insert, update, delete

Veri Ekleme

insert ifadesinin üç ana bileşen vardır:
- Verileri eklemek için tablonun adı
- Tablodaki doldurulacak sütunların isimleri
- Sütunları doldurmak için gerekli olan değerler

Tablodaki her sütun için veri sağlamanız gerekmez (tablodaki tüm sütunlar null değil olarak tanımlanmadıkça). Bazı durumlarda, ilk ekleme ifadesine dahil olmayan bu sütunlara daha sonra bir güncelleştirme ifadesi üzerinden bir değer verilecektir.

Sayısal anahtar verileri oluşturma

Person tablosuna veri eklemeden önce, sayısal birincil anahtarlar için değerlerin nasıl üretildiğini tartışmak faydalı olacaktır. Rastgele bir sayı seçmekten başka, birkaç seçeneğiniz var:
- Şu anda tablodaki en büyük değere bakın ve bir tane ekleyin.
- Veritabanı sunucusunun sizin için değeri sağlamasına izin verin.

İlk seçenek geçerli görünse de, iki kullanıcı aynı anda tabloya bakabileceği ve birincil anahtar için aynı değeri üretebileceği için çok kullanıcılı bir ortamda sorun oluşacaktır. Bunun yerine, bugün tüm veri tabanı sunucuları, sayısal anahtarlar oluşturmak için güvenli ve sağlam bir yöntem sunmaktadır. Oracle Veritabanı gibi bazı sunucularda ayrı bir şema nesnesi kullanılır (sıra olarak adlandırılır); ancak MySQL, birincil anahtar sütununuz için otomatik artış özelliğini açmanız yeterlidir. Normalde, bunu tablo oluştururken yapmalısınız, ancak şimdi yapmak, başka bir SQL şema deyimi öğrenme fırsatı sağlar, tabloyu değiştirmek için:

```
ALTER TABLE person MODIFY person_id SMALLINT UNSIGNED AUTO_INCREMENT;
```

Bu ifadeleri veritabanınızda çalıştırıyorsanız, önce favori_yemek tablosundaki yabancı anahtar kısıtlamasını devre dışı bırakmanız ve ardından işiniz bittiğinde kısıtlamaları yeniden etkinleştirmeniz gerekir. İfadelerin ilerlemesi şu şekilde olacaktır:

```
set foreign_key_checks=0;
ALTER TABLE person
    MODIFY person_id SMALLINT UNSIGNED AUTO_INCREMENT;
set foreign_key_checks=1;
```

Bu ifade, esas olarak kişi tablosundaki person_id sütununu yeniden tanımlar. Tablo detayına bakarsanız, person_id için "Extra" sütununun altında listelenen otomatik artış özelliğini göreceksiniz:

```
mysql> DESC person;
+-------------+-----------------------+------+-----+---------+----------------+
| Field       | Type                  | Null | Key | Default | Extra          |
+-------------+-----------------------+------+-----+---------+----------------+
| person_id   | smallint(5) unsigned  | NO   | PRI | NULL    | auto_increment |
| .           |                       |      |     |         |                |
| .           |                       |      |     |         |                |
| .           |                       |      |     |         |                |
```

Kişi tablosuna veri eklediğinizde, yalnızca person_id sütunu için boş bir değer sağlarsınız ve MySQL sütunu bir sonraki uygun sayı ile doldurur (varsayılan olarak MySQL, otomatik artış sütunları için 1'den başlar).

Insert

Aşağıdaki ifade William Turner için kişi tablosunda bir satır oluşturur:

```
mysql> INSERT INTO person
    ->   (person_id, fname, lname, eye_color, birth_date)
    -> VALUES (null, 'William','Turner', 'BR', '1972-05-27');
Query OK, 1 row affected (0.22 sec)
```

Geri bildirim (“Query OK, 1 row affected”) size ifade söz diziminizin uygun olduğunu ve veritabanına bir satırın eklendiğini (çünkü bu bir ekleme ifadesi olduğundan) söyler. Bir select ifadesi ile tabloya yeni eklenen verilere bakabilirsiniz:

```
mysql> SELECT person_id, fname, lname, birth_date
    -> FROM person;
+-----------+---------+--------+------------+
| person_id | fname   | lname  | birth_date |
+-----------+---------+--------+------------+
|         1 | William | Turner | 1972-05-27 |
+-----------+---------+--------+------------+
1 row in set (0.06 sec)
```

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

Gördüğünüz gibi, MySQL sunucusu birincil anahtar için 1 değeri üretti. Kişi tablosunda sadece tek bir satır olduğu için hangi satırla ilgilendiğimi belirtmeyi ihmal ettik ve tablodaki tüm satırları aldık. Ancak tabloda birden fazla satır varsa, yalnızca person_id sütunu için 1 değerine sahip satır için veri almak istediğimi zi belirtmek için bir where yan tümcesi ekleyebiliriz:

```
mysql> SELECT person_id, fname, lname, birth_date
    -> FROM person
    -> WHERE person_id = 1;
+-----------+---------+--------+------------+
| person_id | fname   | lname  | birth_date |
+-----------+---------+--------+------------+
|         1 | William | Turner | 1972-05-27 |
+-----------+---------+--------+------------+
1 row in set (0.00 sec)
```

Bu sorgu belirli bir birincil anahtar değerini belirtirken, lname sütunu için 'Turner' değerine sahip tüm satırları bulan aşağıdaki sorguda gösterildiği gibi, satırları aramak için tablodaki herhangi bir sütunu kullanabilirsiniz:

```
mysql> SELECT person_id, fname, lname, birth_date
    -> FROM person
    -> WHERE lname = 'Turner';
+-----------+---------+--------+------------+
| person_id | fname   | lname  | birth_date |
+-----------+---------+--------+------------+
|         1 | William | Turner | 1972-05-27 |
+-----------+---------+--------+------------+
1 row in set (0.00 sec)
```

• Adres sütunlarının hiçbiri için değerler sağlanmadı. çünkü bu sütunlar için boş değerlere izin verilir.
• Doğum_tarihi sütunu için sağlanan değer aslında bir dizedir. Tarih ve zaman için gerekli biçimi eşleştirdiğiniz sürece, MySQL dizeyi sizin için bir tarihe dönüştürür.
• Sütun adları ve sağlanan değerler sayı ve tür olarak uyumlu olmalıdır. Yedi sütunu adlandırır ve yalnızca altı değer sağlarsanız veya ilgili sütun için uygun veri türüne dönüştürülemeyen değerler verirseniz, bir hata alırsınız.

William Turner için ayrıca en sevdiği üç yiyeceği saklamak için:

```
mysql> INSERT INTO favorite_food (person_id, food)
    -> VALUES (1, 'pizza');
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO favorite_food (person_id, food)
    -> VALUES (1, 'cookies');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO favorite_food (person_id, food)
    -> VALUES (1, 'nachos');
Query OK, 1 row affected (0.01 sec)
```

İşte William'ın en sevdiği yiyecekleri yan tümceyi kullanarak alfabetik sırayla alan bir sorgu:

```
mysql> SELECT food
    -> FROM favorite_food
    -> WHERE person_id = 1
    -> ORDER BY food;
+---------+
| food    |
+---------+
| cookies |
| nachos  |
| pizza   |
+---------+
3 rows in set (0.02 sec)
```

Yan tümceye göre sıralama, sunucuya sorgu tarafından döndürülen verilerin nasıl sıralanacağını söyler. Order by cümlesi olmadan, tablodaki verilerin belirli bir sırayla alınacağının garantisi yoktur.
William'ın yalnız kalmaması için Susan Smith'i kişi tablosuna eklemek için başka bir ekleme ifadesi çalıştırabilirsiniz:

```
mysql> INSERT INTO person
    -> (person_id, fname, lname, eye_color, birth_date,
    -> street, city, state, country, postal_code)
    -> VALUES (null, 'Susan','Smith', 'BL', '1975-11-02',
    ->  '23 Maple St.', 'Arlington', 'VA', 'USA', '20220');
 Query OK, 1 row affected (0.01 sec)
```

Susan adresini verecek kadar kibar olduğundan, William'ın verilerinin eklendiği zamana göre beş sütun daha ekledik. Tabloyu tekrar sorgularsanız, Susan'ın satırına birincil anahtar değeri için 2 atandığını göreceksiniz:

```
mysql> SELECT person_id, fname, lname, birth_date
    -> FROM person;
+-----------+---------+--------+------------+
| person_id | fname   | lname  | birth_date |
+-----------+---------+--------+------------+
|         1 | William | Turner | 1972-05-27 |
|         2 | Susan   | Smith  | 1975-11-02 |
+-----------+---------+--------+------------+
2 rows in set (0.00 sec)
```

Veri Güncelleme

William Turner için veriler ilk olarak tabloya eklendiğinde, adres sütunları veri ekleme ifadesine dahil edilmedi. Sonraki ifade, bu sütunların daha sonra bir güncelleme ifadesi aracılığıyla nasıl doldurulabileceğini gösterir:

```
mysql> UPDATE person
    -> SET street = '1225 Tremont St.',
    ->   city = 'Boston',
    ->   state = 'MA',
    ->   country = 'USA',
    ->   postal_code = '02138'
    -> WHERE person_id = 1;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Sunucu iki satırlık bir mesajla yanıt verdi: "Rows matched: 1" öğesi, **where** yan tümcesindeki koşulun tablodaki tek bir satırla eşleştiğini ve "Changed: 1" öğesi size, tablodaki tek bir satırın değiştirildiğini söyler. **where** yan tümcesi William'ın satırının birincil anahtarını belirttiğinden, tam olarak olmasını beklediğiniz şey budur. Where yan tümcenizdeki koşullara bağlı olarak, tek bir deyim kullanarak birden fazla satırı değiştirmek de mümkündür. Örneğin, **where** yan tümceniz aşağıdaki gibi görünürse ne olacağını düşünün:

```
WHERE person_id < 10
```

Hem William hem de Susan'ın person_id değeri 10'dan küçük olduğundan, her ikisinin de satırı değiştirilir. where yan tümcesini dikkatli kullanmazsanız, update ifadeniz tablodaki her satırı değiştirecektir.

Veri Silme

Birincil anahtar, ilgilenilen satırı izole etmek için kullanılır, bu nedenle tablodan tek bir satır silinir. update deyimi gibi, where yan tümcenizdeki koşullara bağlı olarak birden fazla satır silinebilir ve where yan tümcesi atlanırsa tüm satırlar silinir.

```
mysql> DELETE FROM person
    -> WHERE person_id = 2;
Query OK, 1 row affected (0.01 sec)
```

Hatalar

Tablo tanımları, birincil anahtar kısıtlamalarının oluşturulmasını içerdiğinden, MySQL, tablolara yinelenen anahtar değerlerin eklenmemesini sağlayacaktır. Sonraki ifade, person_id sütununun otomatik artış özelliğini atlamaya ve person_id 1 olan kişi tablosunda başka bir satır oluşturmaya çalışır:

```
mysql> INSERT INTO person
    ->  (person_id, fname, lname, eye_color, birth_date)
    -> VALUES (1, 'Charles','Fulton', 'GR', '1968-01-15');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

Person_id sütunu için farklı değerlere sahip oldukları sürece, sizi (en azından mevcut şema nesneleri ile) aynı adlara, adreslere, doğum tarihlerine vb. sahip iki satır oluşturmaktan alıkoyan hiçbir şey yoktur.

Favorite_food tablosunun tablo tanımı, person_id sütununda bir yabancı anahtar kısıtlamasının oluşturulmasını içerir. Bu kısıtlama, favori_yemek tablosuna girilen tüm person_id değerlerinin kişi tablosunda var olmasını sağlar. Bu kısıtlamayı ihlal eden bir satır oluşturmaya çalışırsanız şunlar olur:

```
mysql> INSERT INTO favorite_food (person_id, food)
    -> VALUES (999, 'lasagna');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint
fails ('sakila'.'favorite_food', CONSTRAINT 'fk_fav_food_person_id' FOREIGN KEY
('person_id') REFERENCES 'person' ('person_id'))
```

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

Bu durumda, Favorite_food tablosu alt öğe olarak kabul edilir ve person tablosu üst öğe olarak kabul edilir, çünkü Favorite_food tablosu bazı verileri için kişi tablosuna bağımlıdır. Her iki tabloya da veri girmeyi planlıyorsanız, Favorite_food ‘a veri girmeden önce üst öğede bir satır oluşturmanız gerekecektir.

person tablosundaki göz rengi sütunu, kahverengi için 'BR', mavi için 'BL' ve yeşil için 'GR' değerleriyle sınırlıdır. Sütunun değerini yanlışlıkla başka bir değere ayarlamaya çalışırsanız, aşağıdaki yanıtı alırsınız:

```
mysql> UPDATE person
    -> SET eye_color = 'ZZ'
    -> WHERE person_id = 1;
ERROR 1265 (01000): Data truncated for column 'eye_color' at row 1
```

Bir tarih sütununu doldurmak için bir dize oluşturursanız ve bu dize beklenen biçimle eşleşmezse, başka bir hata alırsınız. YYYY-AA-GG varsayılan tarih biçimiyle eşleşmeyen bir tarih biçimini kullanan bir örnek:

```
mysql> UPDATE person
    -> SET birth_date = 'DEC-21-1980'
    -> WHERE person_id = 1;
ERROR 1292 (22007): Incorrect date value: 'DEC-21-1980' for column 'birth_date'
at row 1
```

Genel olarak, varsayılan biçime güvenmek yerine biçim dizesini açıkça belirtmek her zaman iyi bir fikirdir. Hangi biçim dizesinin kullanılacağını belirtmek için str_to_date işlevini kullanan ifadenin başka bir sürümü:

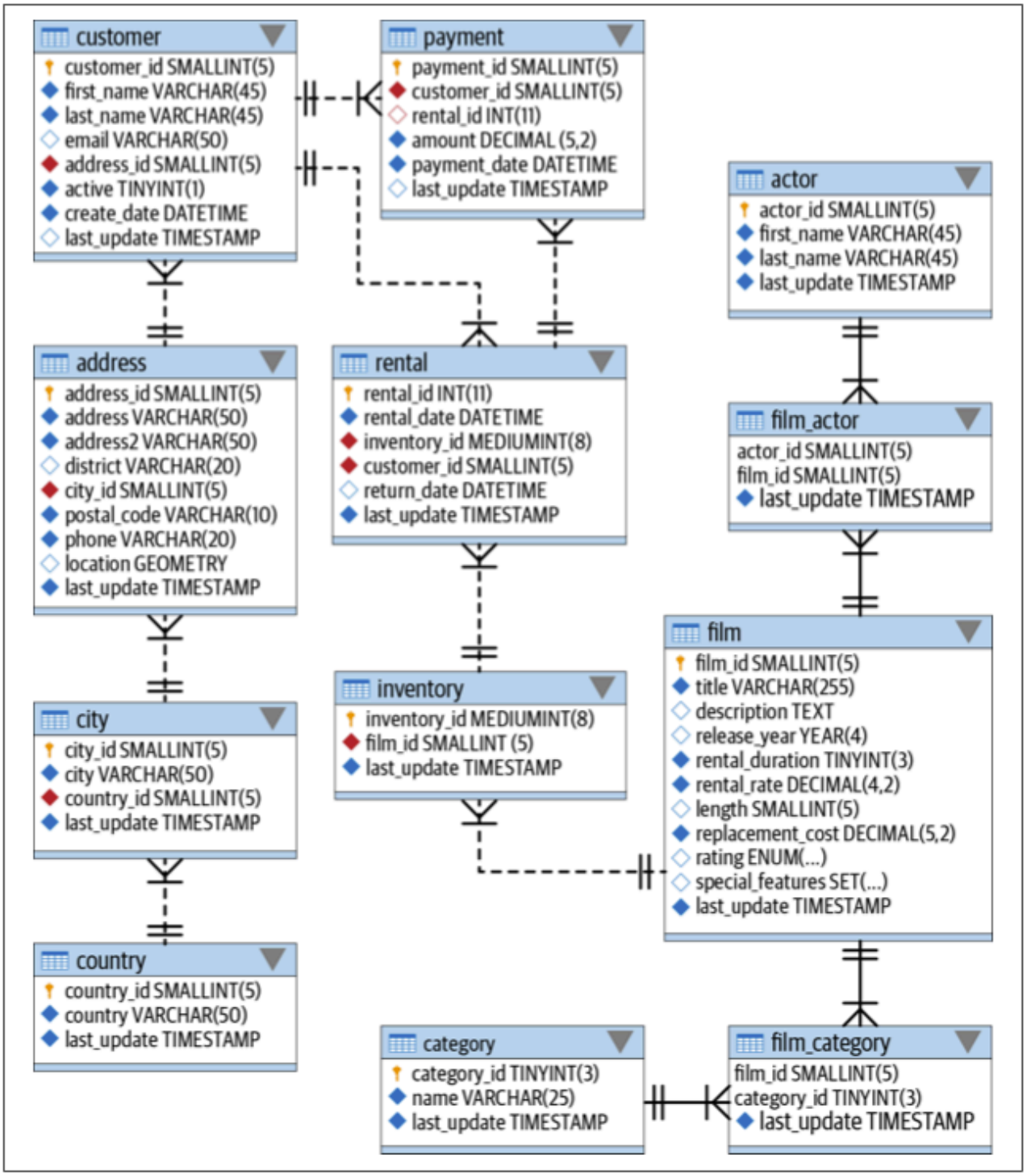
```
mysql> UPDATE person
    -> SET birth_date = str_to_date('DEC-21-1980' , '%b-%d-%Y')
    -> WHERE person_id = 1;
Query OK, 1 row affected (0.12 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Bölümün başlarında, çeşitli zamansal veri türlerini tartıştığımda, YYYY-AA-GG gibi tarih-biçimlendirme dizgilerini gösterdim. Birçok veritabanı sunucusu bu biçimlendirme stilini kullanırken, MySQL dört karakterli bir yılı belirtmek için %Y kullanır. MySQL'de dizeleri tarih saatlerine dönüştürürken ihtiyaç duyabileceğiniz birkaç biçimlendiriciler.

```
%a The short weekday name, such as Sun, Mon, ...
%b The short month name, such as Jan, Feb, ...
%c The numeric month (0..12)
%d The numeric day of the month (00..31)
%f The number of microseconds (000000..999999)
%H The hour of the day, in 24-hour format (00..23)
%h The hour of the day, in 12-hour format (01..12)
%i The minutes within the hour (00..59)
%j The day of year (001..366)
%M The full month name (January..December)
%m The numeric month
%p AM or PM
%s The number of seconds (00..59)
%W The full weekday name (Sunday..Saturday)
%w The numeric day of the week (0=Sunday..6=Saturday)
%Y The four-digit year
```

Sakila Veritabanı

Bu veritabanı, biraz modası geçmiş bir DVD kiralama mağazaları zincirini modelliyor, ancak biraz hayal gücü ile bir video akışı şirketi olarak yeniden adlandırılabilir. Tablolardan bazıları müşteri, film, oyuncu, ödeme, kiralama ve kategoriyi içerir.



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

Table name	Definition
film	A movie that has been released and can be rented
actor	A person who acts in films
customer	A person who watches films
category	A genre of films
payment	A rental of a film by a customer
language	A language spoken by the actors of a film
film_actor	An actor in a film
inventory	A film available for rental

Salika Database için link

```
mysql> desc customer;
+-------------+-------------+------+-----+-------------+----------------------+
| Field       | Type        | Null | Key | Default     | Extra                |
+-------------+-------------+------+-----+-------------+----------------------+
| customer_id | smallint(5)  | NO   | PRI | NULL        | auto_increment       |
               unsigned
| store_id    | tinyint(3)   | NO   | MUL | NULL        |                      |
               unsigned
| first_name  | varchar(45)  | NO   |     | NULL        |                      |
| last_name   | varchar(45)  | NO   | MUL | NULL        |                      |
| email       | varchar(50)  | YES  |     | NULL        |                      |
| address_id  | smallint(5)  | NO   | MUL | NULL        |                      |
               unsigned
| active      | tinyint(1)   | NO   |     | 1           |                      |
| create_date | datetime     | NO   |     | NULL        |                      |
| last_update | timestamp    | YES  |     | CURRENT_    | DEFAULT_GENERATED on
                                        TIMESTAMP     update CURRENT_
                                                      TIMESTAMP          |
+-------------+-------------+------+-----+-------------+----------------------+
```