Veritabanı ve Tablo İşlemleri

MySQL Veri Tipleri

Genel olarak, tüm popüler veritabanı sunucuları, dizeler, tarihler ve sayılar gibi aynı türdeki verileri depolama kapasitesine sahiptir. Tipik olarak farklı oldukları yer, XML ve JSON belgeleri veya uzamsal veriler gibi özel veri türleridir.

Karakter Verileri

Karakter verileri, sabit uzunluklu veya değişken uzunluklu diziler olarak saklanabilir; fark, sabit uzunluklu dizelerin boşluklarla sağdan doldurulması ve her zaman aynı sayıda bayt tüketmesidir. Değişken uzunluklu dizeler ise boşluklarla sağadan doldurulamaz ve her seferinde farklı sayıda baytlık alan kaplar. Bir karakter sütunu tanımlarken, sütunda saklanacak herhangi bir dizenin maksimum boyutunu belirtmelisiniz. Örneğin, uzunluğu 20 karaktere kadar olan dizeleri saklamak istiyorsanız, aşağıdaki tanımlardan birini kullanabilirsiniz:

```
char(20)    /* fixed-length */
varchar(20) /* variable-length */
```

Karakter sütunları için maksimum uzunluk şu anda 255 bayt iken, varchar sütunları 65.535 bayta kadar olabilir. Daha uzun dizeler (e-postalar, XML belgeleri vb.) depolamanız gerekiyorsa, metin türlerinden (orta metin ve uzun metin) birini kullanmak gerekir. Genel olarak, durum kısaltmaları gibi, sütunda saklanacak tüm dizeler aynı uzunlukta olduğunda char türünü, sütunda saklanacak dizeler değişen uzunluklarda olduğunda varchar türünü kullanmalısınız. Hem char hem de varchar, tüm büyük veritabanı sunucularında benzer şekilde kullanılır.

Karakter kümeleri

İngilizce gibi Latin alfabesini kullanan diller için, yeterince az sayıda karakter vardır, öyle ki her bir karakteri depolamak için yalnızca tek bir bayt gerekir. Japonca ve Korece gibi diğer diller çok sayıda karakter içerir, bu nedenle her karakter için birden çok bayt depolama gerektirir. Bu tür karakter kümelerine bu nedenle çok baytlı karakter kümeleri denir.

MySQL, hem tek hem de çok baytlı çeşitli karakter kümelerini kullanarak verileri depolayabilir. Sunucunuzda desteklenen karakter kümelerini görüntülemek için aşağıdaki örnekte gösterildiği gibi show komutunu kullanabilirsiniz:

```
mysql> SHOW CHARACTER SET;
+----------+---------------------------------+--------------------+--------+
| Charset  | Description                     | Default collation  | Maxlen |
+----------+---------------------------------+--------------------+--------+
| armscii8 | ARMSCII-8 Armenian              | armscii8_general_ci |      1 |
| ascii    | US ASCII                        | ascii_general_ci    |      1 |
| big5     | Big5 Traditional Chinese        | big5_chinese_ci     |      2 |
| binary   | Binary pseudo charset           | binary              |      1 |
| cp1250   | Windows Central European        | cp1250_general_ci   |      1 |
| cp1251   | Windows Cyrillic                | cp1251_general_ci   |      1 |
| cp1256   | Windows Arabic                  | cp1256_general_ci   |      1 |
| cp1257   | Windows Baltic                  | cp1257_general_ci   |      1 |
| cp850    | DOS West European               | cp850_general_ci    |      1 |
| cp852    | DOS Central European            | cp852_general_ci    |      1 |
| cp866    | DOS Russian                     | cp866_general_ci    |      1 |
| cp932    | SJIS for Windows Japanese       | cp932_japanese_ci   |      2 |
| dec8     | DEC West European               | dec8_swedish_ci     |      1 |
| eucjpms  | UJIS for Windows Japanese       | eucjpms_japanese_ci |      3 |
| euckr    | EUC-KR Korean                   | euckr_korean_ci     |      2 |
| gb18030  | China National Standard GB18030 | gb18030_chinese_ci  |      4 |
| gb2312   | GB2312 Simplified Chinese       | gb2312_chinese_ci   |      2 |
| gbk      | GBK Simplified Chinese          | gbk_chinese_ci      |      2 |
| geostd8  | GEOSTD8 Georgian                | geostd8_general_ci  |      1 |
| greek    | ISO 8859-7 Greek                | greek_general_ci    |      1 |
| hebrew   | ISO 8859-8 Hebrew               | hebrew_general_ci   |      1 |
| hp8      | HP West European                | hp8_english_ci      |      1 |
| keybcs2  | DOS Kamenicky Czech-Slovak      | keybcs2_general_ci  |      1 |
| koi8r    | KOI8-R Relcom Russian           | koi8r_general_ci    |      1 |
| koi8u    | KOI8-U Ukrainian                | koi8u_general_ci    |      1 |
| latin1   | cp1252 West European            | latin1_swedish_ci   |      1 |
| latin2   | ISO 8859-2 Central European     | latin2_general_ci   |      1 |
| latin5   | ISO 8859-9 Turkish              | latin5_turkish_ci   |      1 |
| latin7   | ISO 8859-13 Baltic              | latin7_general_ci   |      1 |
| macce    | Mac Central European            | macce_general_ci    |      1 |
| macroman | Mac West European               | macroman_general_ci |      1 |
| sjis     | Shift-JIS Japanese              | sjis_japanese_ci    |      2 |
| swe7     | 7bit Swedish                    | swe7_swedish_ci     |      1 |
| tis620   | TIS620 Thai                     | tis620_thai_ci      |      1 |
| ucs2     | UCS-2 Unicode                   | ucs2_general_ci     |      2 |
| ujis     | EUC-JP Japanese                 | ujis_japanese_ci    |      3 |
| utf16    | UTF-16 Unicode                  | utf16_general_ci    |      4 |
| utf16le  | UTF-16LE Unicode                | utf16le_general_ci  |      4 |
| utf32    | UTF-32 Unicode                  | utf32_general_ci    |      4 |
| utf8     | UTF-8 Unicode                   | utf8_general_ci     |      3 |
| utf8mb4  | UTF-8 Unicode                   | utf8mb4_0900_ai_ci  |      4 |
+----------+---------------------------------+--------------------+--------+
41 rows in set (0.04 sec)
```

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

Dördüncü sütundaki maxlen değeri 1'den büyükse, karakter kümesi çok baytlı bir karakter kümesidir.
MySQL sunucusunun önceki sürümlerinde, latin1 karakter seti varsayılan karakter seti olarak otomatik olarak seçilmiştir, ancak sürüm 8 varsayılan olarak utf8mb4'tür. Ancak, veritabanınızdaki her bir karakter sütunu için farklı bir karakter seti kullanmayı seçebilir ve hatta farklı karakter setlerini aynı tablo içinde saklayabilirsiniz. Bir sütun tanımlarken varsayılandan farklı bir karakter kümesi seçmek için, aşağıdaki gibi, tür tanımından sonra desteklenen karakter kümelerinden birini adlandırmanız yeterlidir:

```
varchar(20) character set latin1
```

MySQL ile tüm veritabanınız için varsayılan karakter setini de ayarlayabilirsiniz:

```
create database european_sales character set latin1;
```

Metin Verileri

Varchar sütunları için 64 KB sınırını aşabilecek verileri depolamanız gerekiyorsa, metin türlerinden birini kullanmanız gerekecektir.

Text type	Maximum number of bytes
tinytext	255
text	65,535
mediumtext	16,777,215
longtext	4,294,967,295

Metin türlerinden birini kullanmayı seçerken aşağıdakilerin farkında olmalısınız:
• Bir metin sütununa yüklenen veriler, o tür için maksimum boyutu aşarsa, veriler kesilecektir.
• Veriler sütuna yüklendiğinde sondaki boşluklar kaldırılmayacaktır.
• MySQL, varchar sütunları için 65.535 bayta kadar izin verdiğine göre (sürüm 4'te 255 bayt ile sınırlıydı), minik metin veya metin türünü kullanmaya özel bir ihtiyaç yoktur.
• Serbest biçimli veri girişi için bir sütun oluşturuyorsanız, örneğin şirketinizin müşteri hizmetleri departmanıyla müşteri etkileşimleriyle ilgili verileri tutmak için bir notlar sütunu oluşturuyorsanız, varchar muhtemelen yeterli olacaktır. Ancak belgeleri saklıyorsanız, orta metin veya uzun metin türünü seçmelisiniz.

Sayısal Veriler

“Sayısal” olarak adlandırılan tek bir sayısal veri tipine sahip olmak makul görünse de, aslında burada gösterildiği gibi, sayıların kullanıldığı çeşitli yolları yansıtan birkaç farklı sayısal veri türü vardır:
Bir müşteri siparişinin gönderilip gönderilmediğini gösteren bir sütun;
Boolean olarak adlandırılan bu sütun türü, yanlışı belirtmek için 0 ve doğruyu belirtmek için 1 içerir.
Bir işlem tablosu için sistem tarafından oluşturulan bir birincil anahtar;
Bu veriler genellikle 1'den başlar ve potansiyel olarak çok büyük bir sayıya kadar bir artışla artar.
Müşterinin elektronik alışveriş sepeti için bir ürün numarası;
Bu sütun türünün değerleri, 1 ile belki de 200 arasında pozitif tam sayılar olacaktır.
Devre kartı delme makinesi için konum verileri;
Yüksek hassasiyetli bilimsel veya üretim verileri genellikle sekiz ondalık basamağa kadar doğruluk gerektirir.
Bu tür verileri (ve daha fazlasını) işlemek için MySQL'in birkaç farklı sayısal veri türü vardır. En yaygın olarak kullanılan sayısal türler, tam sayıları veya tam sayıları depolamak için kullanılanlardır. Bu türlerden birini belirtirken, verilerin işaretsiz olduğunu da belirtebilirsiniz; bu, sunucuya sütunda depolanan tüm verilerin sıfırdan büyük veya sıfıra eşit olacağını söyler.

Type	Signed range	Unsigned range
`tinyint`	–128 to 127	0 to 255
`smallint`	–32,768 to 32,767	0 to 65,535
`mediumint`	–8,388,608 to 8,388,607	0 to 16,777,215
`int`	–2,147,483,648 to 2,147,483,647	0 to 4,294,967,295
`bigint`	–2^63 to 2^63 - 1	0 to 2^64 - 1

Tamsayı türlerinden birini kullanarak bir sütun oluşturduğunuzda, MySQL, verileri depolamak için uygun miktarda alan tahsis edecektir, bu, bir tinyint için bir bayttan bir bigint için sekiz bayta kadar değişir. Bu nedenle, depolama alanını boşa harcamadan, sütunda saklandığını hayal edebileceğiniz en büyük sayıyı tutacak kadar büyük bir tür seçmeye çalışmalısınız.

Type	Numeric range
`float(p,s)`	−3.402823466E+38 to −1.175494351E-38 and 1.175494351E-38 to 3.402823466E+38
`double(p,s)`	−1.7976931348623157E+308 to −2.2250738585072014E-308 and 2.2250738585072014E-308 to 1.7976931348623157E+308

Kayan nokta türü kullanırken, bir kesinlik (ondalık noktanın hem solundaki hem de sağındaki izin verilen toplam basamak sayısı) ve bir ölçek (ondalık noktanın sağındaki izin verilen basamak sayısı) belirtebilirsiniz. , ancak bunlar gerekli değildir. Bu değerler Tablo da p ve s olarak gösterilmiştir. Kayan nokta sütununuz için bir kesinlik ve ölçek belirlerseniz, basamak sayısı sütunun ölçeğini ve/veya kesinliğini aşarsa sütunda depolanan verilerin yuvarlanacağını unutmayın. Örneğin, float(4,2) olarak tanımlanan bir sütun, ikisi ondalık sayının solunda ve ikisi ondalık sayının sağında olmak üzere toplam dört basamak depolayacaktır. Bu nedenle, böyle bir sütun 27.44 ve 8.19 sayılarını gayet iyi işleyebilir, ancak 17.8675 sayısı 17.87'ye yuvarlanır ve 178.375 sayısını kayan nokta (4,2) sütununuzda saklamaya çalışmak bir hata oluşturur.

Tamsayı türleri gibi, kayan noktalı sütunlar da işaretsiz olarak tanımlanabilir, ancak bu atama, sütunda depolanabilecek veri aralığını değiştirmek yerine yalnızca negatif sayıların sütunda depolanmasını önler.

Zaman Verileri

Dizeler ve sayılarla birlikte, tarihler ve/veya saatler hakkında bilgilerle çalışıyor olacaksınız. Bu tür verilere zamansal olarak atıfta bulunulur ve bir veritabanındaki bazı zaman veri örnekleri şunları içerir:

• Bir müşterinin siparişini göndermek gibi belirli bir olayın gerçekleşmesinin beklendiği gelecekteki tarih
• Bir müşterinin siparişinin gönderildiği tarih
• Bir kullanıcının tablodaki belirli bir satırı değiştirdiği tarih ve saat
• Bir çalışanın doğum tarihi
• Bir veri ambarındaki yearly_sales olgu tablosundaki bir satıra karşılık gelen yıl
• Bir otomobil montaj hattındaki kablo demetini tamamlamak için gereken ve geçen süre

Type	Default format	Allowable values
date	YYYY-MM-DD	1000-01-01 to 9999-12-31
datetime	YYYY-MM-DD HH:MI:SS	1000-01-01 00:00:00.000000 to 9999-12-31 23:59:59.999999

Type	Default format	Allowable values
timestamp	YYYY-MM-DD HH:MI:SS	1970-01-01 00:00:00.000000 to 2038-01-18 22:14:07.999999
year	YYYY	1901 to 2155
time	HHH:MI:SS	-838:59:59.000000 to 838:59:59.000000

Veritabanı sunucuları zamansal verileri çeşitli şekillerde depolarken, bir biçim dizesinin amacı, verinin alındığında nasıl temsil edileceğini ve eklerken veya güncellerken bir tarih dizesinin nasıl oluşturulması gerektiğini göstermektir. 23 Mart 2020 tarihini YYYY-AA-GG varsayılan biçimini kullanarak bir tarih sütununa eklemek isterseniz, '2020-03-23' dizesini kullanırsınız.

datetime, timestamp, and time türleri ayrıca 6 ondalık basamağa (mikrosaniye) kadar kesirli saniyelere izin verir. Bu veri türlerinden birini kullanarak sütunları tanımlarken 0 ile 6 arasında bir değer sağlayabilirsiniz; örneğin, datetime(2) öğesinin belirtilmesi, zaman değerlerinizin saniyenin yüzde birini içermesine olanak tanır.

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

Daha önce gösterilen örnekleri uygulamak için çeşitli zamansal türlerin nasıl kullanılacağı aşağıda açıklanmıştır:

- Bir müşteri siparişinin beklenen gelecekteki sevkiyat tarihini ve bir çalışanın doğum tarihini tutan sütunlar, gelecekteki bir sevkiyatı planlamak gerçekçi olmadığı için date türünü kullanır.
- Bir müşteri siparişinin gerçekten ne zaman gönderildiğiyle ilgili bilgileri tutan bir sütun, yalnızca gönderinin gerçekleştiği tarihi değil, aynı zamanda saati de izlemek önemli olduğundan, davetime türünü kullanır.
- Bir kullanıcının bir tablodaki belirli bir satırı en son ne zaman değiştirdiğini izleyen bir sütun, timestamp türünü kullanır. timestamp türü, datetime türüyle (yıl, ay, gün, saat, dakika, saniye) aynı bilgileri tutar, ancak bir satır eklendiğinde veya değiştirildiğinde MySQL sunucusu tarafından bir timestamp sütunu otomatik olarak geçerli tarih/saatle doldurulur.
- Yalnızca yıl verilerini tutan bir sütun, year türünü kullanır.
- Sütunlar bir görevi tamamlamak için gereken süreye ilişkin verileri tutan time türünü kullanır. Bu tür veriler için, yalnızca görevi tamamlamak için gereken saat/dakika/saniye sayısıyla ilgilendiğiniz için bir tarih bileşenini depolamak gereksiz ve kafa karıştırıcı olacaktır. Bu bilgi, iki datetime sütunu (biri görev başlangıç tarihi/saati ve diğeri görev tamamlama tarihi/saati için) kullanılarak ve biri diğerinden çıkarılarak elde edilebilir, ancak tek bir saat sütunu kullanmak daha kolaydır.

Tablo İşlemleri

Adım 1: Tasarım

Bir tablo tasarlamaya başlamanın ilk adımı ne tür bilgilerin eklenmesinin faydalı olacağını görmek için biraz beyin fırtınası yapmaktır. Kişiler tablosu için:

Column	Type	Allowable values
name	varchar(40)	
eye_color	char(2)	BL, BR, GR
birth_date	date	
address	varchar(100)	
favorite_foods	varchar(200)	

Adım 2: Geliştirme

Kişi tablosundaki sütunlara ikinci kez bakıldığında şu sorunlar ortaya çıkıyor:

- Ad sütunu aslında bir ad ve soyadından oluşan bileşik bir nesnedir.
- Birden fazla kişi aynı ada, göz rengine, doğum tarihine vb. sahip olabileceğinden, kişi tablosunda benzersizliği garanti eden sütunlar yoktur.
- Adres sütunu ayrıca sokak, şehir, eyalet/il, ülke ve posta kodundan oluşan bileşik bir nesnedir.
- Favorite_foods sütunu, sıfır, bir veya daha fazla bağımsız öğe içeren bir listedir. Kişi tablosunun yabancı anahtarını içeren bu veriler için ayrı bir tablo oluşturmak en iyisi olacaktır, böylece belirli bir yiyeceğin hangi kişiye atfedilebileceğini bilebilirsiniz. Bu hususlar dikkate alındıktan sonra, normalleştirilmiş bir versiyon oluşturulabilir.

person tablosu

Column	Type	Allowable values
person_id	smallint (unsigned)	
first_name	varchar(20)	
last_name	varchar(20)	
eye_color	char(2)	BR, BL, GR
birth_date	date	
street	varchar(30)	
city	varchar(20)	
state	varchar(20)	
country	varchar(20)	
postal_code	varchar(20)	

Kişi tablosunun benzersizliği garanti edecek bir birincil anahtarı (person_id) olduğuna göre, sonraki adım, kişi tablosunun yabancı anahtarını içeren bir favori_yemek tablosu oluşturmaktır.

favorite_food tablosu

Column	Type
person_id	smallint (unsigned)
food	varchar(20)

favorite_food sütununu kişi tablosundan çıkarmak kesinlikle iyi bir fikirdi, ama işimiz bitti mi? Örneğin, bir kişi favori yemek olarak “makarna”yı listelerse, başka bir kişi “spagetti”yi listelerse ne olur? Bu sorunu önlemek için, insanların bir seçenekler listesinden en sevdikleri yiyecekleri seçmelerini sağlamak gerekir, bu durumda food_id ve food_name sütunlarıyla bir food tablosu oluşturmalı ve ardından favorite_food tablosunu şu şekilde değiştirmelisiniz. Food tablosundan bir yabancıl anahtar favorite_food tablosuna eklenmelidir.

Adım 3: SQL Şema İfadeleri Oluşturma

```
CREATE TABLE person
 (person_id SMALLINT UNSIGNED,
  fname VARCHAR(20),
  lname VARCHAR(20),
  eye_color CHAR(2),
  birth_date DATE,
  street VARCHAR(30),
  city VARCHAR(20),
  state VARCHAR(20),
  country VARCHAR(20),
  postal_code VARCHAR(20),
  CONSTRAINT pk_person PRIMARY KEY (person_id)
 );
```

Bu ifadedeki her şey, son madde dışında oldukça açıklayıcıdır; tablonuzu tanımladığınızda, veritabanı sunucusuna tablo için birincil anahtar olarak hangi sütun veya sütunların hizmet edeceğini söylemeniz gerekir. Bunu tablo üzerinde bir kısıtlama oluşturarak yaparsınız. Bir tablo tanımına birkaç tür kısıtlama ekleyebilirsiniz. Burada kısıtlama, birincil anahtar kısıtlaması vardır person_id sütununu için oluşuturulur ve pk_person adı verilir.

Kişi tablosu için faydalı olabilecek başka bir kısıtlama türü daha vardır. Kontrol kısıtlaması adı verilen başka bir kısıtlama türü, belirli bir sütun için izin verilen değerleri sınırlar. MySQL, aşağıdaki gibi bir sütun tanımına bir kontrol kısıtlamasının eklenmesine izin verir:

```
eye_color CHAR(2) CHECK (eye_color IN ('BR','BL','GR')),
```

Kontrol kısıtlamaları çoğu veritabanı sunucusunda beklendiği gibi çalışırken, MySQL sunucusu kontrol kısıtlamalarının tanımlanmasına izin verir ancak bunları için bir zorunluluk sağlamaz.MySQL, CHECK kısıtlamasını veri türü tanımıyla birleştiren ENUM adlı başka bir karakter veri türü sağlayan bir ifade vardır.

```
eye_color ENUM('BR','BL','GR'),
```

```
CREATE TABLE person
 (person_id SMALLINT UNSIGNED,
  fname VARCHAR(20),
  lname VARCHAR(20),
  eye_color ENUM('BR','BL','GR'),
  birth_date DATE,
  street VARCHAR(30),
  city VARCHAR(20),
  state VARCHAR(20),
  country VARCHAR(20),
  postal_code VARCHAR(20),
  CONSTRAINT pk_person PRIMARY KEY (person_id)
 );
mysql> desc person;
+-------------+----------------------+------+-----+---------+-------+
| Field       | Type                 | Null | Key | Default | Extra |
+-------------+----------------------+------+-----+---------+-------+
| person_id   | smallint(5) unsigned | NO   | PRI | NULL    |       |
| fname       | varchar(20)          | YES  |     | NULL    |       |
| lname       | varchar(20)          | YES  |     | NULL    |       |
| eye_color   | enum('BR','BL','GR') | YES  |     | NULL    |       |
| birth_date  | date                 | YES  |     | NULL    |       |
| street      | varchar(30)          | YES  |     | NULL    |       |
| city        | varchar(20)          | YES  |     | NULL    |       |
| state       | varchar(20)          | YES  |     | NULL    |       |
| country     | varchar(20)          | YES  |     | NULL    |       |
| postal_code | varchar(20)          | YES  |     | NULL    |       |
+-------------+----------------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

Açıklama çıktısının 1. ve 2. sütunları açıklayıcıdır. Sütun 3, tabloya veri eklendiğinde belirli bir sütunun atlanıp atlanamayacağını gösterir. Dördüncü sütun, bir sütunun herhangi bir anahtarda (birincil veya yabancı) yer alıp almadığını gösterir; bu durumda person_id sütunu birincil anahtar olarak işaretlenir. Sütun 5, tabloya veri eklerken sütunu atlarsanız belirli bir sütunun varsayılan bir değerle doldurulup doldurulmayacağını gösterir. Altıncı sütun, bir sütuna uygulanabilecek diğer ilgili bilgileri gösterir.

NULL Nedir?

Bazı durumlarda, tablonuzdaki belirli bir sütun için bir değer sağlamak mümkün veya uygulanabilir değildir. Örneğin, yeni bir müşteri siparişi hakkında veri eklerken, ship_date sütunu henüz belirlenemez. Bu durumda, sütunun **null** olduğu söylenir. Bu da bir değerin olmadığını gösterir. Bir tablo tasarlarken, hangi sütunların **null** (varsayılan) ve hangi sütunların null olmasına izin verilmeyeceğini belirtebilirsiniz (tür tanımından sonra **not null** anahtar sözcüğü eklenerek belirlenir).

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

```
mysql>  CREATE TABLE favorite_food
    ->  (person_id SMALLINT UNSIGNED,
    ->  food VARCHAR(20),
    ->  CONSTRAINT pk_favorite_food PRIMARY KEY (person_id, food),
    ->  CONSTRAINT fk_fav_food_person_id FOREIGN KEY (person_id)
    ->  REFERENCES person (person_id)
    ->  );
Query OK, 0 rows affected (0.10 sec)
```

Bir kişinin birden fazla favori yemeği olabileceğinden (bu tablonun ilk başta oluşturulmasının nedeni budur), tabloda benzersizliği garanti etmek için sadece person_id sütunundan daha fazlasını gerektirir. Dolayısıyla bu tablo iki sütunlu bir birincil anahtara sahiptir: person_id ve food.

Favori_yemek tablosu, yabancı anahtar kısıtlaması adı verilen başka bir kısıtlama türü içerir. **Bu, favori_yemek tablosundaki person_id sütununun değerlerini yalnızca kişi tablosunda bulunan değerleri içerecek şekilde kısıtlar.** Bu kısıtlama uygulandığında, person_id'si 27 olan kişinin pizzayı sevdiğini belirten bir değeri favori_food tablosuna kişi tablosunda zaten 27 olan bir satır yoksa ekleme olamayacak anlamına gelir.

```
mysql> desc favorite_food;
+-----------+----------------------+------+-----+---------+-------+
| Field     | Type                 | Null | Key | Default | Extra |
+-----------+----------------------+------+-----+---------+-------+
| person_id | smallint(5) unsigned | NO   | PRI | NULL    |       |
| food      | varchar(20)          | NO   | PRI | NULL    |       |
+-----------+----------------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```