Topic

M3T2.3
Filters - join

https://youtu.be/4Me72elHFs0

Simplest Usage Example

numbers.db

```
0 zero
1 one
2 two
3 three
4 four
5 five
6 six
7 seven
8 eight
9 nine
```

nombres.db

```
0 zero
1 un
2 deux
3 trois
4 quatre
5 cinq
6 six
7 sept
8 huit
9 neuf
```

In both files, the key is the first field.

Fields are separated by a blank space.

Both files are already sorted by ascending key number

Joining these 2 files

```
join numbers.txt nombres.txt

    0 zero zero
    1 one un
    2 two deux
    3 three trois
    4 four quatre
    5 five cinq
    6 six six
    7 seven sept
    8 eight huit
    9 nine neuf
```

Output is on STDOUT.

First field is the key.

Fields are separated by a blank space.

For each key, content of files is inserted in line, in the order that the files are specified.

Preparing our “DB”

names.db

```
Mulder Fox 000112222
Scully Dana 111223333
Anderson Thomas 222334444
Anderson Tiffany 333445555
Cooper Dale 444556666
Cooper Murphy 555667777
Watts Wade 666016666
Cook Samantha 666026666
Flynn Kevin 777889999
McPhearson James 888990000
Snape Severus 999001111
```

usernames.db

```
fmuld 000112222
sdana 111223333
neo 222334444
trinity 333445555
dcoop 444556666
murph 555667777
parzival 666016666
art3mis 666026666
clu 777889999
James 888990000
hbp 999001111
```

All files have their lines already sorted by ascending SSN

So that we don’t have to sort everything all the time

Join works only on files that have been sorted by the keys that we want to join with

Basic Join of 2 “tables”

`join` assumes the 1st field is the key, this is not true for our files

```
join -t ' ' -1 3 -2 2 names.db  usernames.db
```

- -t → specifies the delimiter
- -1 → 1st file key selection
- 3 → key is the 3rd field, assuming delimiter

What does the output look like?

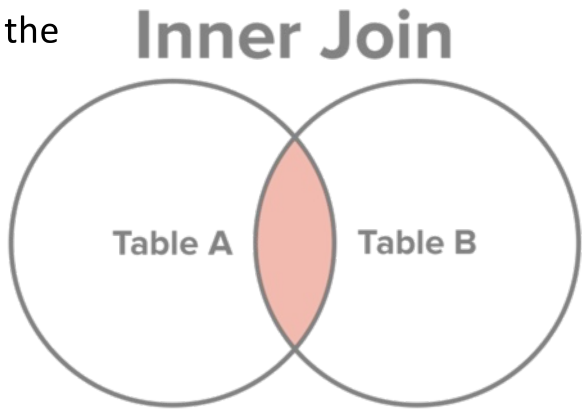
- Key is 1st field in the output
- Combines all fields available in the files in the order the files were specified
- Output is sorted by ascending key as well

What if files are missing information about some keys

- Removing mcphearsons entry from `names.db`
  - 1 SSN appears in `usernames.db` but not `names.db`

```
join -t ' ' -1 3 -2 2 names.db  usernames.db
```

    - By default, nothing appears about McPhearson
    - By default, we want only the keys that are present in BOTH files and ignore the other ones



- Removing dcoop entry in usernames but keep their entry in names.db
  - Same here now the output skips both mcphearson and dale cooper

→Basic join is the intersection

every key must appear in both files

The -a option for outer joins

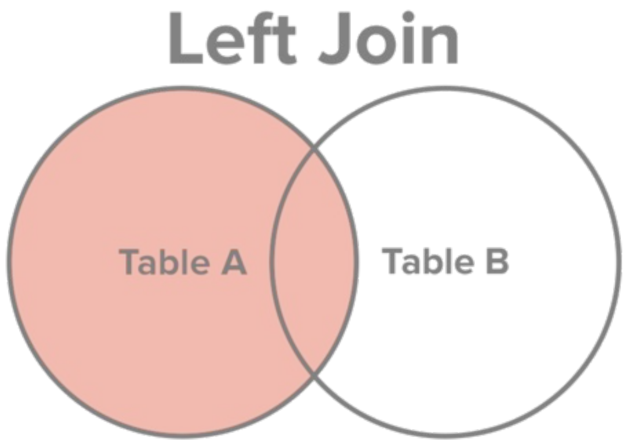
→ Not just the intersection anymore

→ makes sure all entries from a given file are going to show up in output, even if this means they will have partial information

The -a option → Left Outer Join

```
join -t ' ' -1 3 -2 2 -a1 names.db  usernames.db
```

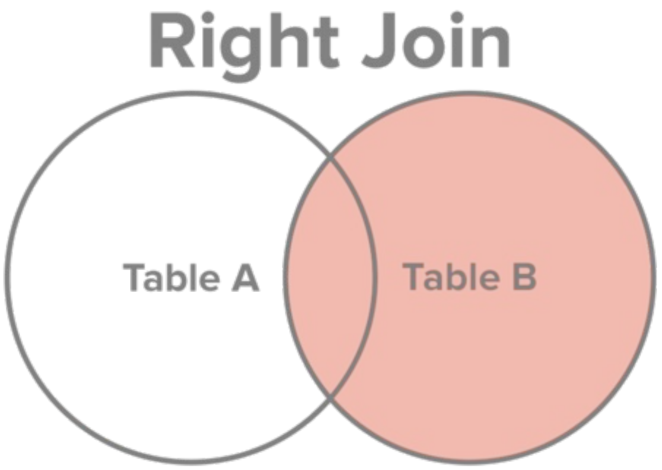
- We want all entries in 1st file even if their key is missing from 2nd file
- Cooper dale shows up in output but without a username field since there is none to be found



The -a option → Right Outer Join

```
join -t ' ' -1 3 -2 2 -a2 names.db  usernames.db
```

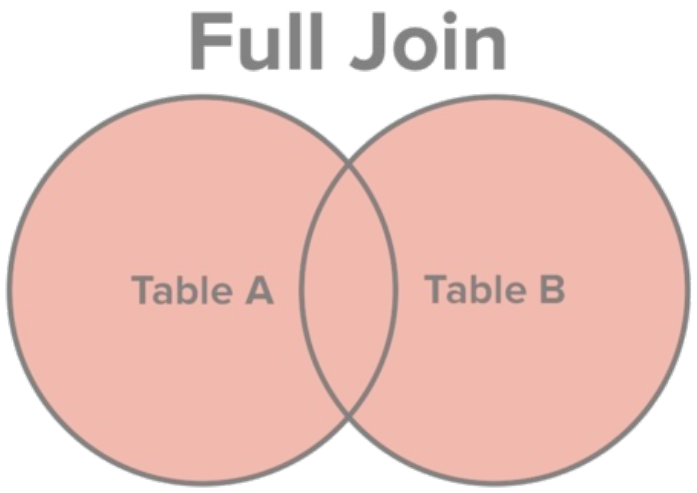
- We want all entries in 2nd file even if their key is missing from 1st file
- Mcphearsons' username shows but not his full name



The -a option → FULL Outer Join

```
join -t ' ' -1 3 -2 2 -a1 -a2 names.db
usernames.db
```

- We want all entries in **both** files even if their key is missing from other file
- We get output lines with missing fields!



The -v option (opposite to -a)

Usage: -v1 or -v2 or -v1 -v2

-v1

- Only Entries from 1st file that do not have corresponding entries in 2nd file must be in output
- "Only what is in file 1 and not in file 2"

```
join -t ' ' -1 3 -2 2 -v1 names.db  usernames.db
```

- Only shows line for dale cooper

```
join -t ' ' -1 3 -2 2 -v1 -v2 names.db  usernames.db
```

- All entries that are in one file but not the other (both of the above)

Let's do some piping!

```
join -t ' ' -1 3 -2 2 names.db  usernames.db
    | cut -d ' ' -f 2,3,4
```

- If we want to see only the names and not the key

```
join -t ' ' -1 3 -2 2 names.db  usernames.db
    | cut -d ' ' --complement -f 1
```

- Extract All fields BUT 1

What if we want the username first and names after?

```
join -t ' ' -1 2 -2 3 usernames.db  names.db
    | cut -d ' ' --complement -f 1
```

The -o option

```
join -t ' ' -1 3 -2 2 -a1 -a2 -e NULL -o 1.1,2.1
names.db  usernames.db
```

- Better than piping into cut
- SELECT the fields that we want in our output
- SELECT the order these fields will show up
- -e replace missing fields by a string