURA, SATIR;
```

Not

Oracle, MS SQL Server ve MySQL'in aksine, Access CROSS JOIN operatörünü desteklemez. Ancak, tüm DBMS'ler FROM cümlesindeki tablolar arasına virgül koyarak çapraz üretmeyi destekler, bu da çapraz birleştirme üretmek için daha yaygın bir yöntemdir.

Not

Adına rağmen, CROSS JOIN gerçek bir birleştirme işlemi değildir çünkü tabloların satırlarını ortak bir özniteliğe dayalı olarak birleştirmez.

7-7 h Tabloları Takma Ad ile Birleştirme

Verilerin alındığı kaynak tabloyu tanımlamak için bir takma ad kullanılabilir. P ve V takma adları, bir sonraki komut dizisinde PRODUCT ve VENDOR tablolarını etiketlemek için kullanılır. Herhangi bir yasal tablo adı takma ad olarak kullanılabilir. (Ayrıca, öznitelik listesi SELECT deyiminde yinelenen adlar içermediği için tablo adı önekleri olmadığına dikkat edin).

```
SEÇİNİZ     P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT, V_AREACODE,
            V_PHONE
FROM        PRODUCT P JOIN VENDOR V ON P.V_CODE 5 V.V_CODE;
```

Not

MS Access, bir tablo diğer adından önce AS anahtar sözcüğünü gerektirir. Oracle ve MySQL tablo diğer adları için AS anahtar sözcüğünü kullanmazken, MS SQL Server AS anahtar sözcüğü olsun ya da olmasın tablo diğer adlarını kabul eder. AS anahtar sözcüğünün kullanılması yukarıdaki sorguyu şu şekilde değiştirecektir:

```
SEÇİNİ      P_DESCRIPT, P_PRICE, V_NAME, V_CONTACT, V_AREACODE, V_PHONE
Z           PRODUCT AS P JOIN VENDOR AS V ON P.V_CODE 5 V.V_CODE;
FROM
```

Bir tablo takma adı belirtme yeteneği çok kullanışlıdır. Gördüğünüz gibi, bir takma ad bir sorgu içinde bir tablo adını kısaltmak için kullanılabilir, ancak bu bir tablo takma adı kullanmanın en yaygın nedeni değildir. Çoğu sınıfta sunulan veri modelleri, en fazla bir düzine ya da daha fazla tablo ile oldukça küçük olma eğilimindedir. Uygulamada, veri modelleri genellikle çok daha büyüktür. Yazarlar, her biri 30.000'den fazla tablo içeren veri modellerine sahip şirketlerle çalışmışlardır! Tahmin edebileceğiniz gibi, bir iş konusuyla ilgili bu kadar çok tablo olduğunda, yaratıcı bir veritabanı tasarımcısı ekibinin bile anlamlı, açıklayıcı varlık isimleri bulması giderek zorlaşır. Sonuç olarak, modelin birçok bölümünde şifreli, kısaltmalarla dolu varlık adları hakim olur. Bir tablo takma adı kullanmak, veritabanı programcısının, tablonun sorgu içinde hangi verileri sağladığını açıklayan bir tablo takma adı kullanarak kodun sürdürülebilirliğini geliştirmesine olanak tanır. Örneğin, hastayla ilgili 20 veri tablosu ve çeşitli poliçe, sigorta ve çalışan muafiyetleriyle ilgili birden fazla tabloya sahip bir sağlık sektörü veri modelinde, hastaya bağlı sigorta kapsamı poliçe muafiyetlerini içeren PDEPINPCEX adlı bir tabloya sorguda EXEMPTS gibi bir takma ad verilebilir. Bu, kolayca anlaşılamayan bir tablo adını anlaşılabilir bir takma adla değiştirerek sorgunun okunabilirliğini büyük ölçüde artırır.

7-7 i Özyinelemeli Birleştirmeler

Tablo takma adı özellikle, tekli ilişkilerle çalışırken olduğu gibi, bir tablonun özyinelemeli bir **sorguda** kendisiyle birleştirilmesi gerektiğinde kullanışlıdır. Örneğin, Şekil 7.34'te gösterilen EMP tablosuyla çalıştığınızı varsayalım.

özyinelemeli sorgu
Bir tabloyu kendisiyle birleştiren bir sorgu.

Şekil 7.34 ÇYP Tablosunun İçeriği

EMP_NUM	EMP_TITLE	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_DOB	EMP_HIRE_DATE	EMP_AREACODE	EMP_PHONE	EMP_MGR
100	Mr.	Kolmycz	George	D	15-Jun-67	15-Mar-10	615	324-5456	
101	Ms.	Lewis	Rhonda	G	19-Mar-90	25-Apr-11	615	324-4472	100
102	Mr.	Vandam	Rhett		14-Nov-83	20-Dec-15	901	675-8993	100
103	Ms.	Jones	Anne	M	16-Oct-99	28-Aug-19	615	898-3456	100
104	Mr.	Lange	John	P	08-Nov-96	20-Oct-19	901	504-4430	105
105	Mr.	Williams	Robert	D	14-Mar-00	08-Nov-20	615	890-3220	
106	Mrs.	Smith	Jeanine	K	12-Feb-93	05-Jan-14	615	324-7883	105
107	Mr.	Diante	Jorge	D	21-Aug-99	02-Jul-19	615	890-4567	105
108	Mr.	Wiesenbach	Paul	R	14-Feb-91	18-Nov-17	615	897-4358	
109	Mr.	Smith	George	K	18-Jun-86	14-Apr-14	901	504-3339	108
110	Mrs.	Genkazi	Leighla	W	19-May-95	01-Dec-15	901	569-0093	108
111	Mr.	Washington	Rupert	E	03-Jan-91	21-Jun-18	615	890-4925	105
112	Mr.	Johnson	Edward	E	14-May-86	01-Dec-08	615	898-4387	100
113	Ms.	Smythe	Melanie	P	15-Sep-95	11-May-20	615	324-9006	105
114	Ms.	Brandon	Marie	G	02-Nov-81	15-Nov-04	901	882-0845	108
115	Mrs.	Saranda	Hermine	R	25-Jul-97	23-Apr-18	615	324-5505	105
116	Mr.	Smith	George	A	08-Nov-90	10-Dec-13	615	890-2984	108

EMP tablosundaki verileri kullanarak, yöneticileriyle birlikte tüm çalışanların bir listesini oluşturabilirsiniz.

EMP tablosunu kendisiyle birleştirerek adlar. Bu durumda, tabloyu kendisinden ayırmak için takma adlar da kullanırsınız. SQL komut dizisi şu şekilde olacaktır:

```
SEÇİNİZ     E.EMP_NUM, E.EMP_LNAME, E.EMP_MGR, M.EMP_LNAME
FROM        EMP E JOIN EMP M ON E.EMP_MGR 5 M.EMP_NUM;
```

Önceki komut dizisinin çıktısı Şekil 7.35'te gösterilmektedir.

Şekil 7.35 Bir Tabloyu Kendisine Eklemek için Diğer Ad Kullanma

EMP_NUM	E.EMP_LNAME	EMP_MGR	M.EMP_LNAME
112	Johnson	100	Kolmycz
103	Jones	100	Kolmycz
102	Vandam	100	Kolmycz
101	Lewis	100	Kolmycz
115	Saranda	105	Williams
113	Smythe	105	Williams
111	Washington	105	Williams
107	Diante	105	Williams
106	Smith	105	Williams
104	Lange	105	Williams
116	Smith	108	Wiesenbach
114	Brandon	108	Wiesenbach
110	Genkazi	108	Wiesenbach
109	Smith	108	Wiesenbach

7-8 Agrega İşleme

Aşağıdaki sorguyu düşünün:

```
SEÇİNİZ      V_CODE, V_NAME, V_STATE, P_CODE, P_DESCRIPT, P_PRICE *
             P_QOH TOPLAM OLARAK
FROM         PRODUCT P JOIN VENDOR V ON P.V_CODE 5 V.V_CODE
WHERE        V_STATE IN ('TN', 'KY')
ORDER BY     V_STATE, TOPLAM DESC;
```

Bu sorgunun işlenişini göz önünde bulundurursanız, neredeyse RDBMS'nin her seferinde bir satır işliyormuş gibi göründüğünü görürsünüz. PRODUCT tablosundaki her satır, eşleşen satırları bulmak için VENDOR tablosundaki her satırla karşılaştırılır. Daha sonra bu eşleşen satırlar, satıcı durumunun listedeki bir değerle eşleşip eşleşmediğini görmek için her satıra bakılarak filtrelenir. Filtrelenen her satır için belirtilen sütunlar alınır ve hesaplanan alan hesaplanır. Döndürülen sütunlar kullanılarak, satırlar duruma göre değerlendirilir ve Şekil 7.36'da gösterilen nihai çıktıyı üretmek üzere satırları sıralamak için toplam alınır. RDBMS, tek tek satır işleme değil, veri kümeleri üzerinde çalışır, ancak bu noktaya kadar tartışılan işlemeyi satır tabanlı gibi hayal edebilirsiniz.

Şekil 7.36 TN veya KY'deki Satıcıdan Alınan Ürünlerin Toplam Değeri

V_CODE	V_NAME	V_STATE	P_CODE	P_DESCRIPT	TOTAL
21344	Gomez Bros.	KY	13-Q2/P2	7.25-in. pwr. saw blade	479.68
21344	Gomez Bros.	KY	14-Q1/L3	9.00-in. pwr. saw blade	314.82
21344	Gomez Bros.	KY	54778-2T	Rat-tail file, 1/8-in. fine	214.57
24288	ORDVA, Inc.	TN	89-WRE-Q	Hicut chain saw, 16 in.	2826.89
21231	D&E Supply	TN	SW-23116	2.5-in. wd. screw, 50	2002.65
21225	Bryson, Inc.	TN	SM-18277	1.25-in. metal screw, 25	1202.28
24288	ORDVA, Inc.	TN	2232/QTY	B&D jigsaw, 12-in. blade	879.36
24288	ORDVA, Inc.	TN	2232/QWE	B&D jigsaw, 8-in. blade	599.22
21225	Bryson, Inc.	TN	23109-HB	Claw hammer	228.85

Ancak, veritabanında satır koleksiyonları ile tek bir birimmiş gibi çalışmayı gerektiren birçok soru sorulmaktadır. Bu tür koleksiyon işlemleri toplama fonksiyonları ile yapılır. Bir toplama fonksiyonu kullanmanın tanımlayıcı özelliği, bir satır koleksiyonunu alıp tek bir indirgemesidir. SQL sayan, minimum ve maksimum değerleri bulan, ortalamaları hesaplayan ve benzeri kullanışlı toplama fonksiyonları sağlar. Daha da iyisi, SQL kullanıcının sorguları yalnızca kopyası olmayan veya kopyaları gruplanabilen girdilerle sınırlamasına olanak tanır.

7-8 a Toplu İşlevler

SQL sizin için belirli bir koşulu içeren satırların sayısını saymak, belirli bir nitelik için minimum veya maksimum değerleri bulmak, belirli bir sütundaki değerleri toplamak ve belirli bir sütundaki değerlerin ortalamasını almak gibi çeşitli matematiksel özetler gerçekleştirebilir. Bazı DBMS ürünleri diğer toplama işlevlerini desteklese de, Tablo 7.7'de gösterilenler en yaygın toplama işlevleridir ve çoğu DBMS ürünü tarafından desteklenir. Toplama işlevleri genellikle SELECT sütun listesinde, bir satır koleksiyonu genelinde hesaplanan bir toplam değeri döndürmek için kullanılır.

Tablo 7.7 Bazı Temel SQL Toplama İşlevleri

Fonksiyon	Çıktı
HESAP	Null olmayan değerler içeren satır sayısı
MIN	Belirli bir sütunda karşılaşılan minimum öznitelik değeri
MAX	Belirli bir sütunda karşılaşılan maksimum öznitelik değeri
SUM	Belirli bir sütun için tüm değerlerin toplamı
AVG	Belirli bir sütun için aritmetik ortalama (ortalama)

HESAP

HESAP
Not null içeren satırların sayısını çıktı olarak veren bir SQL toplama işlevi
Belirli bir sütun veya ifade için değerler, bazen DISTINCT cümlesiyle birlikte kullanılır.

COUNT fonksiyonu, bir niteliğin boş olmayan değerlerinin sayısını saymak için kullanılır. Aşağıdaki kodda, ürün sayısının bir çetelesi hesaplanır ve 16 sonucu döndürülür (bkz. Şekil 7.37).

```
SEÇİNİZ     COUNT(P_CODE)
FROM        ÜRÜN;
```

ÜRÜN Tablosundaki Ürün Kodlarının Sayısı

CountOfP_CODE
16

Toplama işlevinin PRODUCT tablosundaki tüm satır koleksiyonunu aldığına ve sonuç için tek bir satıra indirgediğine dikkat edin. Bu, satır koleksiyonlarını tek bir satıra indirgeyen toplamaların tanımlayıcı davranışlarından biridir. Satır koleksiyonunun tablodaki tüm satırlardan oluşması gerekmez. Örneğin, fiyatı 10 doların altında olan kaç ürün olduğunu belirlemek için aşağıdaki sorguyu kullanabilirsiniz.

```
SEÇİNİZ     COUNT(P_PRICE)
FROM        ÜRÜN
NEREDE      P_PRICE , 10;
```

Toplama fonksiyonları, parantez içinde parametre olarak bir değer, tipik olarak bir nitelik alır. Şekil 7.37'de kod, PRODUCT tablosundaki P_CODE birincil anahtarındaki değerleri saymıştır. Şekil 7.38'de gösterildiği gibi, aynı tablodaki V_CODE özniteliğini sayarken sonuçtaki farka dikkat edin.

```
SEÇİNİZ     COUNT(V_CODE)
FROM        ÜRÜN;
```

ÜRÜN Tablosundaki Satıcı Kodlarının Sayısı

CountOfV_CODE
14

Bu durumda, V_CODE özniteliği boş karakterler içerir. COUNT çeteleye null'ları dahil etmez. Bir sorgunun amacı, herhangi bir sütunun null içerip içermediğine bakılmaksızın bir koleksiyondaki satır sayısını döndürmekse, COUNT(*) sözdizimi kullanılabilir. Çoğu toplama

fonksiyonları parametre olarak tek bir nitelik alır; ancak COUNT, herhangi bir sütundaki değerlere bakılmaksızın satır sayısının döndürülmesi gerektiğini belirtmek için yıldız (*) joker karakterine izin verir.

COUNT, DISTINCT cümlesiyle birlikte de kullanılabilir. DISTINCT anahtar sözcüğünün kullanımına ilişkin önceki örnekte, Şekil 7.7'de gösterildiği gibi, sorgu sonucundaki yinelenen satırları ortadan kaldırmak için SELECT anahtar sözcüğünün hemen ardından DISTINCT kullanılmıştır. Şekil 7.7'de, farklı satıcı kodlarının bir listesi döndürülmüştür. Ancak, PRODUCT tablosunda *kaç* farklı satıcı olduğunu bulmak istediğinizi varsayalım. DISTINCT'i SELECT anahtar sözcüğünden hemen sonra yerleştirmek yerine, DISTINCT COUNT işlevinin içine yerleştirilebilir (bkz. Şekil 7.39).

```
SEÇİNİZ     COUNT(DISTINCT V_CODE) AS "COUNT DISTINCT"
FROM        ÜRÜN;
```

Şekil 7.39 Farklı Satıcı Kodlarının Sayısı

Count Distinct
6

Bu durumda, çetele COUNT tarafından hesaplanmadan önce öznitelikteki değerlere DISTINCT uygulanacaktır. Boş değerlerin V_CODE değerleri olarak sayılmadığına dikkat edin. FROM cümlesi PRODUCT tablosundan tüm değerleri alır, DISTINCT V_CODE içindeki yinelenen değerleri kaldırır ve ardından COUNT, DISTINCT tarafından döndürülen boş olmayan değerleri sayar.

Not

MS Access, DISTINCT tümcesi ile COUNT kullanımını desteklemez. MS Access'te bu tür sorguları kullanmak istiyorsanız, DISTINCT ve NOT NULL cümleleriyle alt sorgular (bu bölümde daha sonra ele alınacaktır) oluşturmanız gerekir. Örneğin, yukarıdaki iki sorgu için eşdeğer MS Access sorguları şunlardır:

```
SEÇİNİZ     COUNT(*)
FROM        (SELECT DISTINCT V_CODE FROM PRODUCT WHERE V_CODE IS NOT NULL);

ve

SELECT      COUNT(*)
FROM        (SELECT DISTINCT V_CODE
            FROM (SELECT V_CODE, P_PRICE FROM PRODUCT
                  WHERE V_CODE IS NOT NULL AND P_PRICE , 10));
```

Alt sorgular bu bölümün ilerleyen kısımlarında ayrıntılı olarak ele alınmaktadır.

MİN ve MAKS

MIN ve **MAX** fonksiyonları, PRODUCT tablosundaki en yüksek ve en düşük (maksimum ve minimum) fiyatlar gibi sorunlara yanıt bulmanıza yardımcı olur. COUNT fonksiyonunun örnekleri, toplama fonksiyonlarının bir satır koleksiyonunu tek bir indirgediğini göstermiştir. Ancak tek bir sütunun alınması gerekli değildir. Önceki örnekler tek bir sütun döndürüyordu çünkü sorgunun SELECT sütun listesinde yalnızca bir sütun belirtilmişti. Aşağıdaki kod, PRODUCT tablosundaki en yüksek ve en düşük fiyatları tek bir sorguda alır (bkz. Şekil 7.40).

MIN
Belirli bir sütundaki minimum öznitelik değerini veren bir SQL toplama işlevi.

MAX
Belirli bir sütundaki maksimum öznitelik değerini veren bir SQL toplama işlevi.

Şekil 7.40 Maksimum ve Minimum Fiyat Çıktısı

MAXPRICE	MINPRICE
256.99	4.99

```
SEÇİNİZ     MAX(P_PRICE) AS MAXPRICE, MIN(P_PRICE) AS MINPRICE
FROM        ÜRÜN;
```

MAX ve MIN toplama fonksiyonları tarih sütunlarıyla da kullanılabilir. Tarih aritmetiğinin daha önceki açıklamasından, tarihlerin veritabanında gün sayısı olarak saklandığını hatırlayın; yani, geçmişte tanımlanmış bir noktadan bu yana geçen gün sayısı. Bir gün sayısı olarak dün bugünden bir eksik, yarın ise bugünden bir fazladır. Bu nedenle, eski tarihler gelecek tarihlerden "daha küçüktür", bu nedenle en eski tarih en küçük tarih ve en gelecek tarih en büyük tarih olacaktır. Örneğin, hangi ürünün en eski envanter tarihine sahip olduğunu bulmak için MIN(P_INDATE) kullanırsınız. Aynı şekilde, bir ürünün en son envanter tarihini bulmak için MAX(P_INDATE) kullanırsınız.

SUM ve AVG

SUM
Belirli bir sütun veya ifade için tüm değerlerin toplamını veren bir SQL toplama işlevi

TOPLA işlevi, koşul(lar)ı kullanarak, belirtilen herhangi bir sayısal nitelik için toplam tutarı hesaplar. Örneğin, müşterilerinizin borçlu olduğu toplam tutarı hesaplamak istiyorsanız, aşağıdaki komutu kullanabilirsiniz:

```
SEÇİNİZ     SUM(CUS_BALANCE) AS TOTBALANCE
FROM        MÜŞTERİ;
```

Bir toplama işlevi parametre olarak bir değer alır, örneğin önceki sorgudaki CUS_BALANCE gibi. Değer genellikle bir tabloda depolanan bir özniteliktir. Ancak, türetilmiş öznitelikler ve formüller de kabul edilebilir. Örneğin, envanterde taşınan tüm kalemlerin toplam değerini bulmak istiyorsanız, aşağıdakini kullanabilirsiniz:

```
SEÇİNİZ     SUM(P_QOH *  P_PRICE) AS TOTVALUE
FROM        ÜRÜN;
```

Toplam değer, eldeki miktar ile tüm kalemlerin fiyatının çarpımının toplamıdır (bkz. Şekil 7.41).

TOTVALUE
15084.52

AVG
Belirtilen bir sütun veya ifade için ortalama ortalamanın çıktısını veren bir SQL toplama işlevi.

AVG işlevi biçimi SUM işlevine benzer ve aynı işletim kısıtlamalarına tabidir. Aşağıdaki komut seti, Şekil 7.42'de gösterildiği gibi, 56,42125'lik hesaplanmış ortalama fiyatı elde etmek için basit bir ortalama P_PRICE değerinin nasıl oluşturulabileceğini göstermektedir.

```
SEÇİNİZ     AVG(P_PRICE) AS AVGPRICE
FROM        ÜRÜN;
```

Şekil 7.42 Ortalama Ürün Fiyatı

AVGPRICE
56.42125

7-8 b Veri Gruplama

Önceki örneklerde, toplama işlevleri verilen tablolardaki tüm satırlardaki verileri özetlemiştir. Ancak bazen, özetleme için tüm tabloyu tek bir veri koleksiyonu olarak ele almak istemezsiniz. Satırlar, SELECT deyimindeki **GROUP BY** cümlesi kullanılarak hızlı ve kolay bir şekilde daha küçük koleksiyonlar halinde gruplandırılabilir. Daha sonra toplama fonksiyonları her bir küçük koleksiyon içindeki verileri toplar. Sözdizimi şöyledir:

GROUP BY
Bir SELECT deyimindeki toplama işlevlerinden herhangi biriyle birleştirildiğinde frekans dağılımları oluşturmak için kullanılan bir SQL tümcesi.

```
SEÇİNİZ      sütun listesi
FROM         tablelist
[NEREDE      koşul listesi]
[GROUP BY    sütun listesi]
[ORDER BY    sütun listesi [ASC | DESC]];
```

Şekil 7.42, veritabanındaki tüm ürünlerin ortalama fiyatını belirlemiştir. Ancak, kullanıcılar tüm ürünlerin fiyatını görmek yerine her bir satıcı tarafından sağlanan ürünlerin ortalama fiyatını görmek isterse ne olur? Aşağıdaki sorgu, Şekil 7.43'te gösterildiği gibi bu soruyu yanıtlayacaktır.

```
SEÇİNİZ      V_CODE, AVG(P_PRICE) AS AVGPRICE
FROM         ÜRÜN
GROUP BY     V_CODE;
```

V_CODE	AVGPRICE
	10.13
21225	8.47
21231	8.45
21344	12.49
23119	41.97
24288	155.59
25595	89.63

Sorgu, tüm PRODUCT tek bir koleksiyon olarak ele almak yerine, satırları her biri V_CODE değerine dayalı birkaç küçük koleksiyona ayırır. Böylece, 21225 satıcısından gelen tüm ürünler bir koleksiyona, 21344 satıcısından gelen tüm ürünler ikinci bir koleksiyona, 25595 satıcısından gelen tüm ürünler üçüncü bir koleksiyona yerleştirilir ve tüm ürünler bir koleksiyonda görünene kadar bu şekilde devam eder. Bu koleksiyonlar GROUP BY cümlesi kullanılarak oluşturulur. GROUP BY, koleksiyonları V_CODE değerine göre oluşturur ve ardından aggregate işlevi her koleksiyonu tek bir satıra indirger ve bu koleksiyon için ortalama fiyatı hesaplar. Toplama işlevi hala şu işi yapar

toplamalar her zaman yapar - bir satır koleksiyonunu tek bir satıra indirger - ancak bu durumda birden fazla koleksiyon vardır. Şekil 7.43'te satıcı kodları null olan ürünlerin birlikte gruplandığına dikkat edin. Toplama işlevleri hesaplama yaparken null'ları yok sayar, ancak GROUP BY cümlesi null'ları içerir ve koleksiyonları oluştururken tüm null'ları aynı kabul eder.

GROUP BY cümlesi ve toplama işlevleri arasındaki etkileşimi anlamak, bunları doğru kullanmak açısından çok önemlidir. Sonucu Şekil 7.44'te gösterilen aşağıdaki sorguyu düşünün:

```
SEÇİNİZ      V_CODE, V_NAME, COUNT(P_CODE) AS NUMPRODS,
             AVG(P_PRICE) AS AVGPRICE
FROM         PRODUCT JOIN VENDOR ON PRODUCT.V_CODE 5
             VENDOR.V_CODE
GROUP BY , V_NAME ORDER BY
V_NAME;
```

V_CODE	V_NAME	NUMPRODS	AVGPRICE
21225	Bryson, Inc.	2	8.47
21231	D&E Supply	1	8.45
21344	Gomez Bros.	3	12.49
24288	ORDVA, Inc.	3	155.59
23119	Randsets Ltd.	2	41.97
25595	Rubicon Systems	3	89.63

Bu sorguda, DBMS önce PRODUCT ve VENDOR tablolarından verileri alır ve ortak öznitelik olarak V_CODE kullanarak bunları birleştirir. Ardından, elde edilen satırlar hem V_CODE hem de V_NAME için aynı değerlere sahip olan satırlara göre koleksiyonlar halinde gruplandırılır. Üçüncü olarak, SELECT sütun listesi yalnızca V_CODE, V_NAME, P_CODE ve P_PRICE özniteliklerini yansıtır. Toplama işlevleri daha sonra her bir koleksiyonu tek bir satıra indirger. Bir koleksiyon içinde fiyatların ortalaması alınır ve ürün kodları sayılır. Her koleksiyon satıcı kodu ve satıcı adı için aynı değere sahip olacak şekilde oluşturulduğundan, DBMS bir koleksiyondaki her satırın bu nitelikler için aynı değere sahip olduğunu kesin olarak bilir. Bu nedenle, koleksiyon tek bir satıra indirgendiğinde, DBMS bu koleksiyon için V_CODE ve V_NAME'i görüntüleyebileceğini bilir çünkü koleksiyondaki tüm satırlar aynı değerlere sahiptir. Son olarak, ORDER BY cümlesi her koleksiyon için elde edilen satırları satıcı adına göre artan sırada sıralar.

Şimdi, aynı sorguyu ancak SELECT sütun listesine bir ek nitelik eklendiğini düşünün:

```
SEÇİNİZ      V_CODE, V_NAME, P_QOH, COUNT(P_CODE), AVG(P_PRICE)
FROM         ÜRÜN SATICIYA KATIL
             ON PRODUCT.V_CODE 5 VENDOR.V_CODE
GROUP BY     V_CODE, V_NAME
ORDER BY     V_NAME;
```

Bu sorgu yürütülmez ancak "bir GROUP BY ifadesi değil" hatası oluşturur. FROM cümlesi, PRODUCT ve VENDOR tablolarını birleştirmek için önceki sorgudaki ile tamamen aynı şekilde çalışır. GROUP BY cümlesi, satıcı kodu ve satıcı adına göre satır koleksiyonları oluşturmak için önceki sorgudaki ile tamamen aynı şekilde çalışır. SELECT sütun listesi, satıcı kodu, satıcı adı, ürün kodu ve ürün fiyatına ek olarak eldeki ürün miktarını da yansıtır. Hata, toplama işlevleri her bir satırı azaltmaya çalıştığında ortaya çıkar.

koleksiyonunu tek bir satıra dönüştürür. Toplamlar, ortalama alarak fiyatı düşürebilir, sayarak ürün kodunu düşürebilir, koleksiyondaki tüm satırlar aynı değere sahip olduğu için satıcı kodunu ve satıcı adını görüntüleyebilir, ancak eldeki miktar özelliği ne olacak? Bir koleksiyondaki her satır, o koleksiyondaki ürünler için eldeki miktar için farklı değerlere sahip olabilir. Sorgu P_QOH'a göre gruplama yapmaz, bu nedenle DBMS tüm satırların aynı değere sahip olduğunu kesin olarak bilemez. Sorgu, P_QOH özniteliğine bir toplama işlevi uygulamaz, bu nedenle DBMS, koleksiyondaki P_QOH değerini temsil edecek tek bir değerin nasıl hesaplanacağını bilemez. Bu nedenle bir hata oluşturulur. Bu hatayı düzeltmek için, tek bir değerin hesaplanabilmesi için ya P_QOH'a bir toplama işlevi uygulanmalıdır ya da P_QOH GROUP BY cümlesine eklenmelidir, böylece DBMS gruptaki her satırın bu öznitelik için aynı değere sahip olmasını zorunlu kılabilir. Bu iki olası çözümün sonuçlarındaki farka dikkat edin.

```
SEÇİNİZ      V_CODE, V_NAME, SUM(P_QOH) AS TOTALQTY, COUNT(P_CODE) AS
             NUMPRODS, AVG(P_PRICE) AS AVGPRICE
FROM         ÜRÜN SATICIYA KATIL
             ON PRODUCT.V_CODE 5 VENDOR.V_CODE
GROUP BY     V_CODE, V_NAME
ORDER BY     V_NAME;
```

Şekil 7.45, P_QOH'a uygulanan bir TOPLA işlevi ile elde edilen sonucu göstermektedir. Sonuç altı satırdan oluşmaktadır.

```
SEÇİNİZ      V_CODE, V_NAME, P_QOH, COUNT(P_CODE) AS NUMPRODS,
             AVG(P_PRICE) AS AVGPRICE
FROM         ÜRÜN SATICIYA KATIL
             ON PRODUCT.V_CODE 5 VENDOR.V_CODE
GROUP BY     V_CODE, V_NAME, P_QOH
ORDER BY     V_NAME;
```

V_CODE	V_NAME	TOTALQTY	NUMPRODS	AVGPRICE
21225	Bryson, Inc.	195	2	8.47
21231	D&E Supply	237	1	8.45
21344	Gomez Bros.	93	3	12.49
23119	Randsets Ltd.	38	2	41.97
24288	ORDVA, Inc.	25	3	155.59
25595	Rubicon Systems	38	3	89.63

Şekil 7.46, GROUP BY cümlesine P_QOH eklendiğinde ortaya çıkan sonucu göstermektedir. Sonuç 14 satırdan oluşmaktadır ve GROUP BY cümlesi tarafından oluşturulan koleksiyonlar artık aynı satır kümelerini içermediği için COUNT ve AVG fonksiyonlarının değerleri değişmiştir. Grup sayısı değişmiştir çünkü satırlar satıcı kodu ve satıcı adına göre gruplandırıldığında (Şekil 7.45'te olduğu gibi), P_QOH'daki değerler için satırlar arasında çeşitlilik vardı. Şekil 7.46'da koleksiyonlar, bir gruptaki tüm satırların V_CODE, V_NAME ve P_QOH için aynı değere sahip olmasını gerektirerek oluşturulmuştur.

GROUP BY cümlesine ek nitelikler eklemek, oluşturulan grupların sayısını her zaman değiştirmez. Örneğin, Şekil 7.44'teki sorguya P_QOH eklemek yerine V_STATE eklemeyi düşünün. Bu durumda, GROUP BY cümlesine V_STATE eklenmesi, ürün sayısı veya ortalama fiyat değerlerini değiştirmez çünkü Şekil 7.47'de gösterildiği gibi, V_CODE ve V_NAME üzerinde gruplama yapılarak oluşturulan her koleksiyonda V_STATE için yalnızca bir değer vardır.

Şekil 7.46 P_QOH'un Dahil Edilmesiyle Değişen Gruplar

V_CODE	V_NAME	P_QOH	NUMPRODS	AVGPRICE
21225	Bryson, Inc.	23	1	9.95
21225	Bryson, Inc.	172	1	6.99
21231	D&E Supply	237	1	8.45
21344	Gomez Bros.	18	1	17.49
21344	Gomez Bros.	32	1	14.99
21344	Gomez Bros.	43	1	4.99
23119	Randsets Ltd.	15	1	39.95
23119	Randsets Ltd.	23	1	43.99
24288	ORDVA, Inc.	6	1	99.87
24288	ORDVA, Inc.	8	1	109.92
24288	ORDVA, Inc.	11	1	256.99
25595	Rubicon Systems	8	1	109.99
25595	Rubicon Systems	12	1	38.95
25595	Rubicon Systems	18	1	119.95

Şekil 7.47 V_STATE Dahil Gruplar

V_CODE	V_NAME	V_STATE	NUMPRODS	AVGPRICE
21225	Bryson, Inc.	TN	2	8.47
21231	D&E Supply	TN	1	8.45
21344	Gomez Bros.	KY	3	12.49
23119	Randsets Ltd.	GA	2	41.97
24288	ORDVA, Inc.	TN	3	155.59
25595	Rubicon Systems	FL	3	89.63

Gördüğünüz gibi, gruplar ve toplama işlevleri kullanan sorgular oluşturulurken çok dikkatli olunmalıdır çünkü tek bir özniteliğin eklenmesi bile sorgu tarafından döndürülen sonuçları önemli ölçüde değiştirebilir.

7-8 c HAVING Cümlesi

Toplama fonksiyonları güçlüdür ve raporlamada sıkça kullanılır. Çoğu zaman, toplama fonksiyonları bir sorgunun SELECT sütun listesinde görünür. Sonuçları hesaplanan bir toplama değerine göre sıralamak için ORDER BY cümlesinde toplama fonksiyonlarını kullanmak da mümkündür. Bununla birlikte, verileri bir toplama değerine göre kısıtlamak biraz daha karmaşıktır ve bir **HAVING** cümlesinin kullanılmasını gerektirebilir. HAVING cümlesinin sözdizimi şöyledir:

SAHİP OLMAK
Seçilen satırları kısıtlamak için bir GROUP BY işleminin çıktısına uygulanan bir cümle.

SEÇİNİZ *sütun listesi*
FROM *tablelist*
[NEREDE *koşul listesi*]
[GROUP BY *sütun listesi*]
[HAVING *koşul listesi*]
[ORDER BY *sütun listesi* [ASC | DESC]];

HAVING cümlesi, SELECT deyimindeki WHERE cümlesi gibi çalışır. Ancak WHERE cümlesi tek tek satırlar için sütunlara ve ifadelere uygulanırken

HAVING cümlesi, bir GROUP BY işleminin çıktısına uygulanır. Örneğin, her bir satıcı tarafından sağlanan envanterdeki ürün sayısını listelemek istediğinizi varsayalım. Ancak bu kez listelemeyi, ürün fiyatları ortalama 10 $'dan az olan satıcıların ürünlerini saymakla sınırlamak istiyorsunuz. Sorgu, Şekil 7.48'de gösterildiği gibi hem bir GROUP BY cümlesi hem de bir HAVING cümlesi gerektirir.

```
SEÇİNİZ          V_CODE, COUNT(P_CODE) AS NUMPRODS
FROM             ÜRÜN
GROUP BY         V_CODE
SAHİP OLMAK AVG(P_PRICE) , 10
SIRALAMA ÖLÇÜTÜ   V_CODE;
```

Şekil 7.48 HAVING Cümlesinin Uygulanması

V_CODE	NUMPRODS
21225	2
21231	1

HAVING cümlesi yerine WHERE cümlesini kullanırsanız, Şekil 7.48'deki sorgu bir hata mesajı üretecektir. Bu, bir sorgunun hem WHERE cümlesi hem de HAVING cümlesi içeremeyeceği anlamına gelmez, sadece cümleler farklı şeyler yapar. WHERE, satırları kısıtlamak için kullanılır ve GROUP BY cümlesinden önce yürütülür. WHERE, GROUP BY'den önce yürütüldüğünden, WHERE bir toplama işlevi içeremez çünkü toplama işlevi tarafından ihtiyaç duyulan koleksiyonlar henüz mevcut değildir. HAVING cümlesi grupları kısıtlamak için kullanılır ve GROUP BY sonra yürütülür. HAVING, toplama işlevleri içerebilir çünkü koleksiyonlar HAVING yürütülmeden önce GROUP BY cümlesi tarafından oluşturulur. HAVING *grupları* kısıtlar. HAVING cümlesinin bir gruptaki bazı satırları kısıtlayıp diğerlerini serbest bırakması mümkün değildir. HAVING tüm grubu tutar veya ortadan kaldırır, bu nedenle HAVING cümlesindeki koşul tüm grup için geçerli olmalıdır. Bu nedenle, HAVING cümlelerinin yalnızca toplama işlevleri içermesine izin , aynı zamanda neredeyse her zaman içerirler.

Birden fazla cümleyi ve toplama işlevini birleştirebilirsiniz. Örneğin, aşağıdaki SQL ifadesini düşünün:

```
SEÇİNİZ          V_CODE, V_NAME, SUM(P_QOH *  P_PRICE) AS TOTCOST
FROM             ÜRÜN SATICIYA KATIL
                 ON PRODUCT.V_CODE 5 VENDOR.V_CODE
NEREDE           P_DISCOUNT -  0
GRUPLAMA         V_CODE, V_NAME
SAHİP OLMAK (SUM(P_QOH *  P_PRICE) . 500)
ORDER BY         SUM(P_QOH *  P_PRICE) DESC;
```

Bu ifade aşağıdakileri yapar:

- Ortak öznitelik olarak V_CODE kullanarak ürün ve satıcı tablolarını birleştirir
- Yalnızca 0'dan büyük indirime sahip satırlarla sınırlandırır
- Kalan satırları V_CODE ve V_NAME'e göre koleksiyonlar halinde gruplar
- Her gruptaki ürünlerin toplam maliyetini toplar
- Yalnızca toplamları 500 $'ı aşan gruplarla sınırlandırır
- Sonuçları toplam maliyete göre azalan sırada listeler

HAVING ve ORDER BY cümlelerinde kullanılan sözdizimine dikkat edin; her iki durumda da SELECT deyiminin sütun listesinde kullanılan sütun ifadesini (formül) belirtmeniz gerekir, bunun yerine

sütun takma adından (TOTCOST) daha büyüktür. Bazı RDBMS'ler sütun ifadesini sütun takma adıyla değiştirmenize izin verirken, diğerleri izin vermez.

7-9 Alt Sorgular

İlişkisel bir veritabanında birleştirmelerin kullanılması, iki veya daha fazla tablodan bilgi almanıza olanak tanır. Örneğin, aşağıdaki sorgu CUSTOMER ve INVOICE tablolarını birleştirerek müşteri verilerini ilgili faturalarıyla birlikte almanızı sağlar.

```
SEÇİNİZ     INV_NUMBER, INVOICE.CUS_CODE, CUS_LNAME, CUS_FNAME
FROM        CUSTOMER C JOIN INVOICE I ON C.CUS_CODE 5 I.CUS_CODE;
```

Önceki sorguda, her iki tablodaki (CUSTOMER ve INVOICE) veriler bir kerede işlenir ve ortak CUS_CODE değerlerine sahip satırlar eşleştirilir.

Bununla birlikte, verilerin *diğer* işlenmiş verilere dayalı olarak işlenmesi genellikle gereklidir. Örneğin, ürün sağlamayan satıcıların bir listesini oluşturmak istediğinizi varsayalım. (SATICI tablosundaki tüm satıcıların ürün sağlamadığını, bazılarının yalnızca *potansiyel* satıcı olduğunu hatırlayın). Daha önce, aşağıdaki sorguyu yazarak böyle bir liste oluşturabileceğinizi öğrenmiştiniz:

```
SEÇİNİZ     V_CODE, V_NAME
FROM        ÜRÜN HAKKI SATICIYA KATIL
            ON PRODUCT.V_CODE 5 VENDOR.V_CODE
NEREDE      P_CODE BOŞTUR;
```

alt sorgu
Başka bir içine gömülmüş (veya yuvalanmış) bir sorgu. *İç içe sorgu* veya *iç sorgu* olarak da bilinir.

Ancak, bu sonuç aşağıdaki gibi bir **alt sorgu** kullanılarak da bulunabilir:

```
SEÇİNİZ     V_CODE, V_NAME
FROM        SATICI
NEREDE      V_CODE NOT IN (SELECT V_CODE FROM PRODUCT WHERE
            V_CODE IS NOT NULL);
```

Benzer şekilde, fiyatı ortalama ürün fiyatına eşit veya daha yüksek olan tüm ürünlerin bir listesini oluşturmak için aşağıdaki sorguyu yazabilirsiniz:

```
SEÇİNİZ     P_CODE, P_PRICE
FROM        ÜRÜN
NEREDE      P_PRICE .5 (SELECT AVG(P_PRICE) FROM PRODUCT);
```

Her iki sorguda da daha önce bilinmeyen bilgilere ulaşmanız gerekiyordu:

- Hangi satıcılar ürün sağlıyor?
- Tüm ürünlerin ortalama fiyatı nedir?

Her iki durumda da, gerekli bilgileri oluşturmak için bir alt sorgu kullandınız ve bu bilgiler daha sonra asıl sorgu için girdi olarak kullanılabilir. Alt sorgular için aşağıdaki anahtar karakteristikleri hatırlamanız gerekir:

- Alt sorgu, başka bir sorgunun içindeki bir (SELECT deyimi).
- Bir alt sorgu normalde parantez içinde ifade edilir.
- SQL ifadesindeki ilk sorgu *dış sorgu* olarak bilinir.
- SQL deyiminin içindeki sorgu *iç sorgu* olarak bilinir.
- Önce iç sorgu yürütülür.
- Bir iç sorgunun çıktısı, dış için girdi olarak kullanılır.
- SQL ifadesinin tamamı bazen *iç* içe *sorgu* olarak adlandırılır.

Bu bölümde, alt sorguların pratik kullanımı hakkında daha fazla bilgi edineceksiniz. Bir alt sorgunun, başka bir sorguya bir veya daha fazla değer döndürmek için SELECT deyiminin kullanımına dayandığını zaten biliyorsunuz, ancak alt sorguların çok çeşitli kullanımları vardır. Örneğin, bir değerin veya değerler listesinin (birden fazla satıcı kodu veya bir tablo gibi) beklendiği INSERT, UPDATE veya DELETE gibi bir SQL veri işleme dili (DML) deyimi içinde bir alt sorgu kullanabilirsiniz.

Alt sorgu her zaman bir karşılaştırma veya atama ifadesinin sağ tarafındadır. Ayrıca, bir alt sorgu bir veya daha fazla değer döndürebilir. Kesin olmak gerekirse, alt sorgu aşağıdakileri döndürebilir:

- *Tek bir değer (bir sütun ve bir satır)*. Bu alt sorgu, bir karşılaştırma ifadesinin sağ tarafında olduğu gibi, tek bir değerin beklendiği her yerde kullanılır. Örnek olarak, ürünlerin ortalama fiyatından daha yüksek fiyata sahip ürünleri aldığınız önceki sorgu verilebilir.
- *Bir değerler listesi (bir sütun ve birden çok satır)*. Bu tür bir alt sorgu, IN cümlesini kullanırken olduğu gibi bir değerler listesinin beklendiği her yerde kullanılır; örneğin, satıcı kodunu yukarıdaki gibi bir satıcılar listesiyle karşılaştırırken. Yine bu durumda, yalnızca bir veri sütunu birden fazla değer örneğine sahiptir. Bu tür bir alt sorgu, WHERE koşullu ifadesinde IN işleciyle birlikte sıklıkla kullanılır.
- *Sanal bir tablo (çok sütunlu, çok satırlı değerler kümesi)*. Bu tür bir alt sorgu, FROM cümlesini kullanırken olduğu gibi, bir tablonun beklendiği her yerde kullanılabilir. Bu bölümün ilerleyen kısımlarında bir örnek göreceksiniz.

Bir alt sorgunun hiçbir değer döndürmeyebileceğini unutmayın; bu bir NULL'dur. Bu gibi durumlarda, dış sorgunun çıktısı, alt sorgunun kullanıldığı yere bağlı olarak (bir karşılaştırmada, ifadede veya tablo kümesinde) bir hata veya null boş küme ile sonuçlanabilir.

İlerleyen bölümlerde, veritabanından veri almak için SELECT deyimi içinde alt sorguların nasıl yazılacağını öğreneceksiniz.

7-9 a WHERE Alt Sorguları

En yaygın alt sorgu türü, bir WHERE karşılaştırma ifadesinin sağ tarafında bir iç SELECT alt sorgusu kullanır. Örneğin, fiyatı ortalama ürün fiyatına eşit veya daha yüksek olan tüm ürünleri bulmak için aşağıdaki sorguyu yazarsınız:

```
SEÇİNİZ     P_CODE, P_PRICE
FROM        ÜRÜN
NEREDE      P_PRICE .5 (SELECT AVG(P_PRICE) FROM PRODUCT);
```

Önceki sorgunun çıktısı Şekil 7.49'da gösterilmektedir. Bu sorgu türünün, bir .., ,, 5, .5 veya ,5 koşullu ifadede kullanıldığında, yalnızca bir değer (bir sütun, bir satır) döndüren bir alt sorgu gerektirdiğini unutmayın. Alt sorgu tarafından oluşturulan değer, karşılaştırılabilir bir veri türünde olmalıdır; karşılaştırma sembolünün solundaki öznitelik bir karakter türüyse, alt sorgu bir karakter dizesi döndürmelidir. Ayrıca, sorgu tek bir değerden daha fazlasını döndürürse, DBMS bir hata üretecektir.

P_CODE	P_PRICE
11QER/31	109.99
2232/QTY	109.92
2232/QWE	99.87
89-WRE-Q	256.99
WR3/TT3	119.95

Not

Bir sütun adının beklendiği her yerde bir ifade kullanabilirsiniz. Hangi ürünün en yüksek envanter değerine sahip olduğunu bilmek istediğinizi varsayalım. Cevabı bulmak için aşağıdaki sorguyu yazabilirsiniz:

```
SEÇİNİ      *
Z           ÜRÜN
FROM        P_QOH *  P_PRICE 5 (SELECT MAX(P_QOH *  P_PRICE) FROM PRODUCT);
NERED
EN
```

Alt sorgular birleştirmelerle birlikte de kullanılabilir. Örneğin, aşağıdaki sorgu pençe çekiç siparişi veren tüm müşterileri listeler:

```
SEÇİNİZ     DISTINCT CUS_CODE, CUS_LNAME, CUS_FNAME
FROM        CUSTOMER JOIN INVOICE USING (CUS_CODE)
            JOIN LINE USING (INV_NUMBER)
            JOIN PRODUCT USING (P_CODE)
NEREDE      P_CODE 5 (SELECT P_CODE FROM PRODUCT WHERE
            P_ DESCRIPT 5 'Claw hammer');
```

Sorgunun sonucu Şekil 7.50'de gösterilmektedir.

Şekil 7.50 Pençe Çekiç Siparişi Veren Müşteriler

CUS_CODE	CUS_LNAME	CUS_FNAME
10011	Dunne	Leona
10014	Orlando	Myron

Yukarıdaki örnekte, iç sorgu pençe çekiç için P_CODE'u bulur. P_CODE daha sonra seçilen satırları LINE tablosundaki P_CODE'un "Pençe çekici" için P_CODE ile eşleştiği satırlarla sınırlamak için kullanılır. Önceki sorgunun bu şekilde yazılabileceğini unutmayın:

```
SEÇİNİZ     DISTINCT CUSTOMER.CUS_CODE, CUS_LNAME, CUS_FNAME
FROM        CUSTOMER JOIN INVOICE ON CUSTOMER.CUS_CODE 5
            INVOICE.CUS_CODE
            JOIN LINE ON INVOICE.INV_NUMBER 5 .INV_NUMBER JOIN
            PRODUCT ON PRODUCT.P_CODE 5 LINE.P_CODE
NEREDE      P_DESCRIPT 5 'Pençe çekici';
```

Orijinal sorgu birden fazla ürün tanımında "Claw hammer" dizesiyle karşılaşırsa bir hata mesajı alırsınız. Bir değeri bir değerler listesiyle karşılaştırmak için, bir sonraki bölümde gösterildiği gibi bir IN işleneni kullanmanız gerekir.

7-9 b IN Alt Sorgular

Bir çekiç veya herhangi bir testere ya da testere bıçağı satın alan tüm müşterileri bulmak isteseydiniz ne olurdu? Ürün tablosunda iki tür çekiç vardır: pençe çekiç ve balyoz. Ayrıca, testere bıçakları ve dekupaj testereleri de dahil olmak üzere, ürün açıklamalarında birden fazla "testere" içeren ürün bulunmaktadır. Bu gibi durumlarda, P_CODE'u tek bir ürün koduyla (tek bir değer) değil, ürün kodu değerlerinin bir listesiyle karşılaştırmanız gerekir. Tek bir özniteliği bir değerler listesiyle karşılaştırmak istediğinizde IN operatörünü kullanırsınız. P_CODE değerleri bilinmediğinde