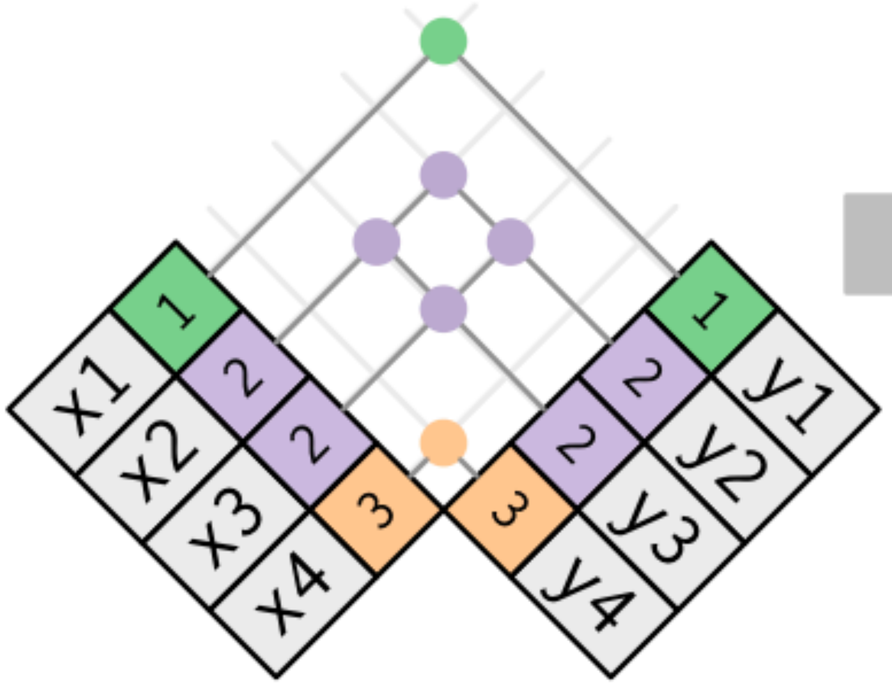
Dealing with duplicate keys can be tricky



When both tables have duplicate keys,
you get all possible combinations.

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

Specifying keys for the match

```
flights2 %>%
  left_join(weather)
#> Joining, by = c("year", "month", "day", "hour", "origin")
#> # A tibble: 336,776 x 18
#>    year month   day  hour origin dest  tailnum carrier  temp  dewp humid
#>   <int> <int> <int> <dbl> <chr>  <chr> <chr>   <chr>   <dbl> <dbl> <dbl>
#> 1  2013     1     1     5 EWR    IAH   N14228  UA       39.0  28.0  64.4
#> 2  2013     1     1     5 LGA    IAH   N24211  UA       39.9  25.0  54.8
#> 3  2013     1     1     5 JFK    MIA   N619AA  AA       39.0  27.0  61.6
#> 4  2013     1     1     5 JFK    BQN   N804JB  B6       39.0  27.0  61.6
#> 5  2013     1     1     6 LGA    ATL   N668DN  DL       39.9  25.0  54.8
#> 6  2013     1     1     5 EWR    ORD   N39463  UA       39.0  28.0  64.4
#> # … with 3.368e+05 more rows, and 7 more variables: wind_dir <dbl>,
#> #   wind_speed <dbl>, wind_gust <dbl>, precip <dbl>, pressure <dbl>,
#> #   visib <dbl>, time_hour <dttm>
```

By default, dplyr functions perform a “**natural join**”, which matches observations using all variables that exist in **both** tables.