Çoklu Tablo Sorguları ve Kümeler

Join

Normalleştirme ile birden fazla tabloya ayrılmış verilerin sorgularla tekrar birleştirilmesi gerekir.

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

mysql> desc address;
+-------------+----------------------+------+-----+-------------------+
| Field       | Type                 | Null | Key | Default           |
+-------------+----------------------+------+-----+-------------------+
| address_id  | smallint(5) unsigned | NO   | PRI | NULL              |
| address     | varchar(50)          | NO   |     | NULL              |
| address2    | varchar(50)          | YES  |     | NULL              |
| district    | varchar(20)          | NO   |     | NULL              |
| city_id     | smallint(5) unsigned | NO   | MUL | NULL              |
| postal_code | varchar(10)          | YES  |     | NULL              |
| phone       | varchar(20)          | NO   |     | NULL              |
| location    | geometry             | NO   | MUL | NULL              |
| last_update | timestamp            | NO   |     | CURRENT_TIMESTAMP |
+-------------+----------------------+------+-----+-------------------+
```

Her müşterinin adını ve soyadını adresiyle birlikte almak istediğinizi varsayalım. Bu nedenle sorgunuzun customer.first_name, customer.last_name ve address.address sütunlarını alması gerekir. Ancak aynı sorguda her iki tablodan nasıl veri alabilirsiniz?

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

Cevap, adres tablosundaki müşterinin kaydının kimliğini tutan customer.address_id sütununda yatar (daha resmi bir ifadeyle, customer.address_id sütunu, adres tablosunun yabancı anahtarıdır).

Bir tablodaki değerlerin başka bir tabloda bulunduğunu doğrulamak için isteğe bağlı olarak bir yabancı anahtar kısıtlaması oluşturulabilir. Önceki örnek için, customer.address_id sütununa eklenen tüm değerlerin address.address_id sütununda bulunmasını sağlamak için müşteri tablosunda bir yabancı anahtar kısıtlaması oluşturulabilir. **İki tabloyu birleştirmek için bir yabancı anahtar kısıtlamasının olması gerekmediğini unutmayın.**

Kartezyen

Başlamanın en kolay yolu, müşteri ve adres tablolarını bir sorgunun from yan tümcesine koymak ve ne olduğunu görmek. Burada, join anahtar sözcüğüyle ayrılmış iki tabloyu adlandıran bir from yan tümcesi ile birlikte, müşterinin adını ve soyadını adresiyle birlikte alan bir sorgu yer alır:

```
mysql> SELECT c.first_name, c.last_name, a.address
    -> FROM customer c JOIN address a;
+------------+-----------+---------------------+
| first_name | last_name | address             |
+------------+-----------+---------------------+
| MARY       | SMITH     | 47 MySakila Drive   |
| PATRICIA   | JOHNSON   | 47 MySakila Drive   |
| LINDA      | WILLIAMS  | 47 MySakila Drive   |
| BARBARA    | JONES     | 47 MySakila Drive   |
| ELIZABETH  | BROWN     | 47 MySakila Drive   |
| JENNIFER   | DAVIS     | 47 MySakila Drive   |
| MARIA      | MILLER    | 47 MySakila Drive   |
| SUSAN      | WILSON    | 47 MySakila Drive   |
...
| SETH       | HANNON    | 1325 Fukuyama Street |
| KENT       | ARSENAULT | 1325 Fukuyama Street |
| TERRANCE   | ROUSH     | 1325 Fukuyama Street |
| RENE       | MCALISTER | 1325 Fukuyama Street |
| EDUARDO    | HIATT     | 1325 Fukuyama Street |
| TERRENCE   | GUNDERSON | 1325 Fukuyama Street |
| ENRIQUE    | FORSYTHE  | 1325 Fukuyama Street |
| FREDDIE    | DUGGAN    | 1325 Fukuyama Street |
| WADE       | DELVALLE  | 1325 Fukuyama Street |
| AUSTIN     | CINTRON   | 1325 Fukuyama Street |
+------------+-----------+---------------------+
361197 rows in set (0.03 sec)
```

Adres tablosunda sadece 599 müşteri ve 603 adres satırı var, peki sonuç kümesi nasıl 361.197 satır ile sonuçlandı? Daha yakından baktığınızda, müşterilerin çoğunun aynı adrese sahip olduğunu görebilirsiniz. Sorgu iki tablonun nasıl birleştirileceğini belirtmediği için, veritabanı sunucusu iki tablonun ikili permütasyonu olan Kartezyen ürününü üretti (599 müşteri x 603 adres = 361.197 permütasyon). Bu tür birleştirme, çapraz birleştirme olarak bilinir ve nadiren kullanılır (en azından bilerek).

Inner Join

Her müşteri için yalnızca tek bir satır döndürülecek şekilde önceki sorguyu değiştirmek için, iki tablonun nasıl ilişkili olduğunu açıklamanız gerekir. Daha önce, customer.address_id sütununun iki tablo arasında bağlantı görevi gördüğünü göstermiştik, bu nedenle bu bilginin from yan tümcesinin **on** alt maddesine eklenmesi gerekiyor:

```
mysql> SELECT c.first_name, c.last_name, a.address
    -> FROM customer c JOIN address a
    ->   ON c.address_id = a.address_id;
+-------------+--------------+----------------------------------------+
| first_name  | last_name    | address                                |
+-------------+--------------+----------------------------------------+
| MARY        | SMITH        | 1913 Hanoi Way                         |
| PATRICIA    | JOHNSON      | 1121 Loja Avenue                       |
| LINDA       | WILLIAMS     | 692 Joliet Street                      |
| BARBARA     | JONES        | 1566 Inegl Manor                       |
| ELIZABETH   | BROWN        | 53 Idfu Parkway                        |
| JENNIFER    | DAVIS        | 1795 Santiago de Compostela Way        |
| MARIA       | MILLER       | 900 Santiago de Compostela Parkway     |
| SUSAN       | WILSON       | 478 Joliet Way                         |
| MARGARET    | MOORE        | 613 Korolev Drive                      |
...
| TERRANCE    | ROUSH        | 42 Fontana Avenue                      |
| RENE        | MCALISTER    | 1895 Zhezqazghan Drive                 |
| EDUARDO     | HIATT        | 1837 Kaduna Parkway                    |
| TERRENCE    | GUNDERSON    | 844 Bucuresti Place                    |
| ENRIQUE     | FORSYTHE     | 1101 Bucuresti Boulevard               |
| FREDDIE     | DUGGAN       | 1103 Quilmes Boulevard                 |
| WADE        | DELVALLE     | 1331 Usak Boulevard                    |
| AUSTIN      | CINTRON      | 1325 Fukuyama Street                   |
+-------------+--------------+----------------------------------------+
599 rows in set (0.00 sec)
```

361.197 satır yerine, sunucuya bir tablodan diğerine geçmek için address_id sütununu kullanarak müşteri ve adres tablolarına **join** talimatı veren **on** alt maddesinin eklenmesi nedeniyle artık beklenen 599 satırınız var. Örneğin, Mary Smith'in müşteri tablosundaki satırı address_id sütununda 5 değerini içerir (örnekte gösterilmemiştir). Sunucu bu değeri adres tablosundaki address_id sütununda 5 değerine sahip satırı aramak için kullanır ve ardından o satırdaki adres sütunundan '1913 Hanoi Way' değerini alır.

Bir tabloda address_id sütunu için bir değer varsa, diğerinde yoksa, bu değeri içeren satırlar için birleştirme başarısız olur ve bu satırlar sonuç kümesinden çıkarılır. Bu birleşim türü **inner join** olarak bilinir ve en sık kullanılan birleşim türüdür. Açıklığa kavuşturmak gerekirse, müşteri tablosundaki bir satır address_id sütununda 999 değerine sahipse ve adres tablosunda address_id sütununda 999 değerinde bir satır yoksa, o müşteri satırı sonuç kümesine dahil edilmeyecektir. Bir eşleşme olup olmadığına bakılmaksızın bir tablodaki veya diğerindeki tüm satırları dahil etmek istiyorsanız, diğer birleştirme türlerini bilmelisiniz.

Önceki örnekte, hangi tür birleştirmenin kullanılacağını from yan tümcesinde belirtmedim. Ancak, bir iç birleştirme kullanarak iki tabloyu birleştirmek istediğinizde, bunu from yan tümcenizde açıkça belirtmelisiniz.

```
SELECT c.first_name, c.last_name, a.address
FROM customer c INNER JOIN address a
  ON c.address_id = a.address_id;
```

Birleştirme türünü belirtmezseniz, sunucu varsayılan olarak bir inner join birleştimesi yapacaktır. İki tabloyu birleştirmek için kullanılan sütunların adları aynıysa, aşağıdaki gibi **on** yerine **using** alt maddesini kullanabilirsiniz:

```
SELECT c.first_name, c.last_name, a.address
FROM customer c INNER JOIN address a
  USING (address_id);
```

ANSI Join Syntax

Tabloları birleştirmek için kullanılan gösterim, ANSI SQL standardının SQL92 sürümünde tanıtıldı. Tüm büyük veritabanları (Oracle Database, Microsoft SQL Server, MySQL, IBM DB2 Universal Database ve Sybase Adaptive Server) SQL92 birleştirme sözdizimini benimsemiştir.

Daha Fazla Tablo Birleştirmek

Üç tabloyu birleştirmek, iki tabloyu birleştirmeye benzer, ancak ufak bir fark vardır. İki tablo birleştirme ile, from yan tümcesinde iki tablo ve bir birleştirme türü ve tabloların nasıl birleştirileceğini tanımlamak için bir **on** yan tümcesi vardır. Üç tablo birleştirme ile, from yan tümcesinde üç tablo ve iki birleştirme türü ve iki on yan tümcesi bulunmalıdır.
Örnek vermek gerekirse, önceki sorguyu müşterinin sokak adresi yerine şehrini döndürecek şekilde değiştirelim. Ancak şehir adı, adres tablosunda saklanmaz, ancak şehir tablosuna yabancı bir anahtar aracılığıyla erişilir.

```
mysql> desc address;
+-------------+----------------------+------+-----+-------------------+
| Field       | Type                 | Null | Key | Default           |
+-------------+----------------------+------+-----+-------------------+
| address_id  | smallint(5) unsigned | NO   | PRI | NULL              |
| address     | varchar(50)          | NO   |     | NULL              |
| address2    | varchar(50)          | YES  |     | NULL              |
| district    | varchar(20)          | NO   |     | NULL              |
| city_id     | smallint(5) unsigned | NO   | MUL | NULL              |
| postal_code | varchar(10)          | YES  |     | NULL              |
| phone       | varchar(20)          | NO   |     | NULL              |
| location    | geometry             | NO   | MUL | NULL              |
| last_update | timestamp            | NO   |     | CURRENT_TIMESTAMP |
+-------------+----------------------+------+-----+-------------------+

mysql> desc city;
+-------------+----------------------+------+-----+-------------------+
| Field       | Type                 | Null | Key | Default           |
+-------------+----------------------+------+-----+-------------------+
| city_id     | smallint(5) unsigned | NO   | PRI | NULL              |
| city        | varchar(50)          | NO   |     | NULL              |
| country_id  | smallint(5) unsigned | NO   | MUL | NULL              |
| last_update | timestamp            | NO   |     | CURRENT_TIMESTAMP |
+-------------+----------------------+------+-----+-------------------+
```

Her müşterinin şehrini göstermek için, address_id sütununu kullanarak müşteri tablosundan adres tablosuna ve ardından city_id sütununu kullanarak adres tablosundan şehir tablosuna geçmeniz gerekir.

```
mysql> SELECT c.first_name, c.last_name, ct.city
    -> FROM customer c
    ->   INNER JOIN address a
    ->   ON c.address_id = a.address_id
    ->   INNER JOIN city ct
    ->   ON a.city_id = ct.city_id;
+-------------+--------------+---------------------------+
| first_name  | last_name    | city                      |
+-------------+--------------+---------------------------+
| JULIE       | SANCHEZ      | A Corua (La Corua)        |
| PEGGY       | MYERS        | Abha                      |
| TOM         | MILNER       | Abu Dhabi                 |
| GLEN        | TALBERT      | Acua                      |
| LARRY       | THRASHER     | Adana                     |
| SEAN        | DOUGLASS     | Addis Abeba               |
...
| MICHELE     | GRANT        | Yuncheng                  |
| GARY        | COY          | Yuzhou                    |
| PHYLLIS     | FOSTER       | Zalantun                  |
| CHARLENE    | ALVAREZ      | Zanzibar                  |
| FRANKLIN    | TROUTMAN     | Zaoyang                   |
| FLOYD       | GANDY        | Zapopan                   |
| CONSTANCE   | REID         | Zaria                     |
| JACK        | FOUST        | Zeleznogorsk              |
| BYRON       | BOX          | Zhezqazghan               |
| GUY         | BROWNLEE     | Zhoushan                  |
| RONNIE      | RICKETTS     | Ziguinchor                |
+-------------+--------------+---------------------------+
599 rows in set (0.03 sec)
```

Bu sorgu için, from yan tümcesinde üç tablo, iki birleştirme türü ve iki **on** alt cümle vardır, bu nedenle işler biraz daha yoğun hale geldi. İlk bakışta, from yan tümcesinde tabloların göründüğü sıra önemli gibi görünebilir, ancak tablo sırasını değiştirirseniz, aynı sonuçları alırsınız. Bu varyasyonların üçü de aynı sonuçları verir:

```
SELECT c.first_name, c.last_name, ct.city
FROM customer c
  INNER JOIN address a
  ON c.address_id = a.address_id
  INNER JOIN city ct
  ON a.city_id = ct.city_id;

SELECT c.first_name, c.last_name, ct.city
FROM city ct
  INNER JOIN address a
  ON a.city_id = ct.city_id
  INNER JOIN customer c
  ON c.address_id = a.address_id;

SELECT c.first_name, c.last_name, ct.city
FROM address a
  INNER JOIN city ct
  ON a.city_id = ct.city_id
  INNER JOIN customer c
  ON c.address_id = a.address_id;
```

Alt Sorgulardan Gelen Tablolar

```
mysql> SELECT c.first_name, c.last_name, addr.address, addr.city
    -> FROM customer c
    ->   INNER JOIN
    ->    (SELECT a.address_id, a.address, ct.city
    ->     FROM address a
    ->       INNER JOIN city ct
    ->       ON a.city_id = ct.city_id
    ->     WHERE a.district = 'California'
    ->    ) addr
    ->   ON c.address_id = addr.address_id;
+------------+-----------+-----------------------+----------------+
| first_name | last_name | address               | city           |
+------------+-----------+-----------------------+----------------+
| PATRICIA   | JOHNSON   | 1121 Loja Avenue      | San Bernardino |
| BETTY      | WHITE     | 770 Bydgoszcz Avenue  | Citrus Heights |
| ALICE      | STEWART   | 1135 Izumisano Parkway| Fontana        |
| ROSA       | REYNOLDS  | 793 Cam Ranh Avenue   | Lancaster      |
| RENEE      | LANE      | 533 al-Ayn Boulevard  | Compton        |
| KRISTIN    | JOHNSTON  | 226 Brest Manor       | Sunnyvale      |
| CASSANDRA  | WALTERS   | 920 Kumbakonam Loop   | Salinas        |
| JACOB      | LANCE     | 1866 al-Qatif Avenue  | El Monte       |
| RENE       | MCALISTER | 1895 Zhezqazghan Drive| Garden Grove   |
+------------+-----------+-----------------------+----------------+
9 rows in set (0.00 sec)
```

4. satırda başlayan ve takma ad olara **addr** değeri verilen alt sorgu, Kaliforniya'daki tüm adresleri bulur. Dış sorgu, California'da yaşayan tüm müşterilerin adını, soyadını, sokak adresini ve şehrini döndürmek için alt sorgu sonuçlarını müşteri tablosuyla birleştirir. Bu sorgu bir alt sorgu kullanılmadan sadece üç tablonun birleştirilmesiyle yazılabilirken, bazen bir veya daha fazla alt sorgu kullanmak performans ve/veya okunabilirlik açısından avantajlı olabilir.
Neler olduğunu görselleştirmenin bir yolu, alt sorguyu kendi başına çalıştırmak ve sonuçlara bakmaktır.

Bu sonuç seti, dokuz California adresinin tamamından oluşur. address_id sütunu aracılığıyla müşteri tablosuna katıldığınızda, sonuç kümeniz bu adreslere atanan müşteriler hakkında bilgi içerecektir.

```
mysql> SELECT a.address_id, a.address, ct.city
    -> FROM address a
    ->   INNER JOIN city ct
    ->   ON a.city_id = ct.city_id
    -> WHERE a.district = 'California';
+------------+-----------------------+---------------+
| address_id | address               | city          |
+------------+-----------------------+---------------+
|          6 | 1121 Loja Avenue      | San Bernardino |
|         18 | 770 Bydgoszcz Avenue  | Citrus Heights |
|         55 | 1135 Izumisano Parkway | Fontana       |
|        116 | 793 Cam Ranh Avenue   | Lancaster     |
|        186 | 533 al-Ayn Boulevard  | Compton       |
|        218 | 226 Brest Manor       | Sunnyvale     |
|        274 | 920 Kumbakonam Loop   | Salinas       |
|        425 | 1866 al-Qatif Avenue  | El Monte      |
|        599 | 1895 Zhezqazghan Drive | Garden Grove  |
+------------+-----------------------+---------------+
9 rows in set (0.00 sec)
```

Birden fazla tablo birleştiriyorsanız, aynı tabloya birden fazla join işlemi gerektiğini görebilirsiniz. Örnek veri tabanında, örneğin, aktörler, film_aktör tablosu aracılığıyla göründükleri filmlerle ilişkilidir. İki belirli oyuncunun yer aldığı tüm filmleri bulmak isteniyorsa:

```
mysql> SELECT f.title
    -> FROM film f
    ->   INNER JOIN film_actor fa
    ->   ON f.film_id = fa.film_id
    ->   INNER JOIN actor a
    ->   ON fa.actor_id = a.actor_id
    -> WHERE ((a.first_name = 'CATE' AND a.last_name = 'MCQUEEN')
    ->     OR (a.first_name = 'CUBA' AND a.last_name = 'BIRCH'));
+----------------------+
| title                |
+----------------------+
| ATLANTIS CAUSE       |
| BLOOD ARGONAUTS      |
| COMMANDMENTS EXPRESS |
| DYNAMITE TARZAN      |
| EDGE KISSING         |
...
| TOWERS HURRICANE     |
| TROJAN TOMORROW      |
| VIRGIN DAISY         |
| VOLCANO TEXAS        |
| WATERSHIP FRONTIER   |
+----------------------+
54 rows in set (0.00 sec)
```

Bu sorgu, Cate McQueen veya Cuba Birch'in göründüğü tüm filmleri döndürür. Bu iki oyuncunun beraber rol aldığı filmleri almak istediğinizi varsayalım. Bunu başarmak için, film_actor tablosunda biri Cate McQueen ve diğeri Cuba Birch ile ilişkili olan iki satırı olan film tablosundaki tüm satırları bulmanız gerekecek. Bu nedenle, sunucunun çeşitli maddelerde hangisinden bahsettiğinizi bilmesi için, film_actor ve actor tablolarını her biri farklı bir takma adla iki kez eklemeniz gerekir:

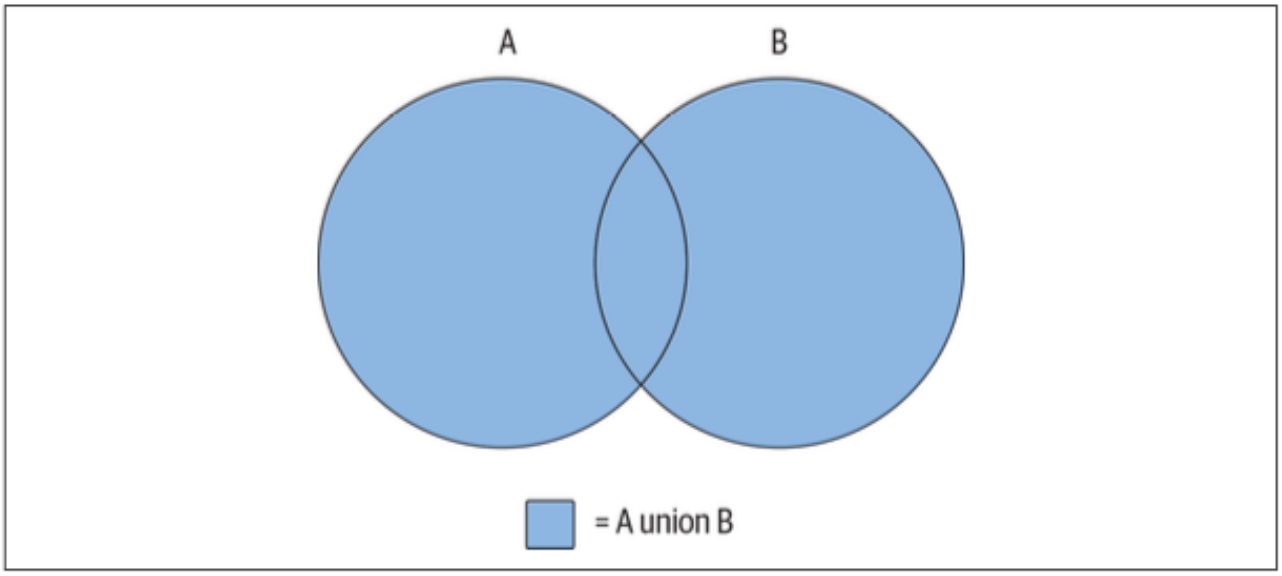
```
mysql> SELECT f.title
    ->  FROM film f
    ->    INNER JOIN film_actor fa1
    ->    ON f.film_id = fa1.film_id
    ->    INNER JOIN actor a1
    ->    ON fa1.actor_id = a1.actor_id
    ->    INNER JOIN film_actor fa2
    ->    ON f.film_id = fa2.film_id
    ->    INNER JOIN actor a2
    ->    ON fa2.actor_id = a2.actor_id
    -> WHERE (a1.first_name = 'CATE' AND a1.last_name = 'MCQUEEN')
    ->   AND (a2.first_name = 'CUBA' AND a2.last_name = 'BIRCH');
+------------------+
| title            |
+------------------+
| BLOOD ARGONAUTS  |
| TOWERS HURRICANE |
+------------------+
2 rows in set (0.00 sec)
```

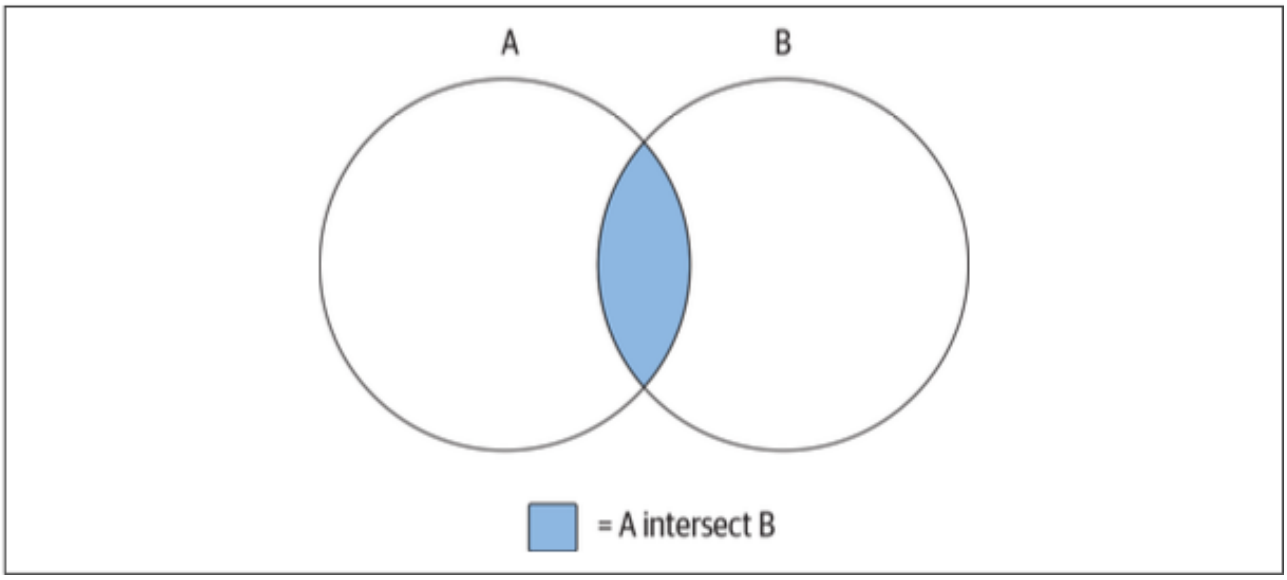
İki oyuncu 52 farklı filmde yer aldı ancak her iki oyuncunun da yer aldığı sadece iki film var. Bu, aynı tablolar birden çok kez kullanıldığından, tablo takma adlarının kullanılmasını gerektiren bir sorgu örneğidir.

Kümeler

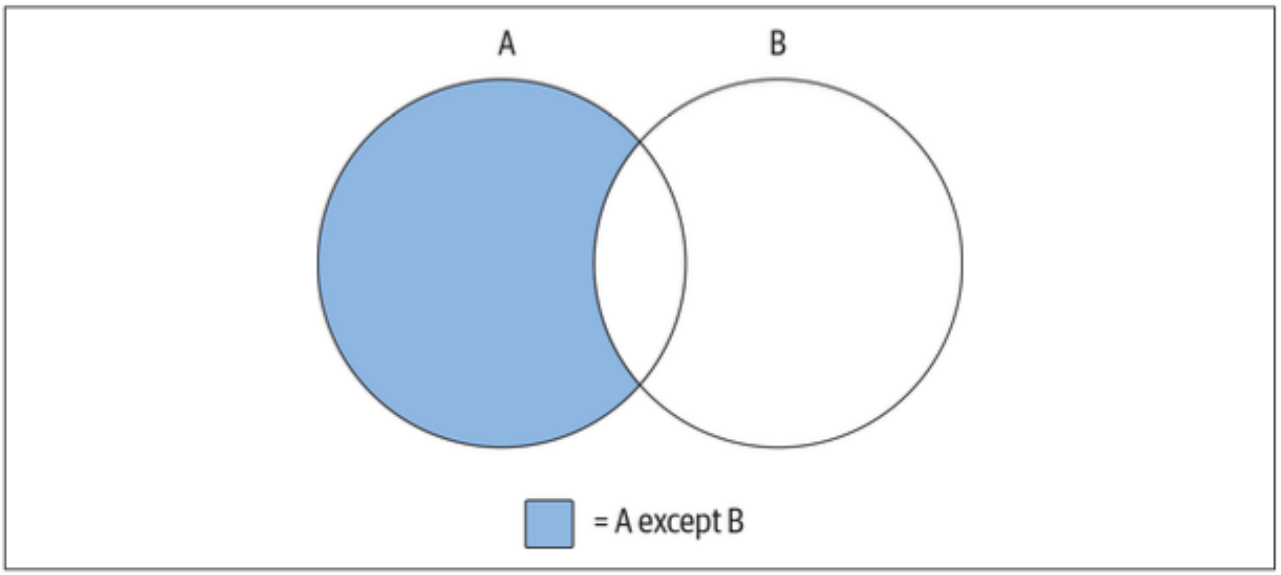
Bir veritabanındaki verilerle her seferinde satırlarla etkileşime girebilseniz de, ilişkisel veritabanları aslında tamamen kümelerle ilgilidir. Bu bölüm, çeşitli küme operatörlerini kullanarak birden çok sonuç kümesini nasıl birleştirebileceğinizi göstermektedir.

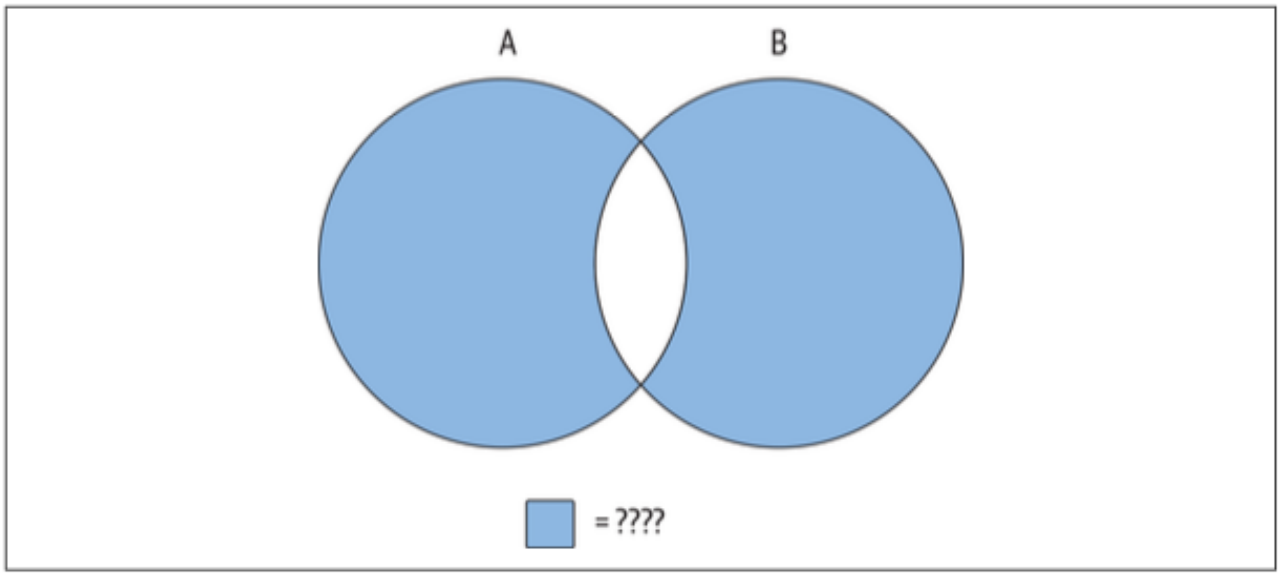
Küme Teorisi





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





Aradığınız veri kümesi, örtüşen bölge olmaksızın tüm A ve B kümelerini içerir. Bu sonucu daha önce gösterilen üç işlemden sadece biriyle elde edemezsiniz; bunun yerine, önce tüm A ve B kümelerini kapsayan bir veri kümesi oluşturmanız ve ardından çakışan bölgeyi kaldırmak için ikinci bir işlem kullanmanız gerekir. Birleşik küme A birleşimi B olarak tanımlanırsa ve örtüşen bölge A kesişen B olarak tanımlanırsa,

```
(A union B) except (A intersect B)
```

Elbette, aynı sonuçlara ulaşmanın genellikle birden fazla yolu vardır; aşağıdaki işlemi kullanarak benzer bir sonuca ulaşabilirsiniz:

```
(A except B) union (B except A)
```

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

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

mysql> desc city;
+-------------+----------------------+------+-----+-------------------+
| Field       | Type                 | Null | Key | Default           |
+-------------+----------------------+------+-----+-------------------+
| city_id     | smallint(5) unsigned | NO   | PRI | NULL              |
| city        | varchar(50)          | NO   |     | NULL              |
| country_id  | smallint(5) unsigned | NO   | MUL | NULL              |
| last_update | timestamp            | NO   |     | CURRENT_TIMESTAMP |
+-------------+----------------------+------+-----+-------------------+
```

Birleştirildiğinde, sonuç kümesindeki ilk sütun hem customer.customer_id hem de city.city_id sütunlarını içerir, ikinci sütun customer.store_id ve city.city sütunlarının birleşimini vb. Bazı sütun çiftlerinin birleştirilmesi kolay olsa da (örneğin, iki sayısal sütun), dize sütunlu sayısal bir sütun veya tarih sütunlu bir dize sütunu gibi diğer sütun çiftlerinin nasıl birleştirilmesi gerektiği açık değildir. Ek olarak, birleştirilmiş tabloların beşinci ila dokuzuncu sütunları, şehir tablosunda yalnızca dört sütun olduğundan, yalnızca müşteri tablosunun beşinci ila dokuzuncu sütunlarından gelen verileri içerecektir. Birleştirmek istediğiniz iki veri seti arasında bazı ortak noktalar olması gerekiyor.

Bu nedenle, iki veri kümesi üzerinde küme işlemleri gerçekleştirirken aşağıdaki yönergeler uygulanmalıdır:

- Her iki veri kümesi de aynı sayıda sütuna sahip olmalıdır.
- İki veri kümesindeki her sütunun veri türleri aynı olmalıdır (veya sunucu birini diğerine dönüştürebilmelidir).

Bu kurallar uygulandığında, pratikte “örtüşen verilerin” ne anlama geldiğini tasavvur etmek daha kolaydır; birleştirilen iki kümeden her sütun çifti, iki tablodaki satırların aynı sayılması için aynı dizeyi, sayıyı veya tarihi içermelidir.

Aşağıdaki şekilde gösterildiği gibi, iki select deyimi arasına bir **set** operatörü yerleştirerek bir **set** işlemi gerçekleştirirsiniz:

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

```
mysql> SELECT 1 num, 'abc' str
    -> UNION
    -> SELECT 9 num, 'xyz' str;
+-----+-----+
| num | str |
+-----+-----+
|   1 | abc |
|   9 | xyz |
+-----+-----+
2 rows in set (0.02 sec)
```

Bireysel sorguların her biri, sayısal bir sütuna ve bir dize sütununa sahip tek bir satırdan oluşan bir veri seti verir. Bu durumda birleşim olan küme operatörü, veritabanı sunucusuna iki kümedeki tüm satırları birleştirmesini söyler. Böylece, son küme iki sütunlu iki sıra içerir. Bu sorgu, birden çok bağımsız sorgu içerdiğinden bileşik sorgu olarak bilinir. Daha sonra göreceğiniz gibi, nihai sonuçlara ulaşmak için birden fazla küme işlemi gerekiyorsa, bileşik sorgular ikiden fazla sorgu içerebilir.

Küme Operatörleri

Union

union ve **union all** birden çok veri kümesini birleştirmenize olanak tanır. İkisi arasındaki fark, **union** birleşik kümeyi sıralar ve kopyaları kaldırır, **union all** tümünü birleştirerek, son veri kümesindeki satır sayısı her zaman birleştirilen kümelerdeki satır sayısının toplamına eşit olacaktır. Sunucunun çakışan verileri kontrol etmesine gerek olmadığından, bu işlem gerçekleştirilmesi en basit küme işlemidir (sunucu açısından).

```
mysql> SELECT 'CUST' typ, c.first_name, c.last_name
    -> FROM customer c
    -> UNION ALL
    -> SELECT 'ACTR' typ, a.first_name, a.last_name
    -> FROM actor a;
+------+------------+-------------+
| typ  | first_name | last_name   |
+------+------------+-------------+
| CUST | MARY       | SMITH       |
| CUST | PATRICIA   | JOHNSON     |
| CUST | LINDA      | WILLIAMS    |
| CUST | BARBARA    | JONES       |
| CUST | ELIZABETH  | BROWN       |
| CUST | JENNIFER   | DAVIS       |
```

```
| CUST | MARIA      | MILLER      |
| CUST | SUSAN      | WILSON      |
| CUST | MARGARET   | MOORE       |
| CUST | DOROTHY    | TAYLOR      |
| CUST | LISA       | ANDERSON    |
| CUST | NANCY      | THOMAS      |
| CUST | KAREN      | JACKSON     |
...
| ACTR | BURT       | TEMPLE      |
| ACTR | MERYL      | ALLEN       |
| ACTR | JAYNE      | SILVERSTONE |
| ACTR | BELA       | WALKEN      |
| ACTR | REESE      | WEST        |
| ACTR | MARY       | KEITEL      |
| ACTR | JULIA      | FAWCETT     |
| ACTR | THORA      | TEMPLE      |
+------+------------+-------------+
799 rows in set (0.00 sec)
```

Sorgu, müşteri tablosundan gelen 599 satır ve aktör tablosundan gelen diğer 200 satır ile 799 ad döndürür. **typ** takma adına sahip ilk sütun gerekli değildir, ancak sorgu tarafından döndürülen her adın kaynağını göstermek için eklenmiştir.

```
mysql> SELECT 'ACTR' typ, a.first_name, a.last_name
    -> FROM actor a
    -> UNION ALL
    -> SELECT 'ACTR' typ, a.first_name, a.last_name
    -> FROM actor a;
+------+-------------+--------------+
| typ  | first_name  | last_name    |
+------+-------------+--------------+
| ACTR | PENELOPE    | GUINESS      |
| ACTR | NICK        | WAHLBERG     |
| ACTR | ED          | CHASE        |
| ACTR | JENNIFER    | DAVIS        |
| ACTR | JOHNNY      | LOLLOBRIGIDA |
| ACTR | BETTE       | NICHOLSON    |
| ACTR | GRACE       | MOSTEL       |
...
| ACTR | BURT        | TEMPLE       |
| ACTR | MERYL       | ALLEN        |
| ACTR | JAYNE       | SILVERSTONE  |
| ACTR | BELA        | WALKEN       |
| ACTR | REESE       | WEST         |
| ACTR | MARY        | KEITEL       |
| ACTR | JULIA       | FAWCETT      |
| ACTR | THORA       | TEMPLE       |
+------+-------------+--------------+
400 rows in set (0.00 sec)
```

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

Sonuçlardan da görebileceğiniz gibi, aktör tablosundaki 200 satır, toplam 400 satır olmak üzere sonuca iki kez dahil edilmiştir.

Bir bileşik sorguda aynı sorguyu iki kez tekrarlamanız pek olası olmasa da, yinelenen verileri döndüren başka bir bileşik sorgu aşağıda verilmiştir:

```
mysql> SELECT c.first_name, c.last_name
    -> FROM customer c
    -> WHERE c.first_name LIKE 'J%' AND c.last_name LIKE 'D%'
    -> UNION ALL
    -> SELECT a.first_name, a.last_name
    -> FROM actor a
    -> WHERE a.first_name LIKE 'J%' AND a.last_name LIKE 'D%';
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| JENNIFER   | DAVIS     |
| JENNIFER   | DAVIS     |
| JUDY       | DEAN      |
| JODIE      | DEGENERES |
| JULIANNE   | DENCH     |
+------------+-----------+
5 rows in set (0.00 sec)
```

Her iki sorgu da baş harfleri J ve D olan kişilerin adlarını döndürür. Sonuç kümesindeki beş satırdan biri kopyadır (Jennifer Davis). Birleştirilmiş tablonuzun yinelenen satırları hariç tutmasını istiyorsanız, **all** yerine **union** operatörünü kullanmanız gerekir:

```
mysql> SELECT c.first_name, c.last_name
    -> FROM customer c
    -> WHERE c.first_name LIKE 'J%' AND c.last_name LIKE 'D%'
    -> UNION
    -> SELECT a.first_name, a.last_name
    -> FROM actor a
    -> WHERE a.first_name LIKE 'J%' AND a.last_name LIKE 'D%';
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| JENNIFER   | DAVIS     |
| JUDY       | DEAN      |
| JODIE      | DEGENERES |
| JULIANNE   | DENCH     |
+------------+-----------+
4 rows in set (0.00 sec)
```

intersect

ANSI SQL belirtimi, kesişimleri gerçekleştirmek için **intersect** operatörünü içerir. Ne yazık ki, MySQL'in 8.0 sürümü kesişme operatörünü uygulamıyor. Oracle veya SQL Server 2008 kullanıyorsanız, intersect'i kullanabileceksiniz; Bu kitaptaki tüm örnekler için MySQL kullandığımdan, ancak bu bölümdeki örnek sorguların sonuç kümeleri imal edilmiştir ve 8.0 sürümüne kadar olan herhangi bir sürümle çalıştırılamaz. Bir bileşik sorgudaki iki sorgu örtüşmeyen veri kümeleri döndürürse, kesişim boş bir küme olacaktır.

```
SELECT c.first_name, c.last_name
FROM customer c
WHERE c.first_name LIKE 'D%' AND c.last_name LIKE 'T%'
INTERSECT
SELECT a.first_name, a.last_name
FROM actor a
WHERE a.first_name LIKE 'D%' AND a.last_name LIKE 'T%';
Empty set (0.04 sec)
```

Baş harfleri D ve T olan hem aktörler hem de müşteriler olsa da, bu kümeler tamamen örtüşmez, bu nedenle iki kümenin kesişimi boş kümeyi verir. Ancak J ve D baş harflerine geri dönersek, kesişim tek bir satır verecektir:

```
SELECT c.first_name, c.last_name
FROM customer c
WHERE c.first_name LIKE 'J%' AND c.last_name LIKE 'D%'
INTERSECT
SELECT a.first_name, a.last_name
FROM actor a
WHERE a.first_name LIKE 'J%' AND a.last_name LIKE 'D%';
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| JENNIFER   | DAVIS     |
+------------+-----------+
1 row in set (0.00 sec)
```

except

ANSI SQL **except**, istisna işlemini gerçekleştirmek için **except** operatörünü içerir. Bir kez daha, ne yazık ki MySQL'in 8.0 sürümü, istisna operatörünü uygulamamaktadır, dolayısıyla bu bölüm için önceki bölümle aynı kurallar geçerlidir. **except** operatörü, ilk sonuç kümesinden ikinci sonuç kümesiyle olan çakışmanın çıkararak sonuç döndürür.

```
SELECT a.first_name, a.last_name
FROM actor a
WHERE a.first_name LIKE 'J%' AND a.last_name LIKE 'D%'
EXCEPT
SELECT c.first_name, c.last_name
FROM customer c
WHERE c.first_name LIKE 'J%' AND c.last_name LIKE 'D%';
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| JUDY       | DEAN      |
| JODIE      | DEGENERES |
| JULIANNE   | DENCH     |
+------------+-----------+
3 rows in set (0.00 sec)
```

Sorgunun bu sürümünde, sonuç kümesi, her iki sorgunun sonuç kümelerinde bulunan **Jennifer Davis** çıkartılır.

Sıralama

Bileşik sorgunuzun sonuçlarının sıralanmasını istiyorsanız, son sorgudan sonra y sıralama ekleyebilirsiniz. Sütun adlarını seçip sıralama yaparken, bileşik sorgunun ilk sorgusundaki sütun adlarından seçim yapmanız gerekecektir. Sıklıkla, bir bileşik sorgudaki her iki sorgu için sütun adları aynıdır.

```
mysql> SELECT a.first_name fname, a.last_name lname
    -> FROM actor a
    -> WHERE a.first_name LIKE 'J%' AND a.last_name LIKE 'D%'
    -> UNION ALL
    -> SELECT c.first_name, c.last_name
    -> FROM customer c
    -> WHERE c.first_name LIKE 'J%' AND c.last_name LIKE 'D%'
    -> ORDER BY lname, fname;
+----------+-----------+
| fname    | lname     |
+----------+-----------+
| JENNIFER | DAVIS     |
| JENNIFER | DAVIS     |
| JUDY     | DEAN      |
| JODIE    | DEGENERES |
| JULIANNE | DENCH     |
+----------+-----------+
5 rows in set (0.00 sec)
```

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