Veri Oluşturma, Manipülasyon ve Dönüştürme

String özel karakterler

Bu bölümdeki bazı örnekler için aşağıdaki tablo oluşturulmuştur:

```
CREATE TABLE string_tbl
 (char_fld CHAR(30),
  vchar_fld VARCHAR(30),
  text_fld TEXT
 );
```

Bir karakter sütununu doldurmanın en basit yolu, aşağıdaki örneklerde olduğu gibi bir dizeyi tırnak içine almaktır:

```
mysql> INSERT INTO string_tbl (char_fld, vchar_fld, text_fld)
    -> VALUES ('This is char data',
    ->   'This is varchar data',
    ->   'This is text data');
Query OK, 1 row affected (0.00 sec)
```

Dizeler tek tırnakla sınırlandırıldığından, tek tırnak veya kesme işareti içeren dizeler konusunda dikkatli olmanız gerekir. Örneğin, sunucu sözcükteki kesme işaretinin dizenin sonunu göstermediğini düşüneceğinden aşağıdaki dizeyi ekleyemeyeceksiniz:

```
UPDATE string_tbl
SET text_fld = 'This string doesn't work';
```

Sunucunun **doesn’t** sözcüğündeki kesme işaretini yok saymasını sağlamak için, dizeye bir çıkış eklemeniz gerekir, böylece sunucu kesme işaretine dizedeki diğer herhangi bir karakter gibi davranır. Her üç sunucu da (Oracle, MySQL, SQL Server), aşağıdaki gibi, doğrudan önce başka bir tek alıntı ekleyerek tek bir alıntıdan kaçmanıza izin verir:

```
mysql> UPDATE string_tbl
    -> SET text_fld = 'This string didn''t work, but it does now';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Oracle Database ve MySQL kullanıcıları, aşağıdaki gibi, hemen önüne bir ters eğik çizgi ekleyerek tek bir alıntıdan kaçmayı da seçebilirler:

```
UPDATE string_tbl SET text_fld =
  'This string didn\'t work, but it does now'
```

Bir ekran veya rapor alanında kullanmak için bir dize alırsanız, gömülü alıntıları işlemek için özel bir şey yapmanız gerekmez:

```
mysql> SELECT text_fld
    -> FROM string_tbl;
+----------------------------------------+
| text_fld                               |
+----------------------------------------+
| This string didn't work, but it does now |
+----------------------------------------+
1 row in set (0.00 sec)
```

Ancak, başka bir programın okuyacağı bir dosyaya eklemek için dizeyi alıyorsanız, kaçışı alınan dizenin bir parçası olarak dahil etmek isteyebilirsiniz. MySQL kullanıyorsanız, tüm dizenin etrafına tırnak işaretleri yerleştiren ve dize içindeki herhangi bir tek tırnak/kesme işaretine çıkışlar ekleyen yerleşik **quote()** metodunu kullanabilirsiniz.

```
mysql> SELECT quote(text_fld)
    -> FROM string_tbl;
+-------------------------------------------+
| QUOTE(text_fld)                           |
+-------------------------------------------+
| 'This string didn\'t work, but it does now' |
+-------------------------------------------+
1 row in set (0.04 sec)
```

Uygulamanızın kapsamı çok ulusluysa, kendinizi klavyenizde görünmeyen karakterler içeren dizelerle çalışırken bulabilirsiniz. Örneğin, Fransızca ve Almanca dilleriyle çalışırken é ve ö gibi aksanlı karakterler eklemeniz gerekebilir. SQL Server ve MySQL sunucuları, ASCII karakter kümesindeki 255 karakterden herhangi birinden dizeler oluşturabilmeniz için yerleşik char() metodunu içerir (Oracle Veritabanı kullanıcıları chr() metodunu kullanabilir). Göstermek için, sonraki örnek, yazılan bir dizeyi ve tek tek karakterler aracılığıyla oluşturulmuş eşdeğerini alır:

```
mysql> SELECT 'abcdefg', CHAR(97,98,99,100,101,102,103);
+---------+-----------------------------------+
| abcdefg | CHAR(97,98,99,100,101,102,103) |
+---------+-----------------------------------+
| abcdefg | abcdefg                           |
+---------+-----------------------------------+
1 row in set (0.01 sec)
```

ASCII karakter kümesindeki 97. karakter a harfidir. Önceki örnekte gösterilen karakterler özel olmasa da, aşağıdaki örnekler para birimi simgeleri gibi diğer özel karakterlerle birlikte aksanlı karakterlerin konumunu gösterir:

```
mysql> SELECT CHAR(128,129,130,131,132,133,134,135,136,137);
+-----------------------------------------------+
| CHAR(128,129,130,131,132,133,134,135,136,137) |
+-----------------------------------------------+
| Çüéâäàåçêë                                    |
+-----------------------------------------------+
1 row in set (0.01 sec)
```

Bu bölümde örnekler için utf8mb4 karakter setini kullanılmıştır. Oturumunuz farklı bir karakter kümesi için yapılandırılmışsa, burada gösterilenden farklı bir karakter kümesi göreceksiniz. Aynı kavramlar geçerlidir, ancak belirli karakterleri bulmak için karakter setinizin düzenine aşina olmanız gerekir.

Dizeleri karakter karakter oluşturmak, özellikle dizedeki karakterlerden yalnızca birkaçı vurguluysa, oldukça sıkıcı olabilir. Neyse ki, bazılarını yazabileceğiniz, diğerlerini **char()** metodu aracılığıyla oluşturabileceğiniz bireysel dizileri birleştirmek için **concat()** metodunu kullanabilirsiniz. Örneğin, aşağıda **concat() ve char()** metolarınıi kullanarak **danke schön** ifadesinin nasıl oluşturulacağı gösterilmektedir:

```
mysql> SELECT CONCAT('danke sch', CHAR(148), 'n');
+-------------------------------------+
| CONCAT('danke sch', CHAR(148), 'n') |
+-------------------------------------+
| danke schön                         |
+-------------------------------------+
1 row in set (0.00 sec)
```

Bir karakteriniz varsa ve onun ASCII eşdeğerini bulmanız gerekiyorsa, dizede en soldaki karakteri alan ve bir sayı döndüren ascii() metodunu kullanabilirsiniz:

```
mysql> SELECT ASCII('ö');
+------------+
| ASCII('ö') |
+------------+
|        148 |
+------------+
1 row in set (0.00 sec)
```

String Manipülasyonları

Her veritabanı sunucusu, dizeleri işlemek için birçok yerleşik metot içerir. Bu bölümde iki tür dizi metodu incelenir: sayı döndürenler ve diziler döndürenler. Ancak başlamadan önce string_tbl tablosundaki verileri aşağıdaki gibi sıfırlanır:

```
mysql> DELETE FROM string_tbl;
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO string_tbl (char_fld, vchar_fld, text_fld)
    -> VALUES ('This string is 28 characters',
    ->   'This string is 28 characters',
    ->   'This string is 28 characters');
Query OK, 1 row affected (0.00 sec)
```

Sayı döndüren dize metotlarından en yaygın kullanılanlarından biri, dizedeki karakter sayısını döndüren **len()** metodudur (SQL Server kullanıcılarının len() metodunu kullanması gerekir). Aşağıdaki sorgu, len() metodunu string_tbl tablosundaki her sütuna uygular:

```
mysql> SELECT LENGTH(char_fld) char_length,
    ->   LENGTH(vchar_fld) varchar_length,
    ->   LENGTH(text_fld) text_length
    -> FROM string_tbl;
+-------------+----------------+-------------+
| char_length | varchar_length | text_length |
+-------------+----------------+-------------+
|          28 |             28 |          28 |
+-------------+----------------+-------------+
1 row in set (0.00 sec)
```

Bir dizenin uzunluğunu bulmanın yanı sıra, bir dize içindeki bir alt dizenin konumunu da bulmak isteyebilirsiniz. Örneğin, **vchar_fld** sütununda **'characters'** dizesinin göründüğü konumu bulmak istiyorsanız, aşağıda gösterildiği gibi **position()** metodunu kullanabilirsiniz:

```
mysql> SELECT POSITION('characters' IN vchar_fld)
    -> FROM string_tbl;
+-----------------------------------+
| POSITION('characters' IN vchar_fld) |
+-----------------------------------+
|                                19 |
+-----------------------------------+
1 row in set (0.12 sec)
```

Alt dize bulunamazsa, position() metodu 0 döndürür.

Aramanızı hedef dizginizin ilk karakterinden başka bir şeyde başlatmak istiyorsanız, isteğe bağlı bir üçüncü parametreye izin vermesi dışında **position()** metoduna benzer olan **locate()** metodunu kullanmanız gerekir.

```
mysql> SELECT LOCATE('is', vchar_fld, 5)
    -> FROM string_tbl;
+----------------------------+
| LOCATE('is', vchar_fld, 5) |
+----------------------------+
|                         13 |
+----------------------------+
1 row in set (0.02 sec)
```

Dizeleri bağımsız değişken olarak alan ve sayıları döndüren başka bir metot, dize karşılaştırma metodu **strcmp()**'dir. Yalnızca MySQL tarafından uygulanan ve Oracle Database veya SQL Server'da olmayan **strcmp(),** argüman olarak iki dize alır ve aşağıdakilerden birini döndürür:

- −1 sıralamada ilk dize ikinci dizeden önce geliyorsa
- •0 dizeler aynıysa
- 1 sıralama düzeninde ilk dize ikinci dizeden sonra geliyorsa

İşlevin nasıl çalıştığını göstermek için önce bir sorgu kullanarak beş dizenin sıralama düzenini ve ardından **strcmp()** kullanarak dizelerin birbirleriyle nasıl karşılaştırıldığını göstermek için **string_tbl** tablosuna beş karakter dizisi eklenir:

```
mysql> DELETE FROM string_tbl;
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO string_tbl(vchar_fld)
    -> VALUES ('abcd'),
    ->        ('xyz'),
    ->        ('QRSTUV'),
    ->        ('qrstuv'),
    ->        ('12345');
Query OK, 5 rows affected (0.05 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT vchar_fld
    -> FROM string_tbl
    -> ORDER BY vchar_fld;
+-----------+
| vchar_fld |
+-----------+
| 12345     |
| abcd      |
| QRSTUV    |
| qrstuv    |
| xyz       |
+-----------+
5 rows in set (0.00 sec)
```

```
mysql> SELECT STRCMP('12345','12345') 12345_12345,
    ->   STRCMP('abcd','xyz') abcd_xyz,
    ->   STRCMP('abcd','QRSTUV') abcd_QRSTUV,
    ->   STRCMP('qrstuv','QRSTUV') qrstuv_QRSTUV,
    ->   STRCMP('12345','xyz') 12345_xyz,
    ->   STRCMP('xyz','qrstuv') xyz_qrstuv;
+-------------+----------+-------------+---------------+-----------+------------+
| 12345_12345 | abcd_xyz | abcd_QRSTUV | qrstuv_QRSTUV | 12345_xyz | xyz_qrstuv |
+-------------+----------+-------------+---------------+-----------+------------+
|           0 |       -1 |          -1 |             0 |        -1 |          1 |
+-------------+----------+-------------+---------------+-----------+------------+
1 row in set (0.00 sec)
```

Bazı durumlarda, dizenin bir kısmını çıkararak veya dizeye ek metin ekleyerek mevcut dizeleri değiştirmeniz gerekecektir. Her veritabanı sunucusu, bu görevlere yardımcı olacak birden çok metot içerir. Başlamadan önce **string_tbl** tablosundaki verileri bir kez daha sıfırladım:

```
mysql> DELETE FROM string_tbl;
Query OK, 5 rows affected (0.00 sec)

mysql> INSERT INTO string_tbl (text_fld)
    -> VALUES ('This string was 29 characters');
Query OK, 1 row affected (0.01 sec)
```

Bölümün başlarında, aksanlı karakterler içeren sözcükler oluşturmaya yardımcı olmak için **concat()** metodu kullanımını gösterilmişti. **concat()** işlevi, saklanan bir dizeye ek karakterler eklemeniz gerektiğinde de dahil olmak üzere birçok başka durumda yararlıdır. Örneğin, aşağıdaki örnek, sonuna ek bir ifade ekleyerek **text_fld** sütununda depolanan dizeyi değiştirir:

```
mysql> UPDATE string_tbl
    -> SET text_fld = CONCAT(text_fld, ', but now it is longer');
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

The contents of the `text_fld` column are now as follows:

```
mysql> SELECT text_fld
    -> FROM string_tbl;
+-----------------------------------------------------+
| text_fld                                            |
+-----------------------------------------------------+
| This string was 29 characters, but now it is longer |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

Bu nedenle, bir dize döndüren tüm işlevler gibi, bir karakter sütununda depolanan verileri değiştirmek için concat() öğesini kullanabilirsiniz.

concat() metodunun başka bir yaygın kullanımı, tek tek veri parçalarından bir dize oluşturmaktır. Örneğin, aşağıdaki sorgu her müşteri için bir anlatı dizisi oluşturur:

```
mysql> SELECT concat(first_name, ' ', last_name,
    ->   ' has been a customer since ', date(create_date)) cust_narrative
    -> FROM customer:
+-----------------------------------------------------------+
| cust_narrative                                            |
+-----------------------------------------------------------+
| MARY SMITH has been a customer since 2006-02-14           |
| PATRICIA JOHNSON has been a customer since 2006-02-14     |
| LINDA WILLIAMS has been a customer since 2006-02-14       |
| BARBARA JONES has been a customer since 2006-02-14        |
| ELIZABETH BROWN has been a customer since 2006-02-14      |
| JENNIFER DAVIS has been a customer since 2006-02-14       |
| MARIA MILLER has been a customer since 2006-02-14         |
| SUSAN WILSON has been a customer since 2006-02-14         |
| MARGARET MOORE has been a customer since 2006-02-14       |
| DOROTHY TAYLOR has been a customer since 2006-02-14       |
...
| RENE MCALISTER has been a customer since 2006-02-14       |
| EDUARDO HIATT has been a customer since 2006-02-14        |
| TERRENCE GUNDERSON has been a customer since 2006-02-14   |
| ENRIQUE FORSYTHE has been a customer since 2006-02-14     |
| FREDDIE DUGGAN has been a customer since 2006-02-14       |
| WADE DELVALLE has been a customer since 2006-02-14        |
| AUSTIN CINTRON has been a customer since 2006-02-14       |
+-----------------------------------------------------------+
599 rows in set (0.00 sec)
```

concat() bir dizenin başına veya sonuna karakter eklemek için kullanışlı olsa da, bir dizenin ortasına karakter eklemeniz veya değiştirmeniz gerekebilir. MySQL, dört argüman alan **insert()** metodunu içerir: **(orijinal dize, başlayacağı konum, değiştirilecek karakter sayısı ve değiştirme dizesi)** Üçüncü argümanın değerine bağlı olarak, metot bir dizeye karakter eklemek veya karakterleri değiştirmek için kullanılabilir. Üçüncü bağımsız değişken için 0 değeriyle, değiştirme dizesi eklenir ve izleyen karakterler aşağıdaki gibi sağa itilir:

```
mysql> SELECT INSERT('goodbye world', 9, 0, 'cruel ') string;
+---------------------+
| string              |
+---------------------+
| goodbye cruel world |
+---------------------+
1 row in set (0.00 sec)
```

Bu örnekte, 9. pozisyondan başlayan tüm karakterler sağa itilir ve **'cruel'** dizesi eklenir. Üçüncü argüman sıfırdan büyükse, o zaman bu karakter sayısı, aşağıdaki gibi, değiştirme dizesi ile değiştirilir:

```
mysql> SELECT INSERT('goodbye world', 1, 7, 'hello') string;
+-------------+
| string      |
+-------------+
| hello world |
+-------------+
1 row in set (0.00 sec)
```

Bir dizgeye karakter eklemenin yanı sıra, bir dizgeden bir alt dizgi çıkarmanız gerekebilir. Bu amaçla, belirli bir konumdan başlayarak belirli sayıda karakteri ayıklayan **substring()** metodu vardır. Aşağıdaki örnek, dokuzuncu konumdan başlayan bir dizeden beş karakter çıkarır:

```
mysql> SELECT SUBSTRING('goodbye cruel world', 9, 5);
+---------------------------------------+
| SUBSTRING('goodbye cruel world', 9, 5) |
+---------------------------------------+
| cruel                                 |
+---------------------------------------+
1 row in set (0.00 sec)
```

Sayısal verilerle çalışmak

Sayısal veri üretimi oldukça basittir. Bir sayı yazabilir, onu başka bir sütundan alabilir veya bir hesaplama yoluyla oluşturabilirsiniz. Tüm olağan aritmetik operatörler (, -, *, /) hesaplama yapmak için kullanılabilir ve önceliği belirtmek için parantezler aşağıdaki gibi kullanılabilir:

```
mysql> SELECT (37 * 59) / (78 - (8 * 6));
+----------------------------+
| (37 * 59) / (78 - (8 * 6)) |
+----------------------------+
|                      72.77 |
+----------------------------+
1 row in set (0.00 sec)
```

Yerleşik sayısal metotların çoğu, bir sayının karekökünü belirlemek gibi belirli aritmetik amaçlar için kullanılır. Tablo tek bir sayısal bağımsız değişken alan ve bir sayı döndüren yaygın sayısal metotlardan bazılarını listeler.

Function name	Description
acos(*x*)	Calculates the arc cosine of *x*
asin(*x*)	Calculates the arc sine of *x*
atan(*x*)	Calculates the arc tangent of *x*
cos(*x*)	Calculates the cosine of *x*
cot(*x*)	Calculates the cotangent of *x*
exp(*x*)	Calculates e^x
ln(*x*)	Calculates the natural log of *x*
sin(*x*)	Calculates the sine of *x*
sqrt(*x*)	Calculates the square root of *x*
tan(*x*)	Calculates the tangent of *x*

Bir sayı başka bir sayıya bölündüğünde kalanı hesaplayan **modulo** operatörü, **mod()** metoduna örnek, 10'un 4'e bölümünden kalanı hesaplar:

```
mysql> SELECT MOD(10,4);
+-----------+
| MOD(10,4) |
+-----------+
|         2 |
+-----------+
1 row in set (0.02 sec)
```

mod() işlevi tipik olarak tamsayı argümanlarıyla kullanılırken, MySQL ile aşağıdaki gibi gerçek sayıları da kullanabilirsiniz:

```
mysql> SELECT MOD(22.75, 5);
+---------------+
| MOD(22.75, 5) |
+---------------+
|          2.75 |
+---------------+
1 row in set (0.02 sec)
```

İki sayısal argüman alan başka bir sayısal metot, aşağıdaki gibi ikinci bir sayının gücüne yükseltilmiş bir sayı döndüren pow() metodudur.

```
mysql> SELECT POW(2,8);
+----------+
| POW(2,8) |
+----------+
|      256 |
+----------+
1 row in set (0.03 sec)
```

Bu nedenle, pow(2,8), 2^8 belirtmenin MySQL eşdeğeridir. Bilgisayar belleği 2^x baytlık parçalar halinde ayrıldığından, pow() işlevi, belirli bir bellek miktarındaki tam bayt sayısını belirlemenin kullanışlı bir yolu olabilir. :

```
mysql> SELECT POW(2,10) kilobyte, POW(2,20) megabyte,
    ->   POW(2,30) gigabyte, POW(2,40) terabyte;
+----------+----------+------------+---------------+
| kilobyte | megabyte | gigabyte   | terabyte      |
+----------+----------+------------+---------------+
|     1024 |  1048576 | 1073741824 | 1099511627776 |
+----------+----------+------------+---------------+
1 row in set (0.00 sec)
```

Kayan noktalı sayılarla çalışırken, her zaman tam kesinliği ile bir sayı ile etkileşime girmek veya bir sayı görüntülemek istemeyebilirsiniz. Örneğin, parasal işlem verilerini altı ondalık basamağa kadar hassasiyetle saklayabilirsiniz, ancak görüntüleme amacıyla en yakın yüzde birliğe yuvarlamak isteyebilirsiniz. Kayan noktalı sayıların kesinliğini sınırlarken dört metot yararlıdır: ceil(), floor(), round() ve truncate().

```
mysql> SELECT CEIL(72.445), FLOOR(72.445);
+--------------+---------------+
| CEIL(72.445) | FLOOR(72.445) |
+--------------+---------------+
|           73 |            72 |
+--------------+---------------+
1 row in set (0.06 sec)

mysql> SELECT CEIL(72.000000001), FLOOR(72.999999999);
+--------------------+---------------------+
| CEIL(72.000000001) | FLOOR(72.999999999) |
+--------------------+---------------------+
|                 73 |                  72 |
+--------------------+---------------------+
1 row in set (0.00 sec)
```

İki tam sayı arasındaki orta noktadan yukarı veya aşağı yuvarlamak için round() kullanabilirsiniz:

```
mysql> SELECT ROUND(72.49999), ROUND(72.5), ROUND(72.50001);
+-----------------+-------------+-----------------+
| ROUND(72.49999) | ROUND(72.5) | ROUND(72.50001) |
+-----------------+-------------+-----------------+
|              72 |          73 |              73 |
+-----------------+-------------+-----------------+
1 row in set (0.00 sec)
```

Çoğu zaman, en yakın tam sayıya yuvarlamak yerine bir sayının ondalık kısmının en azından bir kısmını tutmak isteyeceksiniz; round() işlevi, ondalık basamağın sağındaki kaç basamağa yuvarlanacağını belirtmek için isteğe bağlı ikinci bir

bağımsız değişkene izin verir. Sonraki örnek, 72.0909 sayısını bir, iki ve üç ondalık basamağa yuvarlamak için ikinci bağımsız değişkeni nasıl kullanabileceğinizi gösterir:

```
mysql> SELECT ROUND(72.0909, 1), ROUND(72.0909, 2), ROUND(72.0909, 3);
+------------------+------------------+------------------+
| ROUND(72.0909, 1) | ROUND(72.0909, 2) | ROUND(72.0909, 3) |
+------------------+------------------+------------------+
|             72.1 |            72.09 |           72.091 |
+------------------+------------------+------------------+
1 row in set (0.00 sec)
```

Round() işlevi gibi, truncate() işlevi de ondalık sayının sağındaki basamak sayısını belirtmek için isteğe bağlı ikinci bir argümana izin verir, ancak truncate() istenmeyen rakamları yuvarlamadan atar. Sonraki örnek, 72.0909 sayısının bir, iki ve üç ondalık basamağa nasıl kesileceğini gösterir:

```
mysql> SELECT TRUNCATE(72.0909, 1), TRUNCATE(72.0909, 2),
    ->   TRUNCATE(72.0909, 3);
+----------------------+----------------------+----------------------+
| TRUNCATE(72.0909, 1) | TRUNCATE(72.0909, 2) | TRUNCATE(72.0909, 3) |
+----------------------+----------------------+----------------------+
|                 72.0 |                72.09 |               72.090 |
+----------------------+----------------------+----------------------+
1 row in set (0.00 sec)
```

Hem truncate() hem de round(), ikinci argüman için negatif bir değere de izin verir; bu, ondalık basamağın solundaki sayıların kesildiği veya yuvarlandığı anlamına gelir. Bu ilk başta garip bir şey gibi görünebilir, ancak geçerli uygulamalar var. Örneğin, sadece 10 adet olarak satın alınabilen bir ürünü satabilirsiniz. Bir müşteri 17 adet sipariş verecekse, müşterinin sipariş miktarını değiştirmek için aşağıdaki yöntemlerden birini seçebilirsiniz:

```
mysql> SELECT ROUND(17, -1), TRUNCATE(17, -1);
+---------------+-----------------+
| ROUND(17, -1) | TRUNCATE(17, -1) |
+---------------+-----------------+
|            20 |               10 |
+---------------+-----------------+
1 row in set (0.00 sec)
```

Negatif değerlere izin veren sayısal sütunlarla çalışıyorsanız birkaç sayısal işlev yararlı olabilir. Örneğin, hesap tablosundan aşağıdaki verileri kullanarak bir dizi banka hesabının mevcut durumunu gösteren bir rapor oluşturmanız istendiğini varsayalım:

```
+------------+--------------+---------+
| account_id | acct_type    | balance |
+------------+--------------+---------+
|        123 | MONEY MARKET |  785.22 |
|        456 | SAVINGS      |    0.00 |
|        789 | CHECKING     | -324.22 |
+------------+--------------+---------+

mysql> SELECT account_id, SIGN(balance), ABS(balance)
    -> FROM account;
+------------+--------------+--------------+
| account_id | SIGN(balance) | ABS(balance) |
+------------+--------------+--------------+
|        123 |             1 |       785.22 |
|        456 |             0 |         0.00 |
|        789 |            -1 |       324.22 |
+------------+--------------+--------------+
3 rows in set (0.00 sec)
```

Zamansal veriler ile çalışmak

Aşağıdaki yollardan herhangi biriyle geçici veriler oluşturabilirsiniz:

• Mevcut bir **date, datetime veya time** sütunundan veri kopyalama

• Bir **date, datetime veya time** döndüren yerleşik bir metot yürütme

• Sunucu tarafından değerlendirilecek geçici verilerin bir dize temsilini oluşturma

Son yöntemi kullanmak için, tarihleri biçimlendirmede kullanılan çeşitli bileşenleri anlamanız gerekir.

Component	Definition	Range
YYYY	Year, including century	1000 to 9999
MM	Month	01 (January) to 12 (December)
DD	Day	01 to 31
HH	Hour	00 to 23
HHH	Hours (elapsed)	−838 to 838
MI	Minute	00 to 59
SS	Second	00 to 59

Sunucunun **date, datetime veya time** olarak yorumlayabileceği bir dize oluşturmak için çeşitli bileşenleri Tabloda gösterilen sırayla bir araya getirmeniz gerekir.

Type	Default format
date	YYYY-MM-DD
datetime	YYYY-MM-DD HH:MI:SS
timestamp	YYYY-MM-DD HH:MI:SS
time	HHH:MI:SS

Böylece, bir **datetime** sütununu 15:30 ile doldurmak için. 17 Eylül 2019'da aşağıdaki dizeyi oluşturmanız gerekecek:

```
'2019-09-17 15:30:00'
```

Sunucu, örneğin bir **datetime** sütununu güncellerken veya bir **datetime** argümanı alan yerleşik bir metodu çağırırken bir **datetime** değeri bekliyorsa, gerekli tarih bileşenleriyle uygun şekilde biçimlendirilmiş bir dize sağlayabilirsiniz. Örneğin, bir film kiralamanın iade tarihini değiştirmek için kullanılan bir ifade:

```
UPDATE rental
SET return_date = '2019-09-17 15:30:00'
WHERE rental_id = 99999;
```

Sunucu, dize bir **datetime** sütununu doldurmak için kullanıldığından, **set** yan tümcesinde sağlanan dizenin bir **datetime** değeri olması gerektiğini belirler. Bu nedenle sunucu, dizeyi varsayılan **datetime** biçiminde bulunan altı bileşene (yıl, ay, gün, saat, dakika, saniye) ayrıştırarak sizin için dizeyi dönüştürmeye çalışacaktır.

Sunucu bir **datetime** değeri beklemiyorsa veya **datetime** varsayılan olmayan bir biçim kullanarak göstermek istiyorsanız, sunucuya dizeyi **datetime**'a dönüştürmesini söylemeniz gerekir. Örneğin, **cast()** metodu kullanarak bir **datetime** değeri döndüren basit bir sorgu:

```
mysql> SELECT CAST('2019-09-17 15:30:00' AS DATETIME);
+----------------------------------------+
| CAST('2019-09-17 15:30:00' AS DATETIME) |
+----------------------------------------+
| 2019-09-17 15:30:00                    |
+----------------------------------------+
1 row in set (0.00 sec)
```

Cast() fonksiyonunu ile örnek, **datetime** değerlerinin nasıl oluşturulacağını gösterirken, aynı mantık **date** ve **time** türleri için ayrı ayrı da geçerlidir. Aşağıdaki sorgu, bir **date** değeri ve bir **time** değeri oluşturmak için **cast()** işlevini kullanır:

```
mysql> SELECT CAST('2019-09-17' AS DATE) date_field,
    ->   CAST('108:17:57' AS TIME) time_field;
+------------+------------+
| date_field | time_field |
+------------+------------+
| 2019-09-17 | 108:17:57  |
+------------+------------+
1 row in set (0.00 sec)
```

Dizeler açık veya örtük olarak geçici değerlere dönüştürüldüğünde, tüm tarih bileşenlerini gerekli sırada sağlamanız gerekir. Bazı sunucular tarih formatı konusunda oldukça katı iken, MySQL sunucusu bileşenler arasında kullanılan ayırıcılar konusunda oldukça hoşgörülüdür. Örneğin MySQL, aşağıdaki dizelerin tümünü 17 Eylül 2019'da 3:30 PM'nin geçerli temsilleri olarak kabul edecektir.:

```
'2019-09-17 15:30:00'
'2019/09/17 15:30:00'
'2019,09,17,15,30,00'
'20190917153000'
```

Bir dizeden zamansal veri üretmeniz gerekiyorsa ve dize, cast() işlevini kullanmak için uygun biçimde değilse, tarih dizesi ile birlikte bir biçim dizesi sağlamanıza izin veren yerleşik bir metot kullanabilirsiniz. MySQL, bu amaç için str_to_date() işlevini içerir. Örneğin, bir dosyadan ‘September 17, 2019’ dizesini aldığınızı ve bir tarih sütununu güncellemek için kullanmanız gerektiğini varsayalım. Dize gerekli YYYY-AA-GG biçiminde olmadığından, dizeyi yeniden biçimlendirmek yerine str_to_date() metodunu kullanabilirsiniz:

```
UPDATE rental
SET return_date = STR_TO_DATE('September 17, 2019', '%M %d, %Y')
WHERE rental_id = 99999;
```

str_to_date() çağrısındaki ikinci argüman, bu durumda bir ay adı (%M), sayısal bir gün (%d) ve dört basamaklı bir sayısal yıl (%Y) ile tarih dizisinin biçimini tanımlar.). 30'dan fazla tanınan format bileşeni olmasına rağmen, Tablo en yaygın olarak kullanılan bir düzine kadar bileşeni tanımlar.

Format component	Description
%M	Month name (January to December)
%m	Month numeric (01 to 12)
%d	Day numeric (01 to 31)
%j	Day of year (001 to 366)
%W	Weekday name (Sunday to Saturday)
%Y	Year, four-digit numeric
%y	Year, two-digit numeric
%H	Hour (00 to 23)
%h	Hour (01 to 12)
%i	Minutes (00 to 59)
%s	Seconds (00 to 59)
%f	Microseconds (000000 to 999999)
%p	A.M. or P.M.

str_to_date() işlevi, biçim dizesinin içeriğine bağlı olarak bir **date, datetime veya time** değeri döndürür. Örneğin, biçim dizesi yalnızca %H, %i ve %s içeriyorsa, bir zaman değeri döndürülecektir.

Geçerli tarih/saati oluşturmaya çalışıyorsanız, aşağıdaki yerleşik işlevler sistem saatine erişeceği ve geçerli tarih ve/veya saati sizin için bir dize olarak döndüreceği için bir dize oluşturmanız gerekmez:

```
mysql> SELECT CURRENT_DATE(), CURRENT_TIME(), CURRENT_TIMESTAMP();
+----------------+----------------+---------------------+
| CURRENT_DATE() | CURRENT_TIME() | CURRENT_TIMESTAMP() |
+----------------+----------------+---------------------+
| 2019-06-05     | 16:54:36       | 2019-06-05 16:54:36 |
+----------------+----------------+---------------------+
1 row in set (0.12 sec)
```

Yerleşik zamansal metotların çoğu, bir tarihi argüman olarak alır ve başka bir tarih döndürür. MySQL'in date_add() metodu, örneğin, başka bir tarih oluşturmak için belirli bir tarihe herhangi bir tür aralığı (örneğin, günler, aylar, yıllar) eklemenize izin verir. Geçerli tarihe nasıl beş gün ekleneceğini gösteren bir örnek:

```
mysql> SELECT DATE_ADD(CURRENT_DATE(), INTERVAL 5 DAY);
+-----------------------------------------+
| DATE_ADD(CURRENT_DATE(), INTERVAL 5 DAY) |
+-----------------------------------------+
| 2019-06-10                              |
+-----------------------------------------+
1 row in set (0.06 sec)
```

İkinci argüman üç öğeden oluşur: **interval** anahtar sözcüğü, **istenen miktar ve aralığın türü.** Tablo yaygın olarak kullanılan bazı aralık türlerini göstermektedir.

Interval name	Description
second	Number of seconds
minute	Number of minutes
hour	Number of hours
day	Number of days
month	Number of months
year	Number of years
minute_second	Number of minutes and seconds, separated by ":"
hour_second	Number of hours, minutes, and seconds, separated by ":"
year_month	Number of years and months, separated by "-"

Tabloda listelenen ilk altı tür oldukça basit olsa da, son üç tür birden fazla öğeye sahip oldukları için biraz daha fazla açıklama gerektirir. Örneğin, bir filmin başlangıçta belirtilen süreden 3 saat, 27 dakika ve 11 saniye sonra iade edildiği söylenirse, sorunu şu şekilde düzeltebilirsiniz:

```
UPDATE rental
SET return_date = DATE_ADD(return_date, INTERVAL '3:27:11' HOUR_SECOND)
WHERE rental_id = 99999;
```

Bu örnekte, date_add() işlevi, return_date sütunundaki değeri alır ve buna 3 saat, 27 dakika ve 11 saniye ekler. Ardından, return_date sütununu değiştirmek için elde edilen değeri kullanır.

Veya İK'da çalışıyorsanız ve 4789 numaralı çalışanın gerçekte olduğundan daha yaşlı olduğunu iddia ettiyseniz, doğum tarihine aşağıdaki gibi 9 yıl 11 ay ekleyebilirsiniz:

```
UPDATE employee
SET birth_date = DATE_ADD(birth_date, INTERVAL '9-11' YEAR_MONTH)
WHERE emp_id = 4789;
```

Bir tarihe bir aralık eklemek istediğiniz bazı durumlar vardır ve nereye varmak istediğinizi bilirsiniz, ancak oraya varmanın kaç gün sürdüğünü bilmezsiniz. Örneğin, bir banka müşterisinin online bankacılık sistemine giriş yaptığını ve ay sonu için bir transfer planladığını varsayalım. İçinde bulunulan ayı hesaplayan ve ardından o aydaki gün sayısını arayan bir kod yazmak yerine, işi sizin için yapan last_day() metodunu çağırabilirsiniz. Müşteri 17 Eylül 2019 tarihinde transfer isterse, Eylül ayının son gününü aşağıdaki adresten bulabilirsiniz:

```
mysql> SELECT LAST_DAY('2019-09-17');
+-----------------------+
| LAST_DAY('2019-09-17') |
+-----------------------+
| 2019-09-30             |
+-----------------------+
1 row in set (0.10 sec)
```

Bir tarih veya tarih saat değeri sağlasanız da, **last_day()** işlevi her zaman bir tarih döndürür. Bu işlev çok büyük bir zaman kazandırıcı gibi görünmese de, Şubat ayının son gününü bulmaya çalışıyorsanız ve mevcut yılın artık yıl olup olmadığını anlamanız gerekiyorsa, temel mantık yanıltıcı olabilir.

Dize değerleri döndüren zamansal işlevlerin çoğu, bir tarih veya saatin bir bölümünü çıkarmak için kullanılır. Örneğin MySQL, belirli bir tarihin haftanın hangi gününe denk geldiğini belirlemek için aşağıdaki gibi **dayname()** metodunu içerir:

```
mysql> SELECT DAYNAME('2019-09-18');
+-----------------------+
| DAYNAME('2019-09-18') |
+-----------------------+
| Wednesday             |
+-----------------------+
1 row in set (0.00 sec)
```

Tarih değerlerinden bilgi çıkarmak için bu tür birçok işlev MySQL'e dahildir, ancak bir metdoun birkaç varyasyonunu hatırlamak bir düzine farklı işlevi hatırlamaktan daha kolay olduğundan, bunun yerine **extract ()** metodunu kullanmak daha kolaydır.

Extract() metodu, tarihin hangi öğesinin ilginizi çektiğini tanımlamak için **date_add()** işleviyle aynı aralık türlerini kullanır Örneğin, bir tarih saat değerinin yalnızca yıl kısmını çıkarmak istiyorsanız aşağıdakileri yapabilirsiniz:

```
mysql> SELECT EXTRACT(YEAR FROM '2019-09-18 22:19:05');
+-----------------------------------------+
| EXTRACT(YEAR FROM '2019-09-18 22:19:05') |
+-----------------------------------------+
|                                    2019 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

Tarihlerle çalışırken bir diğer yaygın aktivite, iki tarih değeri almak ve iki tarih arasındaki aralıkların (günler, haftalar, yıllar) sayısını belirlemektir. Bu amaçla MySQL, iki tarih arasındaki tam gün sayısını döndüren **datediff()** metodunu içerir.

```
mysql> SELECT DATEDIFF('2019-09-03', '2019-06-21');
+--------------------------------------+
| DATEDIFF('2019-09-03', '2019-06-21') |
+--------------------------------------+
|                                   74 |
+--------------------------------------+
1 row in set (0.00 sec)
```

datediff() işlevi, argümanlarında günün saatini yok sayar. İlk tarih için gece yarısına kadar bir saniye ve ikinci tarih için gece yarısından sonra bir saniye olarak ayarlayarak günün bir saatini eklesem bile, bu zamanların hesaplama üzerinde hiçbir etkisi olmayacaktır:

```
mysql> SELECT DATEDIFF('2019-09-03 23:59:59', '2019-06-21 00:00:01');
+-------------------------------------------------------+
| DATEDIFF('2019-09-03 23:59:59', '2019-06-21 00:00:01') |
+-------------------------------------------------------+
|                                                    74 |
+-------------------------------------------------------+
1 row in set (0.00 sec)
```

Argümanları değiştirir ve önce önceki tarihi alırsam, datediff() aşağıdaki gibi negatif bir sayı döndürür:

```
mysql> SELECT DATEDIFF('2019-06-21', '2019-09-03');
+--------------------------------------+
| DATEDIFF('2019-06-21', '2019-09-03') |
+--------------------------------------+
|                                  -74 |
+--------------------------------------+
1 row in set (0.00 sec)
```

Değerleri birbirlerine dönüştürmek için **cast()'ı** kullanabilirsiniz.

```
mysql> SELECT CAST('1456328' AS SIGNED INTEGER);
+-----------------------------------+
| CAST('1456328' AS SIGNED INTEGER) |
+-----------------------------------+
|                           1456328 |
+-----------------------------------+
1 row in set (0.01 sec)
```

Bir dizeyi sayıya dönüştürürken, cast() işlevi tüm dizeyi soldan sağa dönüştürmeye çalışır; dizede sayısal olmayan herhangi bir karakter bulunursa, dönüştürme hatasız olarak durur. Aşağıdaki örneği göz önünde bulundurun:

```
mysql> SELECT CAST('999ABC111' AS UNSIGNED INTEGER);
+--------------------------------------+
| CAST('999ABC111' AS UNSIGNED INTEGER) |
+--------------------------------------+
|                                  999 |
+--------------------------------------+
1 row in set, 1 warning (0.08 sec)

mysql> show warnings;
+---------+------+---------------------------------------------+
| Level   | Code | Message                                     |
+---------+------+---------------------------------------------+
| Warning | 1292 | Truncated incorrect INTEGER value: '999ABC111' |
+---------+------+---------------------------------------------+
1 row in set (0.07 sec)
```

Bu durumda, dizenin ilk üç basamağı dönüştürülürken, dizenin geri kalanı atılır ve sonuç olarak 999 değeri elde edilir. Ancak sunucu, tüm dizenin değişmediğini size bildirmek için bir uyarı verdi.