Sorgular

Sorgu Mekaniği

Select ifadesini incelemeden önce, sorguların MySQL sunucusu (veya bu konuda herhangi bir veritabanı sunucusu) tarafından nasıl yürütüldüğüne bakmak ilginç olabilir. Eğer mysql komut satırı aracını kullanıyorsanız (ki öyle olduğunuzu varsayıyorum), o zaman kullanıcı adınızı ve şifrenizi (ve MySQL sunucusu farklı bir sunucuda çalışıyorsa muhtemelen bir ana bilgisayar adını) girerek MySQL sunucusunda oturum açmış olursunuz. bilgisayar). Sunucu, kullanıcı adınızın ve şifrenizin doğru olduğunu onayladıktan sonra, kullanmanız için bir veritabanı bağlantısı oluşturulur. Bu bağlantı, talep eden uygulama tarafından (bu durumda, mysql aracıdır), uygulama bağlantıyı bırakana kadar (yani, çıkış yazmanın bir sonucu olarak) veya sunucu bağlantıyı kapatana kadar (yani, sunucu kapatılır). MySQL sunucusuna yapılan her bağlantıya, ilk oturum açtığınızda size gösterilen bir tanımlayıcı atanır:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.15 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
```

Bu durumda bağlantı kimliğim 11'dir. Bu bilgi, saatlerce çalışan hatalı biçimlendirilmiş bir sorgu gibi bir şeyler ters giderse veritabanı yöneticiniz için yararlı olabilir, bu nedenle not almak isteyebilirsiniz.
Sunucu, kullanıcı adınızı ve parolanızı doğruladıktan ve size bir bağlantı verdiğinde, sorguları (diğer SQL ifadeleriyle birlikte) yürütmeye hazırsınız demektir. Sunucuya her sorgu gönderildiğinde, sunucu deyimin yürütülmesinden önce aşağıdakileri kontrol eder:

- İfadeyi yürütme izniniz var mı?
- İstenen verilere erişim izniniz var mı?
- İfade sözdiziminiz doğru mu?

İfadeniz bu üç testi geçerse, sorgunuz, görevi sorgunuzu yürütmenin en verimli yolunu belirlemek olan sorgu iyileştiriciye teslim edilir. Optimize edici, from yan tümcenizde belirtilen tabloların hangi sırayla birleştirileceğine ve hangi dizinlerin mevcut olduğuna bakar ve ardından sunucunun sorgunuzu yürütmek için kullandığı bir yürütme planı seçer.
Sunucu, sorgunuzu yürütmeyi bitirdiğinde, sonuç kümesi, çağıran uygulamaya döndürülür. Sonuç kümesi sadece satırlar ve sütunlar içeren başka bir tablodur. Sorgunuz herhangi bir sonuç vermezse, mysql aracı size aşağıdaki örneğin sonunda bulunan mesajı gösterecektir:

```
mysql> SELECT first_name, last_name
    -> FROM customer
    -> WHERE last_name = 'ZIEGLER';
Empty set (0.02 sec)
```

Sorgu bir veya daha fazla satır döndürürse, mysql aracı, bir sonraki örnekte gösterildiği gibi, sütun başlıkları ekleyerek ve -, | ve sembollerini kullanarak sütunların etrafına kutular oluşturarak sonuçları biçimlendirir:

```
mysql> SELECT *
    -> FROM category;
+-------------+-------------+---------------------+
| category_id | name        | last_update         |
+-------------+-------------+---------------------+
|           1 | Action      | 2006-02-15 04:46:27 |
|           2 | Animation   | 2006-02-15 04:46:27 |
|           3 | Children    | 2006-02-15 04:46:27 |
|           4 | Classics    | 2006-02-15 04:46:27 |
|           5 | Comedy      | 2006-02-15 04:46:27 |
|           6 | Documentary | 2006-02-15 04:46:27 |
|           7 | Drama       | 2006-02-15 04:46:27 |
|           8 | Family      | 2006-02-15 04:46:27 |
|           9 | Foreign     | 2006-02-15 04:46:27 |
|          10 | Games       | 2006-02-15 04:46:27 |
|          11 | Horror      | 2006-02-15 04:46:27 |
|          12 | Music       | 2006-02-15 04:46:27 |
|          13 | New         | 2006-02-15 04:46:27 |
|          14 | Sci-Fi      | 2006-02-15 04:46:27 |
|          15 | Sports      | 2006-02-15 04:46:27 |
|          16 | Travel      | 2006-02-15 04:46:27 |
+-------------+-------------+---------------------+
16 rows in set (0.02 sec)
```

Bu sorgu, kategori tablosundaki tüm satırların üç sütununu da döndürür. Son veri satırı görüntülendikten sonra, mysql aracı size kaç satır döndürüldüğünü bildiren bir mesaj görüntüler, bu durumda bu sayı 16'dır.

Sorgu Cümleleri

Birkaç bileşen veya tümce, select ifadesini oluşturur. MySQL (seçme maddesi) kullanırken bunlardan sadece biri zorunlu olsa da, genellikle mevcut altı maddeden en az ikisini veya üçünü dahil edeceksiniz.

Select	Sorgunun sonuç kümesine hangi sütunların ekleneceğini belirler
From	Verilerin alınacağı tabloları ve tabloların nasıl birleştirileceğini tanımlar
Where	İstenmeyen verileri filtreler
Group by	Ortak sütun değerlerine göre satırları gruplamak için kullanılır
Having	İstenmeyen grupları filtreler
Order by	Bir veya daha fazla sütun tarafından ayarlanan nihai sonucun satırlarını sıralar

Select

Select cümlesi bir select deyiminin ilk cümlesi olsa da, veritabanı sunucusunun değerlendirdiği son cümlelerden biridir. Bunun nedeni, nihai sonuç kümesine nelerin dahil edileceğini belirlemeden önce, nihai sonuç kümesine dahil edilebilecek tüm olası sütunları bilmeniz gerekmesidir. Bu nedenle, select yan tümcesinin rolünü tam olarak anlamak için, from yan tümcesi hakkında biraz bilgi sahibi olmanız gerekecektir. İşte başlamak için bir sorgu:

```
mysql> SELECT *
    -> FROM language;
+-------------+----------+---------------------+
| language_id | name     | last_update         |
+-------------+----------+---------------------+
|           1 | English  | 2006-02-15 05:02:19 |
|           2 | Italian  | 2006-02-15 05:02:19 |
|           3 | Japanese | 2006-02-15 05:02:19 |
|           4 | Mandarin | 2006-02-15 05:02:19 |
|           5 | French   | 2006-02-15 05:02:19 |
|           6 | German   | 2006-02-15 05:02:19 |
+-------------+----------+---------------------+
6 rows in set (0.03 sec)
```

Bu sorguda, from yan tümcesi tek bir tabloyu listeler ve select yan tümcesi, dil tablosundaki tüm sütunların (* ile gösterilen) sonuç kümesine dahil edilmesi gerektiğini belirtir. Bu sorgu şu şekilde açıklanabilir:

Dil tablosundaki tüm sütunları ve tüm satırları bana göster.

Tüm sütunları yıldız işaretiyle belirtmenin yanı sıra, açıkça şunları yapabilirsiniz: ilgilendiğiniz sütunları adlandırın, örneğin:

```
mysql> SELECT language_id, name, last_update
    -> FROM language;
+-------------+----------+---------------------+
| language_id | name     | last_update         |
+-------------+----------+---------------------+
|           1 | English  | 2006-02-15 05:02:19 |
|           2 | Italian  | 2006-02-15 05:02:19 |
|           3 | Japanese | 2006-02-15 05:02:19 |
|           4 | Mandarin | 2006-02-15 05:02:19 |
|           5 | French   | 2006-02-15 05:02:19 |
|           6 | German   | 2006-02-15 05:02:19 |
+-------------+----------+---------------------+
6 rows in set (0.00 sec)
```

Dil tablosundaki tüm sütunlar select yan tümcesinde adlandırıldığından, sonuçlar ilk sorguyla aynıdır. Dil tablosuna sütunların yalnızca bir alt kümesini dahil etmeyi de seçebilirsiniz:

```
mysql> SELECT name
    -> FROM language;
+----------+
| name     |
+----------+
| English  |
| Italian  |
| Japanese |
| Mandarin |
| French   |
| German   |
+----------+
6 rows in set (0.00 sec)
```

Select yan tümcesi sorgu sonuç kümesinden, olası tüm sütunlardan hangisinin listeye dahil edilmesi gerektiğini belirler.
Yalnızca tablodaki sütunları veya from yan tümcesinde belirtilen tabloları dahil etmekle sınırlı olsaydınız, işler oldukça sıkıcı olurdu. Bununla birlikte, aşağıdaki gibi şeyler ekleyerek, seçim yan tümcenizde bazı şeyleri renklendirebilirsiniz:

- Sayılar veya dizeler gibi değişmezler
- transaction.amount * -1 gibi ifadeler
- ROUND(transaction.amount, 2) gibi yerleşik fonksiyon çağrıları
- Kullanıcı tanımlı fonksiyon çağrıları

Sonraki sorgu, çalışan tablosuna karşı tek bir sorguda bir tablo sütunu, bir hazır bilgi, bir ifade ve yerleşik bir fonksiyon çağrısının kullanımını gösterir:

```
mysql> SELECT language_id,
    ->   'COMMON' language_usage,
    ->   language_id * 3.1415927 lang_pi_value,
    ->   upper(name) language_name
    -> FROM language;
+-------------+----------------+---------------+---------------+
| language_id | language_usage | lang_pi_value | language_name |
+-------------+----------------+---------------+---------------+
|           1 | COMMON         |     3.1415927 | ENGLISH       |
|           2 | COMMON         |     6.2831854 | ITALIAN       |
|           3 | COMMON         |     9.4247781 | JAPANESE      |
|           4 | COMMON         |    12.5663708 | MANDARIN      |
|           5 | COMMON         |    15.7079635 | FRENCH        |
|           6 | COMMON         |    18.8495562 | GERMAN        |
+-------------+----------------+---------------+---------------+
6 rows in set (0.04 sec)
```

Yalnızca yerleşik bir fonksiyonu yürütmeniz veya basit bir ifadeyi değerlendirmeniz gerekiyorsa, from yan tümcesini tamamen atlayabilirsiniz. İşte bir örnek:

```
mysql> SELECT version(),
    ->   user(),
    ->   database();
+-----------+----------------+------------+
| version() | user()         | database() |
+-----------+----------------+------------+
| 8.0.15    | root@localhost | sakila     |
+-----------+----------------+------------+
1 row in set (0.00 sec)
```

Bu sorgu yalnızca üç yerleşik fonksiyonu çağırdığından ve herhangi bir tablodan veri almadığından, from yan tümcesine gerek yoktur.

Sütun Takma Adları

MySQL aracı, sorgularınız tarafından döndürülen sütunlar için etiketler oluştursa da, kendi etiketlerinizi atamak isteyebilirsiniz. Bir tablodan bir sütuna yeni bir etiket atamak isteyebilirsiniz (eğer yanlış veya belirsiz bir şekilde adlandırılmışsa), Fonksiyon çağrılarındaki sonuç kümelerinde takma ad kullanmak neredeyse zorunludur. Select yan tümcenizin her öğesinden sonra bir sütun takma adı ekleyerek bunu yapabilirsiniz. Üç sütun için sütun takma adlarını içeren dil tablosuna yönelik önceki sorgu:
Select yan tümcesine bakarsanız, ikinci, üçüncü ve dördüncü sütunlardan sonra

```
mysql> SELECT language_id,
    ->   'COMMON' language_usage,
    ->   language_id * 3.1415927 lang_pi_value,
    ->   upper(name) language_name
    -> FROM language;
+-------------+----------------+---------------+---------------+
| language_id | language_usage | lang_pi_value | language_name |
+-------------+----------------+---------------+---------------+
|           1 | COMMON         |     3.1415927 | ENGLISH       |
|           2 | COMMON         |     6.2831854 | ITALIAN       |
|           3 | COMMON         |     9.4247781 | JAPANESE      |
|           4 | COMMON         |    12.5663708 | MANDARIN      |
|           5 | COMMON         |    15.7079635 | FRENCH        |
|           6 | COMMON         |    18.8495562 | GERMAN        |
+-------------+----------------+---------------+---------------+
6 rows in set (0.04 sec)
```

anguage_usage, lang_pi_value, ve language_name sütun takma adlarının nasıl eklendiğini görebilirsiniz. Sütun takma adları yerindeyken çıktının anlaşılmasının daha kolay olduğu kabul edilir ve sorguyu etkileşimli olarak mysql aracıyla değil, Java veya Python içinden veriyorsanız, programlı olarak çalışmak daha kolay olur. Sütun takma adlarınızı daha da öne çıkarmak için, aşağıdaki gibi, takma ad adından önce as anahtar sözcüğünü kullanma seçeneğiniz de vardır:

```
mysql> SELECT language_id,
    ->   'COMMON' AS language_usage,
    ->   language_id * 3.1415927 AS lang_pi_value,
    ->   upper(name) AS language_name
    -> FROM language;
```

Yinelenenleri Kaldırma

Bazı durumlarda, bir sorgu yinelenen veri satırları döndürebilir. Örneğin, bir filmde yer alan tüm oyuncuların kimliklerini alacak olsaydınız, aşağıdakileri görürsünüz:

```
mysql> SELECT actor_id FROM film_actor ORDER BY actor_id;
+----------+
| actor_id |
+----------+
|        1 |
|        1 |
|        1 |
|        1 |
|        1 |
|        1 |
|        1 |
|        1 |
|        1 |
|        1 |
...
|      200 |
|      200 |
|      200 |
|      200 |
|      200 |
|      200 |
|      200 |
|      200 |
|      200 |
+----------+
5462 rows in set (0.01 sec)
```

Bazı oyuncular birden fazla filmde yer aldığından, aynı oyuncu kimliğini birden çok kez göreceksiniz. Bu durumda muhtemelen istediğiniz şey, göründükleri her film için tekrarlanan oyuncu kimliklerini görmek yerine, farklı aktör gruplarıdır. Bunu, aşağıda gösterildiği gibi, select anahtar sözcüğünden hemen sonra **distinct** anahtar kelimeyi ekleyerek başarabilirsiniz:

Sonuç kümesi artık, bir oyuncunun her film görünümü için bir tane olmak üzere 5.462 satır yerine, her bir farklı oyuncu için bir tane olmak üzere 200 satır içeriyor.

```
mysql> SELECT DISTINCT actor_id FROM film_actor ORDER BY actor_id;
+----------+
| actor_id |
+----------+
|        1 |
|        2 |
|        3 |
|        4 |
|        5 |
|        6 |
|        7 |
|        8 |
|        9 |
|       10 |
...
|      192 |
|      193 |
|      194 |
|      195 |
|      196 |
|      197 |
|      198 |
|      199 |
|      200 |
+----------+
200 rows in set (0.01 sec)
```

Sunucunun yinelenen verileri kaldırmasını istemiyorsanız veya sonuç kümenizde yinelenen veriler olmayacağından eminseniz, **distinct** belirtmek yerine **all** anahtar sözcüğünü belirtebilirsiniz. Ancak, **all** anahtar sözcüğü varsayılandır ve hiçbir zaman açıkça adlandırılması gerekmez, bu nedenle çoğu programcı sorgularına **all** dahil etmez.

distinct bir sonuç kümesi oluşturmanın, verilerin sıralanmasını gerektirdiğini ve bunun da büyük sonuç kümeleri için zaman alıcı olabileceğini unutmayın. Hiçbir kopya olmadığından emin olmak için **distinct** kullanma tuzağına düşmeyin; bunun yerine, çalıştığınız verileri anlamak için zaman ayırın, böylece yinelemelerin mümkün olup olmadığını anlarsınız.

From

Şimdiye kadar, from yan tümceleri tek bir tablo içeren sorgular gördünüz. Çoğu SQL kitabı from yan tümcesini bir veya daha fazla tablonun basit bir listesi olarak tanımlasa da, from yan tümcesi, bir sorgu tarafından kullanılan tabloları ve tabloları birbirine bağlamanın yollarını tanımlar.

Tablolar

Tablo terimi ile karşı karşıya kaldığında, çoğu insan bir veritabanında depolanan bir dizi ilgili satır düşünür. Bu, bir tür tabloyu tanımlasa da, verilerin nasıl depolanabileceğine dair herhangi bir fikri ortadan kaldırarak ve yalnızca ilgili satırlar üzerinde yoğunlaşarak kelimeyi daha genel bir şekilde tanımlamak gerekirse. Dört farklı türde tablo düşünülebilir:

- Kalıcı tablolar (yani, create table deyimi kullanılarak oluşturulmuş)
- Türetilmiş tablolar (yani, bir alt sorgu tarafından döndürülen ve bellekte tutulan satırlar)
- Geçici tablolar (yani bellekte tutulan uçucu veriler)
- Sanal tablolar (yani, create view deyimi kullanılarak oluşturulmuş)

Bu tablo türlerinin her biri, bir sorgunun from yan tümcesine dahil edilebilir. Şimdiye kadar, from yan tümcesine kalıcı bir tablo eklemek konusunu gördük bu yüzden bir from yan tümcesinde başvurulabilecek diğer tablo türlerini kısaca açıklamak gerekirse.

Türetilmiş (alt sorgu tarafından oluşturulan) tablolar

Alt sorgu, başka bir sorgu içinde yer alan bir sorgudur. Alt sorgular parantez içine alınır ve bir select deyiminin çeşitli bölümlerinde bulunabilir; Ancak from yan tümcesi içinde bir alt sorgu, diğer tüm sorgu yan tümcelerinden görülebilen ve from yan tümcesinde adlandırılan diğer tablolarla etkileşime girebilen türetilmiş bir tablo oluşturma rolüne hizmet eder. İşte basit bir örnek:

```
mysql> SELECT concat(cust.last_name, ', ', cust.first_name) full_name
    -> FROM
    ->  (SELECT first_name, last_name, email
    ->   FROM customer
    ->   WHERE first_name = 'JESSIE'
    ->  ) cust;
```

```
+---------------+
| full_name     |
+---------------+
| BANKS, JESSIE |
| MILAM, JESSIE |
+---------------+
2 rows in set (0.00 sec)
```

Bu örnekte, müşteri tablosuna yönelik bir alt sorgu, üç sütun döndürür ve üst sorgu, kullanılabilir üç sütundan ikisine başvurdu. Alt sorguya, bu durumda cust olan diğer adı aracılığıyla üst sorgu tarafından başvurulur. Cust'taki veriler, sorgu süresi boyunca bellekte tutulur ve daha sonra atılır. Bu, from yan tümcesindeki bir alt sorgunun basitleştirilmiş ve özellikle kullanışlı olmayan bir örneğidir.

Geçici tablolar

Uygulamalar farklılık gösterse de, her ilişkisel veritabanı geçici tabloları tanımlama becerisine izin verir. Bu tablolar kalıcı tablolar gibi görünür, ancak geçici bir tabloya eklenen veriler bir noktada (genellikle bir işlemin sonunda veya veritabanı oturumunuz kapatıldığında) kaybolur. Soyadları J ile başlayan aktörlerin geçici olarak nasıl saklanabileceğini gösteren basit bir örnek:

```
mysql> CREATE TEMPORARY TABLE actors_j
    ->  (actor_id smallint(5),
    ->   first_name varchar(45),
    ->   last_name varchar(45)
    ->  );
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO actors_j
    -> SELECT actor_id, first_name, last_name
    -> FROM actor
    -> WHERE last_name LIKE 'J%';
Query OK, 7 rows affected (0.03 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM actors_j;
+----------+------------+-----------+
| actor_id | first_name | last_name |
+----------+------------+-----------+
|      119 | WARREN     | JACKMAN   |
|      131 | JANE       | JACKMAN   |
|        8 | MATTHEW    | JOHANSSON |
|       64 | RAY        | JOHANSSON |
|      146 | ALBERT     | JOHANSSON |
|       82 | WOODY      | JOLIE     |
|       43 | KIRK       | JOVOVICH  |
+----------+------------+-----------+
7 rows in set (0.00 sec)
```

Kaynak: Alan Beaulieu, 2020, Learning SQL, 3rd Edition, O'Reilly.

Bu yedi satır geçici olarak hafızada tutulur ve oturumunuz kapatıldıktan sonra kaybolur.

Views

View, veri sözlüğünde depolanan bir sorgudur. Bir tablo gibi görünür ve çalışır, ancak bir View ile ilişkili veri yoktur (bu yüzden ona sanal tablo denilebilir). Bir View'e karşı bir sorgu yayınladığınızda, sorgunuz, yürütülecek son bir sorgu oluşturmak için View tanımıyla birleştirilir.
Örneklemek gerekirse, çalışan tablosunu sorgulayan ve mevcut sütunlardan dördünü içeren bir View tanımı aşağıda verilmiştir:

```
mysql> CREATE VIEW cust_vw AS
    -> SELECT customer_id, first_name, last_name, active
    -> FROM customer;
Query OK, 0 rows affected (0.12 sec)
```

View oluşturulduğunda, hiçbir ek veri oluşturulmaz veya depolanmaz: sunucu, gelecekte kullanmak üzere yalnızca select ifadesini saklar. Artık görünüm mevcut olduğuna göre, şu şekilde olduğu gibi ona karşı sorgular gönderebilirsiniz:

```
mysql> SELECT first_name, last_name
    -> FROM cust_vw
    -> WHERE active = 0;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| SANDRA     | MARTIN    |
| JUDITH     | COX       |
| SHEILA     | WELLS     |
| ERICA      | MATTHEWS  |
| HEIDI      | LARSON    |
| PENNY      | NEAL      |
| KENNETH    | GOODEN    |
| HARRY      | ARCE      |
| NATHAN     | RUNYON    |
| THEODORE   | CULP      |
| MAURICE    | CRAWLEY   |
| BEN        | EASTER    |
| CHRISTIAN  | JUNG      |
| JIMMIE     | EGGLESTON |
| TERRANCE   | ROUSH     |
+------------+-----------+
15 rows in set (0.00 sec)
```

Kaynak: Alan Beaulieu, 2020, Learning SQL, 3rd Edition, O'Reilly.

Görünümler, sütunları kullanıcılardan gizlemek ve karmaşık veritabanı tasarımlarını basitleştirmek dahil olmak üzere çeşitli nedenlerle oluşturulur.

Tabloları Bağlama

from yan tümcesi tanımından ikinci önemli nokta, from yan tümcesinde birden fazla tablo görünüyorsa, tabloları bağlamak için kullanılan koşulların da dahil edilmesi gerektiğidir. Bu, MySQL veya başka bir veritabanı sunucusu için bir gereklilik değildir, ancak birden çok tabloyu birleştirmenin ANSI onaylı yöntemidir ve çeşitli veritabanı sunucuları arasında en taşınabilir olanıdır.

```
mysql> SELECT customer.first_name, customer.last_name,
    ->   time(rental.rental_date) rental_time
    -> FROM customer
    ->   INNER JOIN rental
    ->   ON customer.customer_id = rental.customer_id
    -> WHERE date(rental.rental_date) = '2005-06-14';
+------------+-----------+-------------+
| first_name | last_name | rental_time |
+------------+-----------+-------------+
| JEFFERY    | PINSON    | 22:53:33    |
| ELMER      | NOE       | 22:55:13    |
| MINNIE     | ROMERO    | 23:00:34    |
| MIRIAM     | MCKINNEY  | 23:07:08    |
| DANIEL     | CABRAL    | 23:09:38    |
| TERRANCE   | ROUSH     | 23:12:46    |
| JOYCE      | EDWARDS   | 23:16:26    |
| GWENDOLYN  | MAY       | 23:16:27    |
| CATHERINE  | CAMPBELL  | 23:17:03    |
| MATTHEW    | MAHAN     | 23:25:58    |
| HERMAN     | DEVORE    | 23:35:09    |
| AMBER      | DIXON     | 23:42:56    |
| TERRENCE   | GUNDERSON | 23:47:35    |
| SONIA      | GREGORY   | 23:50:11    |
| CHARLES    | KOWALSKI  | 23:54:34    |
| JEANETTE   | GREENE    | 23:54:46    |
+------------+-----------+-------------+
16 rows in set (0.01 sec)
```

Önceki sorgu, hem müşteri tablosundan (first_name, last_name) hem de kiralama tablosundan (kiralık_tarihi) verilerini görüntüler, böylece her iki tablo da from yan tümcesine dahil edilir. İki tabloyu birbirine bağlama mekanizması (join), hem müşteri hem de kiralama tablolarında depolanan customer_id sütunudur. Böylece, veritabanı sunucusuna, müşterinin kiralama tablosundaki tüm kiralamalarını bulmak için müşteri tablosundaki customer_id sütununun değerini kullanması talimatı verilir. İki tablo için birleştirme koşulları from yan tümcesinin ON alt ifadesi ile gerçekleştirilir. birleştirme koşulu on customer.customer_id = rent.customer_id'dir. where yan tümcesi birleştirmenin bir parçası değildir ve kiralama tablosunda 16.000'den fazla satır olduğundan, yalnızca sonuç kümesini oldukça küçük tutmak için dahil edilmiştir.

Tablo Takma Adlarını Tanımlama

Tek bir sorguda birden çok tablo birleştirildiğinde, select, where, group by, having, ve order by yan cümleleri kullanıldığındahangi tabloya başvurduğunuzu belirlemenin bir yoluna ihtiyacınız vardır. from yan tümcesi dışında bir tabloya başvururken iki seçeneğiniz vardır:

- employe.emp_id gibi tüm tablo adını kullanın.
- Her tabloya bir takma ad atayın ve verdiğiniz adı sorgu boyunca kullanın.

```
SELECT c.first_name, c.last_name,
  time(r.rental_date) rental_time
FROM customer c
  INNER JOIN rental r
  ON c.customer_id = r.customer_id
WHERE date(r.rental_date) = '2005-06-14';
```

From maddesine yakından bakarsanız, müşteri tablosuna c takma adı ve kiralama tablosuna r takma adı atandığını görürsünüz. Bu takma adlar daha sonra birleştirme koşulu tanımlanırken **ON** yan tümcesinde ve sonuç kümesine dahil edilecek sütunları belirtirken **select** yan tümcesinde kullanılır. Takma ad kullanmanın karışıklığa neden olmadan daha kompakt bir ifade oluşturduğunu kabul edeceğinizi umulur (takma ad seçimleriniz makul olduğu sürece). Ek olarak, daha önce sütun takma adları için gösterilene benzer şekilde, **as** anahtar sözcüğünü tablo takma adlarınızla birlikte kullanabilirsiniz:

```
SELECT c.first_name, c.last_name,
  time(r.rental_date) rental_time
FROM customer AS c
  INNER JOIN rental AS r
  ON c.customer_id = r.customer_id
WHERE date(r.rental_date) = '2005-06-14';
```