ST720 Data Science

Relational Data

Seung Jun Shin (sjshin@krea.ac.kr)

Department of Statistics, Korea University

Introduction

- Multiple tables of data are called **relational data** becacuse it is the reltions, not just the individual datasets, that are important.
- The most common place to find relational data is in a *relational* databases management system (or RDBMS), a term that encompasses almost all modern databases.
- Three families of verbs that work with pairs of tables:
  - **Mutating joins**: add new variables by matching observations
  - **Filtering joins**: filter observations based on the other table
  - **Set operations**: treat observations as set elements

nycflights13

- nycflights13 contains four tibbles (airlines, airports, planes, weather) that are related to the flights table

```
flights  %>% print(n = 5)
```

```
## # A tibble: 336,776 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_tim
##   <int> <int> <int>    <int>          <int>     <dbl>    <int
## 1  2013     1     1      517            515         2      83
## 2  2013     1     1      533            529         4      85
## 3  2013     1     1      542            540         2      92
## 4  2013     1     1      544            545        -1     100
## 5  2013     1     1      554            600        -6      81
## # ... with 3.368e+05 more rows, and 12 more variables:
## #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, fli
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm
```

nycflights13

- `airlines` lets you look up the full carrier name from its abbreviated code:

```
airlines  %>% print(n = 5)
```

```
## # A tibble: 16 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.
## # ... with 11 more rows
```

nycflights13

- `airports` gives information about each airport, identified by the `faa` airport code:

```
airports %>% print(n = 5)
```

```
## # A tibble: 1,458 x 8
##   faa   name                           lat   lon   alt    tz dst
##   <chr> <chr>                        <dbl> <dbl> <int> <dbl> <chr>
## 1 04G   Lansdowne Airport             41.1 -80.6  1044    -5 A
## 2 06A   Moton Field Municipal ~       32.5 -85.7   264    -6 A
## 3 06C   Schaumburg Regional           42.0 -88.1   801    -6 A
## 4 06N   Randall Airport               41.4 -74.4   523    -5 A
## 5 09J   Jekyll Island Airport        31.1 -81.4    11    -5 A
## # ... with 1,453 more rows
```

nycflights13

- `planes` gives information about each plane, identified by its tailnum:

```
planes %>% print(n = 5)
```

```
## # A tibble: 3,322 x 9
##   tailnum  year type       manufacturer   model  engines seat
##   <chr>   <int> <chr>      <chr>          <chr>    <int> <int
## 1 N10156   2004 Fixed win~ EMBRAER        EMB-1~       2    5
## 2 N102UW   1998 Fixed win~ AIRBUS INDUST~ A320-~       2   18
## 3 N103US   1999 Fixed win~ AIRBUS INDUST~ A320-~       2   18
## 4 N104UW   1999 Fixed win~ AIRBUS INDUST~ A320-~       2   18
## 5 N10575   2002 Fixed win~ EMBRAER        EMB-1~       2    5
## # ... with 3,317 more rows
```

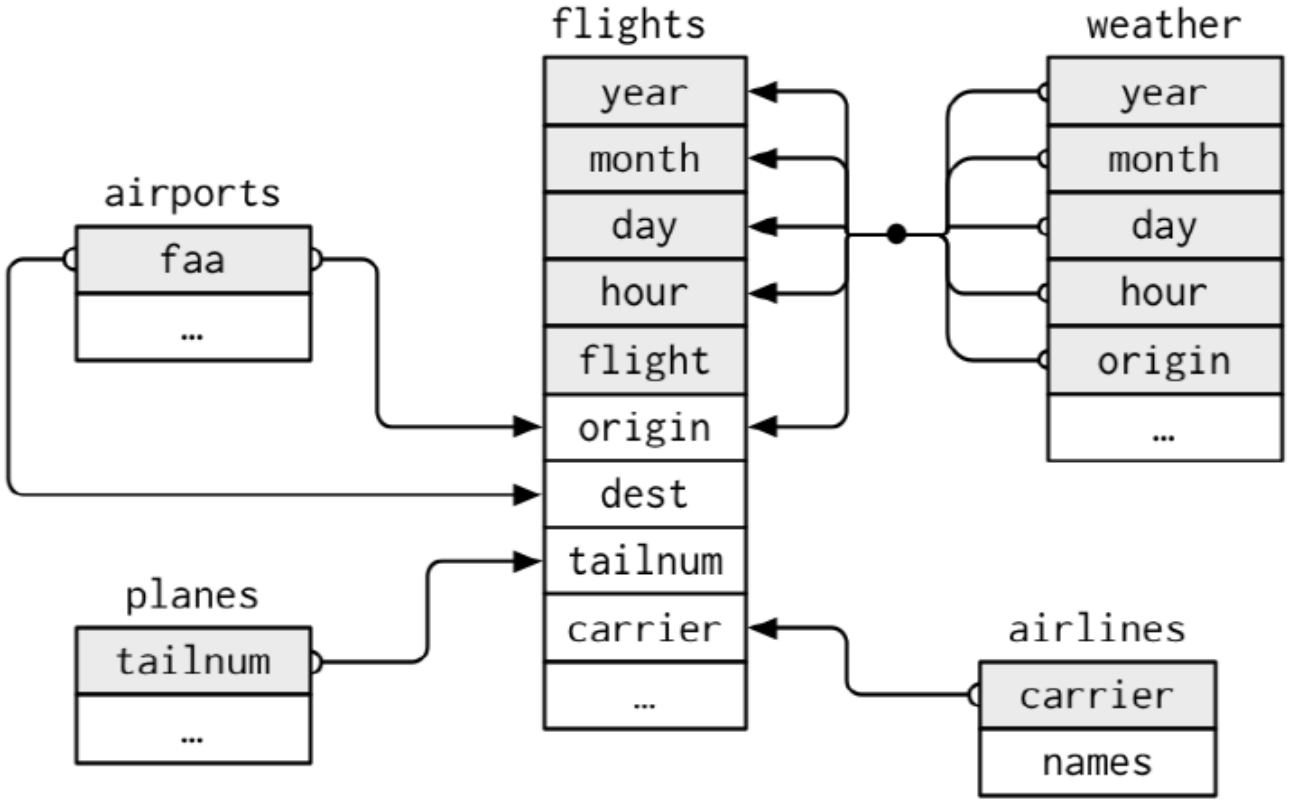
nycflights13

- `weather` gives the weather at each NYC airport for each hour:

```
weather %>% print(n = 5)
```

```
## # A tibble: 26,115 x 15
##   origin  year month   day  hour  temp  dewp humid wind_dir w
##   <chr>  <dbl> <dbl> <int> <int> <dbl> <dbl> <dbl>    <dbl>
## 1 EWR     2013     1     1     1  39.0  26.1  59.4      270
## 2 EWR     2013     1     1     2  39.0  27.0  61.6      250
## 3 EWR     2013     1     1     3  39.0  28.0  64.4      240
## 4 EWR     2013     1     1     4  39.9  28.0  62.2      250
## 5 EWR     2013     1     1     5  39.0  28.0  64.4      260
## # ... with 2.611e+04 more rows, and 5 more variables: wind_gu
## #   precip <dbl>, pressure <dbl>, visib <dbl>, time_hour <dtt
```

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).

Keys

- Variable(s) that connect each pair of tables and uniquely identify an observation.
- **primary key**: (in its own table)

ex) `planes$tailnum` uniquely identifies each plane in the `planes` table.

- **foreign key**: (in another table)

ex) `flights$tailnum` matches each flight to a unique plane.

- A variable can be both a primary key and a forreign key.

Keys

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

```
## # A tibble: 0 x 2
## # ... with 2 variables: tailnum <chr>, n <int>
```

```
weather %>%
  count(year, month, day, hour, origin) %>%
  filter(n > 1)
```

```
## # A tibble: 3 x 6
##    year month   day  hour origin     n
##   <dbl> <dbl> <int> <int> <chr>  <int>
## 1  2013    11     3     1 EWR        2
## 2  2013    11     3     1 JFK        2
## 3  2013    11     3     1 LGA        2
```

Keys

```
flights %>%
  count(year, month, day, flight) %>%
  filter(n > 1)
```

```
## # A tibble: 29,768 x 5
##     year month   day flight     n
##    <int> <int> <int>  <int> <int>
##  1  2013     1     1      1     2
##  2  2013     1     1      3     2
##  3  2013     1     1      4     2
##  4  2013     1     1     11     3
##  5  2013     1     1     15     2
##  6  2013     1     1     21     2
##  7  2013     1     1     27     4
##  8  2013     1     1     31     2
##  9  2013     1     1     32     2
## 10  2013     1     1     35     2
## # ... with 29,758 more rows
```

Keys

```
flights %>%
  count(year, month, day, tailnum, flight) %>%
  filter(n > 1)
```

```
## # A tibble: 11 x 6
##     year month   day tailnum flight     n
##    <int> <int> <int> <chr>    <int> <int>
##  1  2013     2     9 <NA>       303     2
##  2  2013     2     9 <NA>       655     2
##  3  2013     2     9 <NA>      1623     2
##  4  2013     6     8 N487WN    2269     2
##  5  2013     6    15 N230WN    2269     2
##  6  2013     6    22 N440LV    2269     2
##  7  2013     6    29 N707SA    2269     2
##  8  2013     7     6 N259WN    2269     2
##  9  2013     8     3 N446WN    2269     2
## 10  2013     8    10 N478WN    2269     2
## 11  2013    12    15 <NA>       398     2
```

Mutating joins

- combine variables from two tables
  1. match obesrvations by their keys
  2. copy across variables from one table to the other.

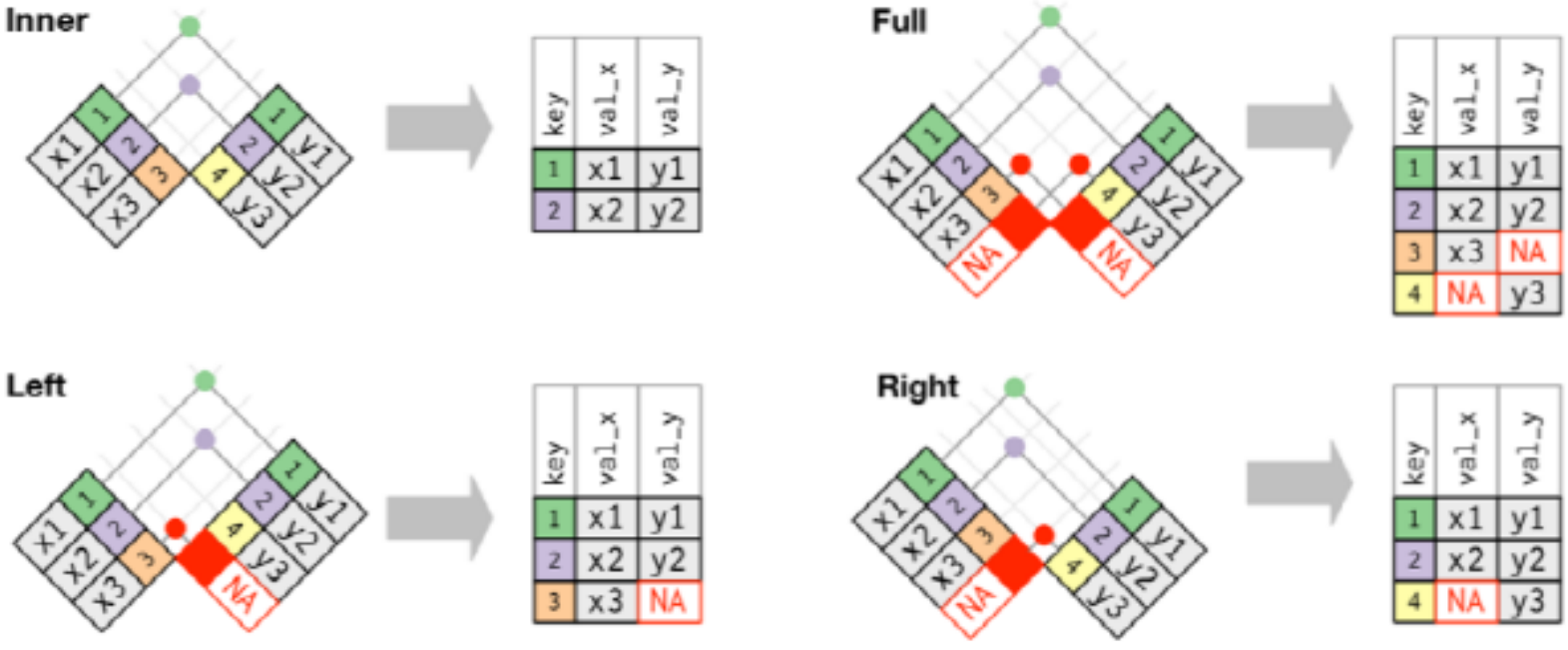
```
(x <- tribble( ~key, ~val_x,
                  1, "x1",
                  2, "x2",
                  3, "x3" ))
```

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

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

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

Mutating joins



- **inner join** excludes unmatched rows.
- **left join** keeps all observations in `x`.
- **right join** keeps all observations in `y`.
- **full join** keeps all observations in `x` and `y`.

Mutating joins

```
x %>%
  inner_join(y, by = "key")
```

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

```
x %>%
  left_join(y, 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>
```

```
x %>%
  full_join(y, by = "key")
```

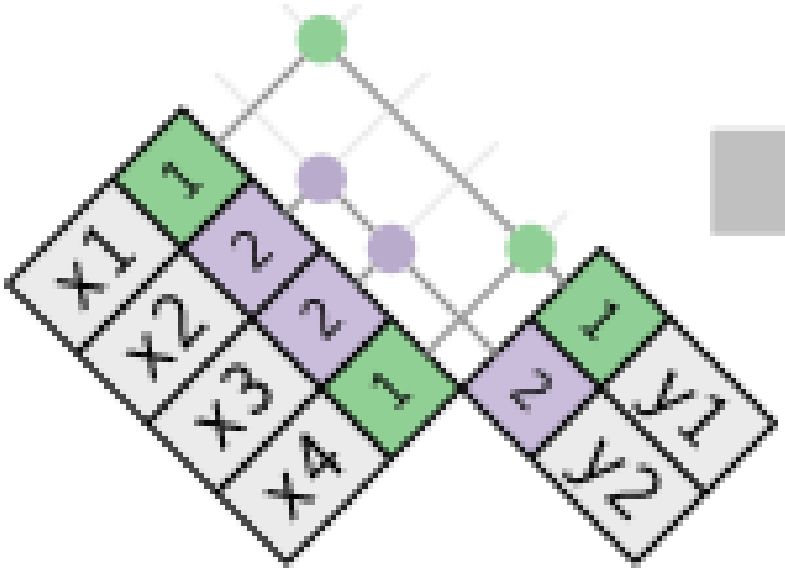
```
## # A tibble: 4 x 3
##     key val_x val_y
##   <dbl> <chr> <chr>
## 1     1 x1    y1
## 2     2 x2    y2
## 3     3 x3    <NA>
## 4     4 <NA>  y3
```

```
x %>%
  right_join(y, 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
```

Duplecate Keys

- One table has duplicate keys (one to many)



val_x	key	val_y
x1	1	y1
x2	2	y2
x3	2	y2
x4	1	y1

Duplecate Keys

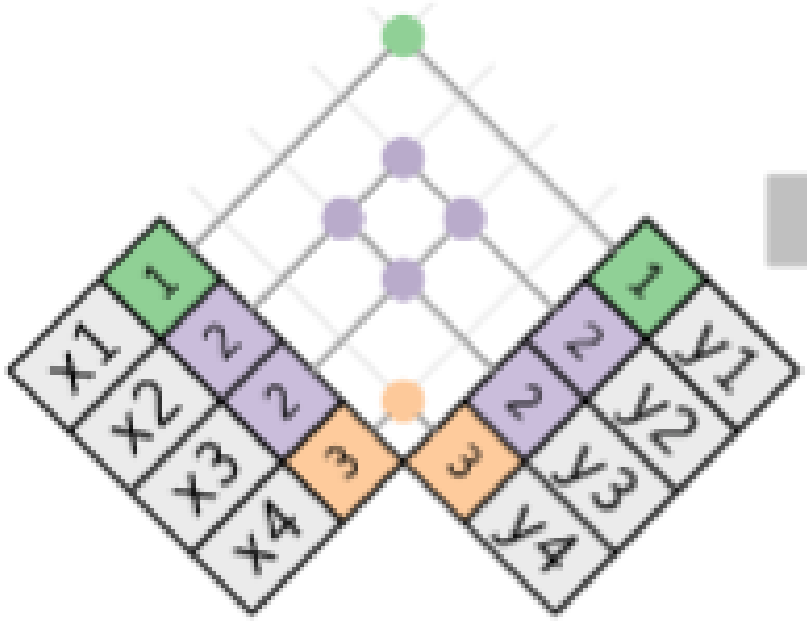
```
x <- tribble(
  ~key, ~val_x,
  1, "x1",
  2, "x2",
  2, "x3",
  1, "x4"
)
y <- tribble(
  ~key, ~val_y,
  1, "y1",
  2, "y2"
)
```

```
left_join(x, y, 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
```

Duplicate Keys

- Both tables have duplicate keys (many to many)
- Gives all the combinations.



key	val_x	val_y
1	x1	y1
2	x2	y2
2	x2	y3
2	x3	y2
2	x3	y3
3	x4	y4

Duplicate Keys

```
x <- tribble(
  ~key, ~val_x,
  1, "x1",
  2, "x2",
  2, "x3",
  1, "x4"
)
y <- tribble(
  ~key, ~val_y,
  1, "y1",
  2, "y2",
  2, "y3",
  3, "y4"
)
```

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

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

Defining Key Columns

- Pairs of tables have always been joined by a single variable, and that variable ahs the same name in both tables.
- You can use other values for `by` to connect the tables in other ways.
  - `by = NULL` uses all variables that apprear in both tables. (natral join)
  - `by = "x"` specifies the names of variables.
  - `by c("a" = "b")` matches variable `a` in table `x` to variable `b` in table `y`. Variable names in `x` is used to report.

Defining Key Columns

```
flights2 <- flights %>%
  select(year:day, hour, origin, dest, tailnum, carrier)
flights2 %>% print(n = 5)
```

```
## # A tibble: 336,776 x 8
##    year month   day  hour origin dest  tailnum carrier
##   <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>
## 1  2013     1     1     5 EWR    IAH   N14228  UA
## 2  2013     1     1     5 LGA    IAH   N24211  UA
## 3  2013     1     1     5 JFK    MIA   N619AA  AA
## 4  2013     1     1     5 JFK    BQN   N804JB  B6
## 5  2013     1     1     6 LGA    ATL   N668DN  DL
## # ... with 3.368e+05 more rows
```

Defining Key Columns

```
flights2 %>%
  left_join(weather) %>%
  print(n = 5)
```

```
## Joining, by = c("year", "month", "day", "hour", "origin")
```

```
## # A tibble: 336,776 x 18
##    year month   day  hour origin dest  tailnum carrier  temp
##   <dbl> <dbl> <int> <dbl> <chr>  <chr> <chr>   <chr>   <dbl>
## 1  2013     1     1     5 EWR    IAH   N14228  UA       39.0
## 2  2013     1     1     5 LGA    IAH   N24211  UA       39.9
## 3  2013     1     1     5 JFK    MIA   N619AA  AA       39.0
## 4  2013     1     1     5 JFK    BQN   N804JB  B6       39.0
## 5  2013     1     1     6 LGA    ATL   N668DN  DL       39.9
## # ... with 3.368e+05 more rows, and 7 more variables: wind_di
## #   wind_speed <dbl>, wind_gust <dbl>, precip <dbl>, pressure
## #   visib <dbl>, time_hour <dttm>
```

Defining Key Columns

```
flights2 %>%
  left_join(planes, by = "tailnum") %>%
  print(n = 5)
```

```
## # A tibble: 336,776 x 16
##   year.x month   day  hour origin dest  tailnum carrier year.
##    <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>    <int
## 1   2013     1     1     5 EWR    IAH   N14228  UA         199
## 2   2013     1     1     5 LGA    IAH   N24211  UA         199
## 3   2013     1     1     5 JFK    MIA   N619AA  AA         199
## 4   2013     1     1     5 JFK    BQN   N804JB  B6         201
## 5   2013     1     1     6 LGA    ATL   N668DN  DL         199
## # ... with 3.368e+05 more rows, and 6 more variables: manufac
## #   model <chr>, engines <int>, seats <int>, speed <int>, eng
```

Defining Key Columns

```
flights2 %>%
  left_join(airports, c("dest" = "faa")) %>%
  print(n = 5)
```

```
## # A tibble: 336,776 x 15
##    year month   day  hour origin dest  tailnum carrier name
##   <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <chr>
## 1  2013     1     1     5 EWR    IAH   N14228  UA      Geor~
## 2  2013     1     1     5 LGA    IAH   N24211  UA      Geor~
## 3  2013     1     1     5 JFK    MIA   N619AA  AA      Miam~
## 4  2013     1     1     5 JFK    BQN   N804JB  B6      <NA>
## 5  2013     1     1     6 LGA    ATL   N668DN  DL      Hart~
## # ... with 3.368e+05 more rows, and 4 more variables: alt <in
## #   dst <chr>, tzone <chr>
```

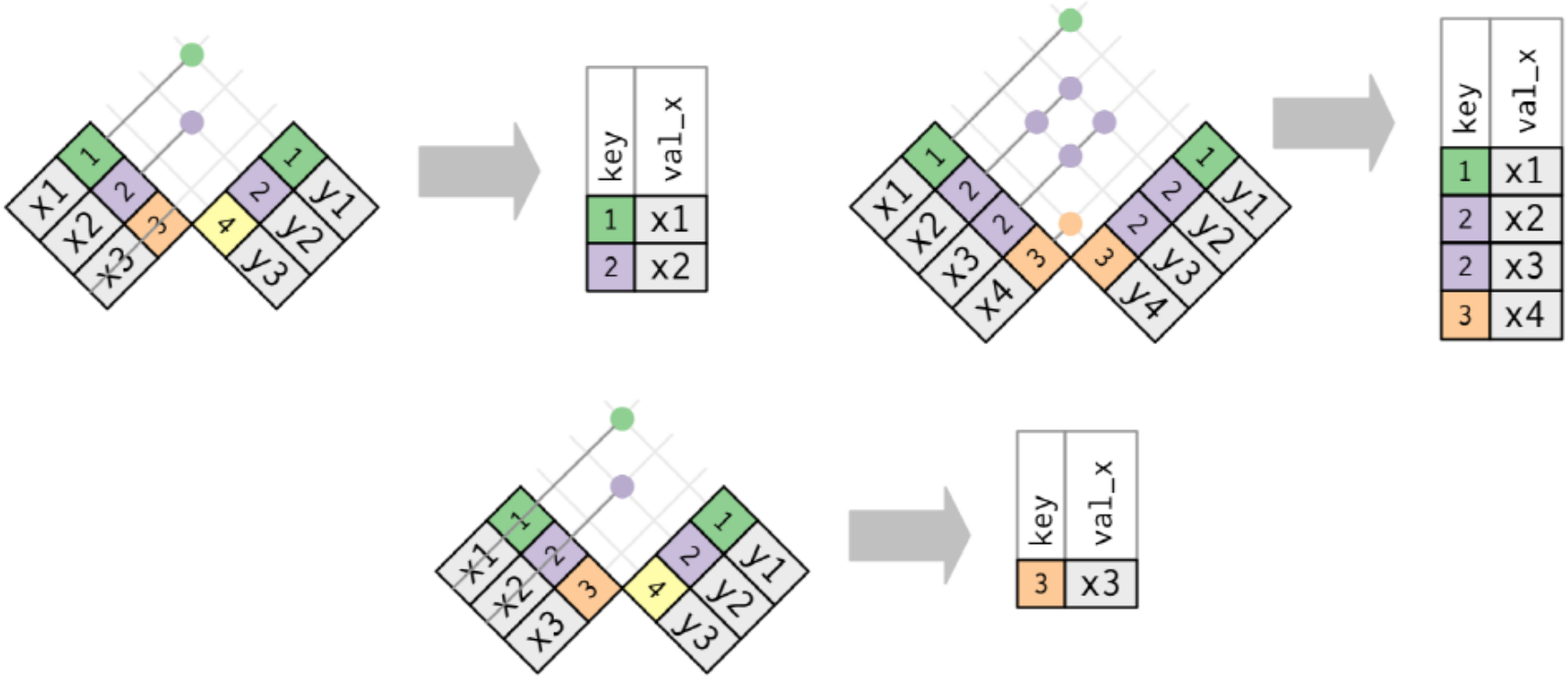
Defining Key Columns

```
flights2 %>%
  left_join(airports, c("origin" = "faa")) %>%
  print(n = 5)
```

```
## # A tibble: 336,776 x 15
##    year month   day  hour origin dest  tailnum carrier name
##   <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <chr>
## 1  2013     1     1     5 EWR    IAH   N14228  UA      Newa~
## 2  2013     1     1     5 LGA    IAH   N24211  UA      La G~
## 3  2013     1     1     5 JFK    MIA   N619AA  AA      John~
## 4  2013     1     1     5 JFK    BQN   N804JB  B6      John~
## 5  2013     1     1     6 LGA    ATL   N668DN  DL      La G~
## # ... with 3.368e+05 more rows, and 4 more variables: alt <in
## #   dst <chr>, tzone <chr>
```

Filtering joins

- filtering joins match observations in the same way as mutating joins, but affect the observations, not the variables.
- `semi_join(x, y)` **keeps** all obs in `x` that have a match in `y`.



Filtering joins

```
top_dest <- flights %>%
  count(dest, sort = TRUE) %>%
  head(10)
top_dest
```

```
## # A tibble: 10 x 2
##    dest      n
##    <chr> <int>
##  1 ORD   17283
##  2 ATL   17215
##  3 LAX   16174
##  4 BOS   15508
##  5 MCO   14082
##  6 CLT   14064
##  7 SFO   13331
##  8 FLL   12055
##  9 MIA   11728
## 10 DCA    9705
```

Filtering joins

```
flights %>%
  filter(dest %in% top_dest$dest) %>%
  print(n = 5)
```

```
## # A tibble: 141,145 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_tim
##   <int> <int> <int>    <int>          <int>     <dbl>    <int
## 1  2013     1     1      542            540         2      92
## 2  2013     1     1      554            600        -6      81
## 3  2013     1     1      554            558        -4      74
## 4  2013     1     1      555            600        -5      91
## 5  2013     1     1      557            600        -3      83
## # ... with 1.411e+05 more rows, and 12 more variables:
## #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, fli
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm
```

Filtering joins

```
flights %>%
  semi_join(top_dest) %>%
  print(n = 5)
```

```
## Joining, by = "dest"
```

```
## # A tibble: 141,145 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_tim
##   <int> <int> <int>    <int>          <int>     <dbl>    <int
## 1  2013     1     1      542            540         2      92
## 2  2013     1     1      554            600        -6      81
## 3  2013     1     1      554            558        -4      74
## 4  2013     1     1      555            600        -5      91
## 5  2013     1     1      557            600        -3      83
## # ... with 1.411e+05 more rows, and 12 more variables:
## #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, fli
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm
```

Filtering joins

```
flights %>%
  anti_join(planes, by = "tailnum") %>%
  count(tailnum, sort = TRUE)
```

```
## # A tibble: 722 x 2
##    tailnum     n
##    <chr>   <int>
##  1 <NA>     2512
##  2 N725MQ    575
##  3 N722MQ    513
##  4 N723MQ    507
##  5 N713MQ    483
##  6 N735MQ    396
##  7 NOEGMQ    371
##  8 N534MQ    364
##  9 N542MQ    363
## 10 N531MQ    349
## # ... with 712 more rows
```

Join problems

- Identify the variables that form the primary key in each table.
- Check that none of the variables in the primary key are missing.
- Check that foreign keys match parimary keys in another table (with `anti_join()`).

Set operations

```
df1 <- tribble(
  ~x, ~y,
  1,  1,
  2,  1
)
```

```
df2 <- tribble(
  ~x, ~y,
  1,  1,
  1,  2
)
```

```
intersect(df1, df2)
```

```
## # A tibble: 1 x 2
##       x     y
##   <dbl> <dbl>
## 1     1     1
```

```
union(df1, df2)
```

```
## # A tibble: 3 x 2
##       x     y
##   <dbl> <dbl>
## 1     1     1
## 2     2     1
## 3     1     2
```

```
setdiff(df1, df2)
```

```
## # A tibble: 1 x 2
##       x     y
##   <dbl> <dbl>
## 1     2     1
```

```
setdiff(df2, df1)
```

```
## # A tibble: 1 x 2
##       x     y
##   <dbl> <dbl>
## 1     1     2
```