# Relational Data

David Gerard

2023-10-19

## Learning Objectives

- What is relational data.
- `inner_join()`, `left_join()`, `right_join()`, `full_join()`, `semi_join()`, `anti_join()`.
- SQL.
- Chapter 13 of RDS.
- Data Transformation Cheatsheet.

## Relational Data

- Load the tidyverse

  ```r
  library(tidyverse)
  ```

- Many datasets have more than two data frames.
- These data frames are often connected (rows in one correspond to rows in another)
- Consider the data in the nycflights13 package.

  ```r
  library(nycflights13)
  ```

  - `airlines`: Airline names.

    ```r
    data("airlines")
    head(airlines)
    ```

    ```
    ## # A tibble: 6 x 2
    ##   carrier name
    ##   <chr>   <chr>
    ## 1 9E      Endeavor Air Inc.
    ## 2 AA      American Airlines Inc.
    ## 3 AS      Alaska Airlines Inc.
    ## 4 B6      JetBlue Airways
    ## 5 DL      Delta Air Lines Inc.
    ## 6 EV      ExpressJet Airlines Inc.
    ```

  - `airports`: Airport metadata

    ```r
    data("airports")
    head(airports)
    ```

    ```
    ## # A tibble: 6 x 8
    ##   faa   name                             lat   lon   alt    tz dst   tzone
    ##   <chr> <chr>                          <dbl> <dbl> <dbl> <dbl> <chr> <chr>
    ## 1 04G   Lansdowne Airport               41.1 -80.6  1044    -5 A     America/Ne~
    ```

```
## 2 06A   Moton Field Municipal Airport   32.5 -85.7   264    -6 A     America/Ch~
## 3 06C   Schaumburg Regional             42.0 -88.1   801    -6 A     America/Ch~
## 4 06N   Randall Airport                 41.4 -74.4   523    -5 A     America/Ne~
## 5 09J   Jekyll Island Airport           31.1 -81.4    11    -5 A     America/Ne~
## 6 0A9   Elizabethton Municipal Airport  36.4 -82.2  1593    -5 A     America/Ne~
```

- `planes`: Plane metadata.

```r
data("planes")
head(planes)
```

```
## # A tibble: 6 x 9
##   tailnum  year type               manufacturer model engines seats speed engine
##   <chr>   <int> <chr>              <chr>        <chr>   <int> <int> <int> <chr>
## 1 N10156   2004 Fixed wing multi ~ EMBRAER      EMB-~       2    55    NA Turbo~
## 2 N102UW   1998 Fixed wing multi ~ AIRBUS INDU~ A320~       2   182    NA Turbo~
## 3 N103US   1999 Fixed wing multi ~ AIRBUS INDU~ A320~       2   182    NA Turbo~
## 4 N104UW   1999 Fixed wing multi ~ AIRBUS INDU~ A320~       2   182    NA Turbo~
## 5 N10575   2002 Fixed wing multi ~ EMBRAER      EMB-~       2    55    NA Turbo~
## 6 N105UW   1999 Fixed wing multi ~ AIRBUS INDU~ A320~       2   182    NA Turbo~
```

- `weather`: Hourly weather data

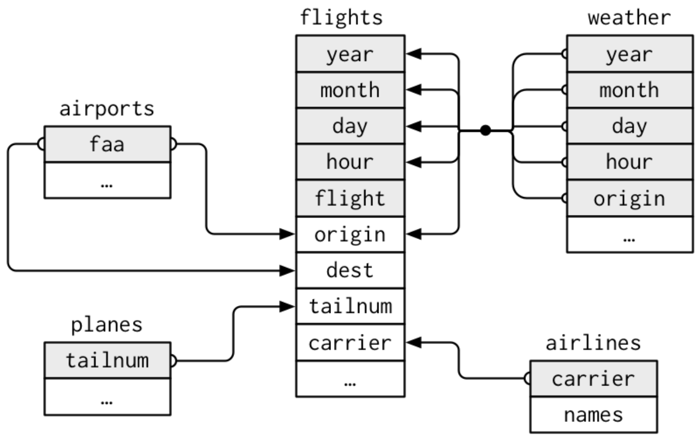
```r
data("weather")
head(weather)
```

```
## # A tibble: 6 x 15
##   origin  year month   day  hour  temp  dewp humid wind_dir wind_speed wind_gust
##   <chr>  <int> <int> <int> <int> <dbl> <dbl> <dbl>    <dbl>      <dbl>     <dbl>
## 1 EWR     2013     1     1     1  39.0  26.1  59.4      270      10.4         NA
## 2 EWR     2013     1     1     2  39.0  27.0  61.6      250       8.06        NA
## 3 EWR     2013     1     1     3  39.0  28.0  64.4      240      11.5         NA
## 4 EWR     2013     1     1     4  39.9  28.0  62.2      250      12.7         NA
## 5 EWR     2013     1     1     5  39.0  28.0  64.4      260      12.7         NA
## 6 EWR     2013     1     1     6  37.9  28.0  67.2      240      11.5         NA
## # i 4 more variables: precip <dbl>, pressure <dbl>, visib <dbl>,
## #   time_hour <dttm>
```

- `flights`: Flights data

```r
data("flights")
head(flights)
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```




- For nycflights13:
  - `flights` connects to `planes` via a single variable, `tailnum`.
  - `flights` connects to `airlines` through the `carrier` variable.
  - `flights` connects to `airports` in two ways: via the `origin` and `dest` variables.
  - `flights` connects to `weather` via `origin` (the location), and `year`, `month`, `day` and `hour` (the time).
- Variables used to connect a pair of data frames are called **keys**.
- **Primary key**: Identifies rows in its own table.
- **Foreign key**: Identifies rows in another table.
- *Example*: `planes$tailnum` is a primary key because it uniquely identifies rows in `planes`.

```
planes %>%
  group_by(tailnum) %>%
  count() %>%
  filter(n > 1)
```

```
## # A tibble: 0 x 2
## # Groups:   tailnum [0]
## # i 2 variables: tailnum <chr>, n <int>
```

- *Example*: `flights$tailnum` is a foreign key because it uniquely identifies rows in `planes`. There are multiple rows with the same `tailnum` in `flights`, so `flights$tailnum` is *not* a primary key.

```
flights %>%
  group_by(tailnum) %>%
  count() %>%
  filter(n > 1)
```

```
## # A tibble: 3,873 x 2
## # Groups:   tailnum [3,873]
##    tailnum     n
##    <chr>   <int>
##  1 D942DN      4
##  2 N0EGMQ    371
##  3 N10156    153
##  4 N102UW     48
##  5 N103US     46
##  6 N104UW     47
```

```
##   7 N10575    289
##   8 N105UW     45
##   9 N107US     41
##  10 N108UW     60
## # i 3,863 more rows
```

- *Example*: `weather$origin` is *part* of the primary key for `weather` (along with `year`, `month`, `day`, and `hour`) and a foreign key for `airports` (`weather$origin` is connected to `airports$faa`).
- If a table lacks a primary key (like `flights`) then you can add one with `mutate()` and `row_number()`.

```r
flights %>%
  mutate(row = row_number()) %>%
  select(row, everything())
```

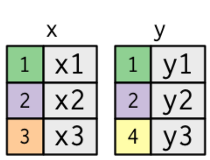
```
## # A tibble: 336,776 x 20
##      row  year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1     1  2013     1     1      517            515         2      830
##  2     2  2013     1     1      533            529         4      850
##  3     3  2013     1     1      542            540         2      923
##  4     4  2013     1     1      544            545        -1     1004
##  5     5  2013     1     1      554            600        -6      812
##  6     6  2013     1     1      554            558        -4      740
##  7     7  2013     1     1      555            600        -5      913
##  8     8  2013     1     1      557            600        -3      709
##  9     9  2013     1     1      557            600        -3      838
## 10    10  2013     1     1      558            600        -2      753
## # i 336,766 more rows
## # i 12 more variables: sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

- **Exercise** (RDS 13.3.1.2): Identify the primary keys in the following data frames.
  - `Lahman::Batting`,
  - `babynames::babynames`,
  - `nasaweather::atmos`,
  - `fueleconomy::vehicles`,
  - `ggplot2::diamonds`.

  (You might need to install some packages and read some documentation.)

# Join Set-Up

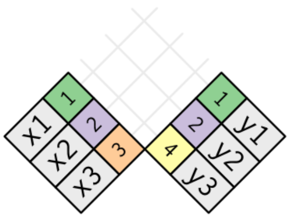
- Suppose we have the following two data frames



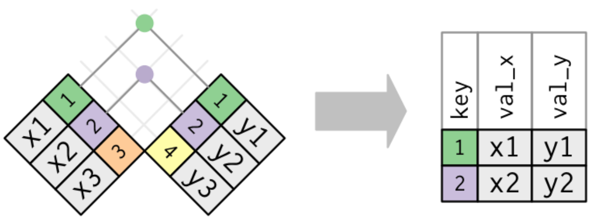

```r
x <- tribble(~key, ~val_x,
             #---  ------
             1,    "x1",
```

```
             2,    "x2",
             3,    "x3")
y <- tribble(~key, ~val_y,
             #---  ------
             1,    "y1",
             2,    "y2",
             4,    "y3")
```

- A join connects rows of `x` to rows of `y`.

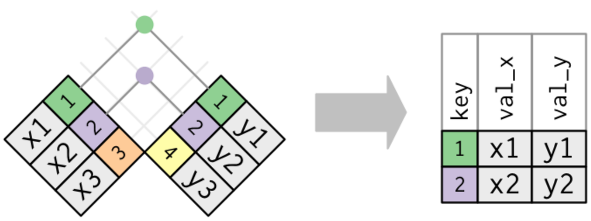



- E.g. match row `1` of `x` with row `1` of `y`, and row `2` of `x` with row `2` of `y`.




# Inner Join

- `inner_join(x, y)` matches the rows of `x` with rows of `y` only when their keys are equal.



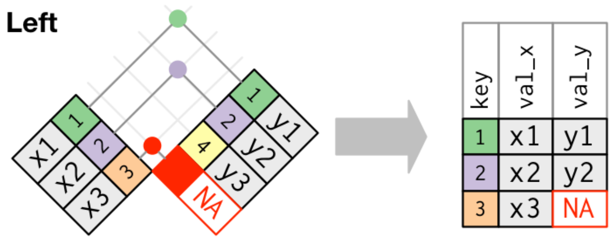

```
inner_join(x, y, by = join_by(key))
```

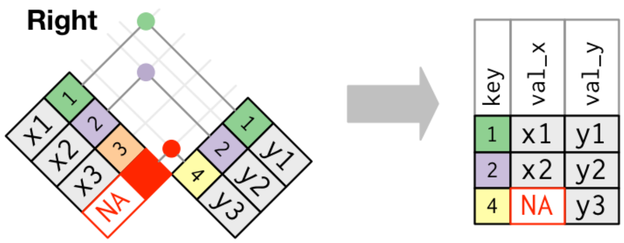
```
## # A tibble: 2 x 3
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
```

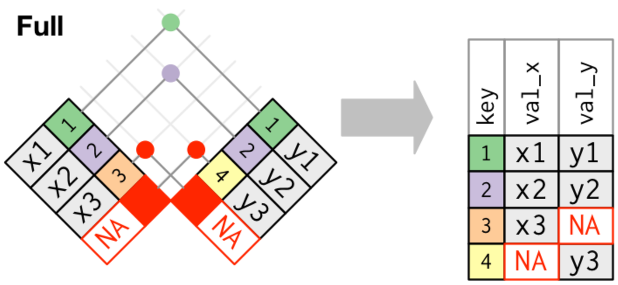
- Put the key you are joining by inside `join_by()`.
- Keeps all rows that appear in *both* data frames.
- **Exercise**: Select all flights that use a plane where you have some annotation.

# Outer Join

- Keeps all rows that appear in *at least one* data frame.




- `left_join(x, y)` keeps all rows of `x`.

```
left_join(x, y, by = join_by(key))
```

```
## # A tibble: 3 x 3
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
```

- `left_join()` is by far the most common joiner, and you should always use this unless you have a good reason not to.

- `right_join(x, y)` keeps all rows of `y`.

```
right_join(x, y, by = join_by(key))
```

```
## # A tibble: 3 x 3
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     4 <NA>  y3
```

- `full_join(x, y)` keeps all rows of both.

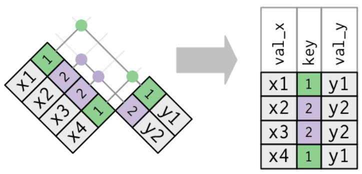
```r
full_join(x, y, by = join_by(key))
```

```
## # A tibble: 4 x 3
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
## 4     4 <NA>  y3
```

- **Exercise**: Add the full airline names to the `flights` data frame.

## Duplicate Keys

- If you have duplicate keys in one table, then the rows from the data frame where there is no duplication are copied multiple times in the new data frame.



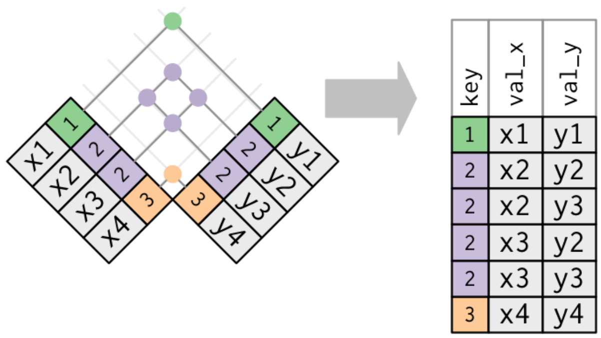

(useful for adding summary data to a table)

```r
x_mult <- tribble(~key, ~val_x,
                  ##--  ------
                  1,    "x1",
                  2,    "x2",
                  2,    "x3",
                  1,    "x4")

left_join(x_mult, y, by = join_by(key))
```

```
## # A tibble: 4 x 3
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     2 x3    y2
## 4     1 x4    y1
```

- If you have duplicate keys in both (usually a mistake), then you get every possible combination of the values in x and y at the key values where there are duplications. You'll get a warning about this.




```r
y_mult <- tribble(~key, ~val_y,
                  ##--  ------
                  1,    "y1",
                  2,    "y2",
                  2,    "y3",
                  1,    "y4")

left_join(x_mult, y_mult, by = join_by(key))
```

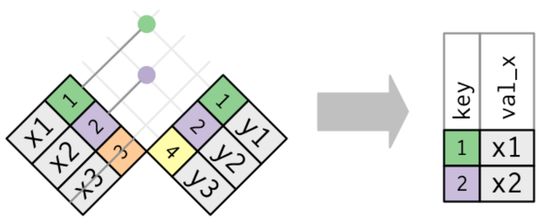
```
## Warning in left_join(x_mult, y_mult, by = join_by(key)): Detected an unexpected many-to-many rel
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 2 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
## # A tibble: 8 x 3
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     1 x1    y4
## 3     2 x2    y2
## 4     2 x2    y3
## 5     2 x3    y2
## 6     2 x3    y3
## 7     1 x4    y1
## 8     1 x4    y4
```

- **Exercise**: In the previous two exercises, we had some duplicate keys. For each exercise, which data frame had the duplicate keys?
- **Exercise**: Is there a relationship between the age of a plane and its delays?

# Filtering Joins

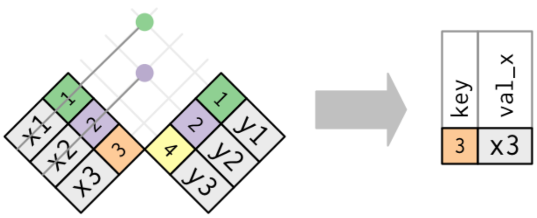
- `semi_join()` *keeps* all of the rows in `x` that have a match in `y` (but don't add the variables of `y` to `x`).



```r
semi_join(x, y, by = join_by(key))
```

```
## # A tibble: 2 x 2
##     key val_x
##   <dbl> <chr>
## 1     1 x1
## 2     2 x2
```

- `anti_join()` *drops* all of the rows in `x` that have a match in `y` (but don't add the variables of `y` to `x`).




```r
anti_join(x, y, by = join_by(key))
```

```
## # A tibble: 1 x 2
##     key val_x
##   <dbl> <chr>
## 1     3 x3
```

- **Exercise**: Find the 10 days of the year that have the highest median departure delay, then select all flights from those 10 days.

# Other Key Names

- If the primary and foreign keys do not match, you need to specify that using a logical condition inside `join_by()`. E.g. `join_by(a == b)`, where `a` is the key in `x` and `b` is the key in `y`.

```r
left_join(flights, airports, by = join_by(origin == faa))
```

```
## # A tibble: 336,776 x 26
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 18 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, name <chr>, lat <dbl>,
## #   lon <dbl>, alt <dbl>, tz <dbl>, dst <chr>, tzone <chr>
```

- If you have multiple variables acting as the key, you just add those arguments in `join_by()`.

```
left_join(flights, weather, by = join_by(origin, year, month, day, hour))
```

```
## # A tibble: 336,776 x 29
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 21 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour.x <dttm>, temp <dbl>, dewp <dbl>,
## #   humid <dbl>, wind_dir <dbl>, wind_speed <dbl>, wind_gust <dbl>,
## #   precip <dbl>, pressure <dbl>, visib <dbl>, time_hour.y <dttm>
```