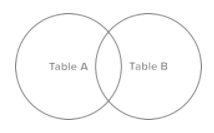
**Basic SQL Join Types**

There are four basic types of SQL joins: inner, left, right, and full. The easiest and most intuitive way to explain the difference between these four types is by using a Venn diagram, which shows all possible logical relations between data sets.

Again, it's important to stress that before you can begin using any join type, you'll need to extract the data and load it into files, where you can query tables from multiple sources. You build that process manually.

Let’s say we have two sets of data in our relational database: table A and table B, with some sort of relation specified by primary and foreign keys. The result of joining these tables together can be visually represented by the following diagram:

The extent of the overlap, if any, is determined by how many records in Table A match the records in Table B. Depending on what subset of data we would like to select from the two tables, the four join types can be visualized by highlighting the corresponding sections of the Venn diagram:



Let’s look at a manageably small example

Table A: customer\_id is the “primary key”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **customer\_id** | **first\_name** | **last\_name** | **email** | **address** | **city** | **state** | **zipcode** |
| 1 | George | Washington | gwashington@usa.gov | 3200 Mt Vernon Hwy | Mount Vernon | VA | 22121 |
| 2 | John | Adams | jadams@usa.gov | 1250 Hancock St | Quincy | MA | 02169 |
| 3 | Thomas | Jefferson | tjefferson@usa.gov | 931 Thomas Jefferson Pkwy | Charlottesville | VA | 22902 |
| 4 | James | Madison | jmadison@usa.gov | 11350 Constitution Hwy | Orange | VA | 22960 |
| 5 | James | Monroe | jmonroe@usa.gov | 2050 James Monroe Parkway | Charlottesville | VA | 22902 |

Note that (1) not every customer in our customers table has placed an order and (2) there are a few orders for which no customer record exists in our customers table.

Table B: customer\_id is the “foreign key”

order\_id = “primary key”

|  |  |  |  |
| --- | --- | --- | --- |
| **order\_id** | **order\_date** | **amount** | **customer\_id** |
| 1 | 07/04/1776 | $234.56 | 1 |
| 2 | 03/14/1760 | $78.50 | 3 |
| 3 | 05/23/1784 | $124.00 | 2 |
| 4 | 09/03/1790 | $65.50 | 3 |
| 5 | 07/21/1795 | $25.50 | 10 |
| 6 | 11/27/1787 | $14.40 | 9 |

**Inner Join:** Let’s say we wanted to get a list of those customers who placed an order and the details of the order they placed. This would be a perfect fit for an inner join, since an inner join returns records at the intersection of the two tables.

|  |  |  |
| --- | --- | --- |
| SQL | SAS | R |
| select first\_name, last\_name, order\_date, order\_amount  from customers c  inner join orders o  on c.customer\_id = c.customer\_id |  |  |

Results:

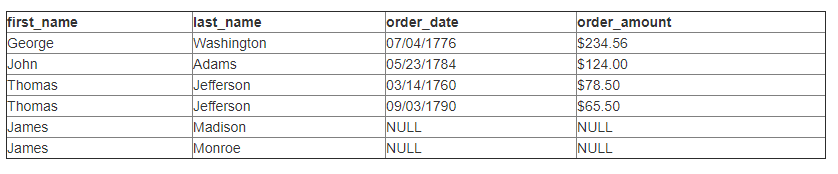
|  |  |  |  |
| --- | --- | --- | --- |
| **first\_name** | **last\_name** | **order\_date** | **order\_amount** |
| George | Washington | 07/4/1776 | $234.56 |
| John | Adams | 05/23/1784 | $124.00 |
| Thomas | Jefferson | 03/14/1760 | $78.50 |
| Thomas | Jefferson | 09/03/1790 | $65.50 |

Note that only George Washington, John Adams and Thomas Jefferson placed orders, with Thomas Jefferson placing two separate orders on 3/14/1760 and 9/03/1790.

**Left Join:** If we wanted to simply append information about orders to our customers table, regardless of whether a customer placed an order or not, we would use a left join. A left join returns all records from table A and any matching records from table B.

|  |  |  |
| --- | --- | --- |
| SQL | SAS | R |
| select first\_name, last\_name, order\_date, order\_amount  from customers c  left join orders o  on c.customer\_id = c.customer\_id |  |  |

Results:



Note that since there were no matching records for James Madison and James Monroe in our orders table, the order\_date and order\_amount are NULL, which simply means there is no data for these fields.

So why would this be useful? By simply adding a “where order\_date is NULL” line to our SQL query, it returns a list of all customers who have not placed an order:

select first\_name, last\_name, order\_date, order\_amount

from customers c

left join orders o

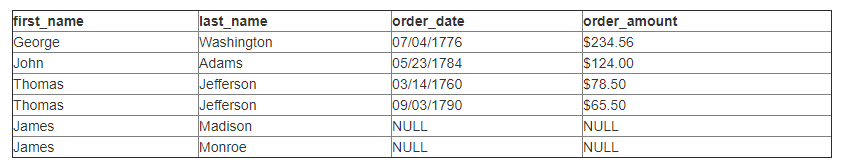
on c.customer\_id = o.customer\_id

where order\_date is NULL

**Right Join:** Right join is a mirror version of the left join and allows to get a list of all orders, appended with customer information.

|  |  |  |
| --- | --- | --- |
| SQL | SAS | R |
| select first\_name, last\_name, order\_date, order\_amount  from customers c  right join orders o  on c.customer\_id = c.customer\_id |  |  |

Results:



Note that since there were no matching customer records for orders placed in 1795 and 1787, the first\_name and last\_name fields are NULL in the resulting set.

Also note that the order in which the tables are joined is important. We are right joining the orders table to the customers table. If we were to right join the customers table to the orders table, the result would be the same as left joining the orders table to the customers table.

Why is this useful? Simply adding a “where first\_name is NULL” line to our SQL query returns a list of all orders for which we failed to record information about the customers who placed them:

select first\_name, last\_name, order\_date, order\_amount

from customers c

right join orders o

on c.customer\_id = o.customer\_id

where first\_name is NULL

**Full Join:** Finally, for a list of all records from both tables, we can use a full join.

|  |  |  |
| --- | --- | --- |
| SQL | SAS | R |
| select first\_name, last\_name, order\_date, order\_amount  from customers c  full join orders o  on c.customer\_id = o.customer\_id |  |  |

Results:

