

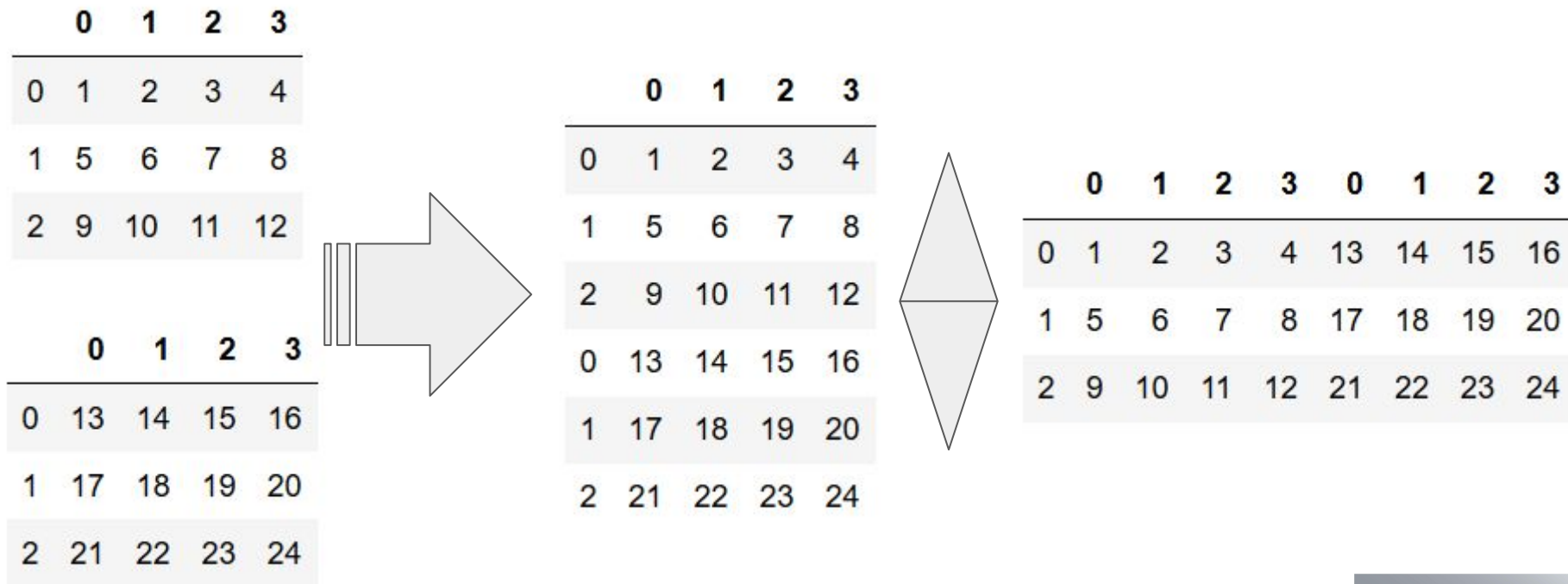
Lezione 9:

Unire i DataFrame

join-merge-concat



Cosa si intende per unire

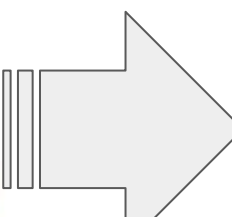


Quali difficoltà?

- Almeno quattro metodi diversi, ma simili
- Ridondanza
- Numerosi parametri
- Utile la conoscenza di SQL



Unire in verticale: **append**



	0	1	2	3
0	0	1	2	3
1	5	6	7	8
2	9	10	11	12

	0	1	2	3
0	13	14	15	16
1	17	18	19	20
2	21	22	23	24

	0	1	2	3
0	0	1	2	3
1	5	6	7	8
2	9	10	11	12
0	13	14	15	16
1	17	18	19	20
2	21	22	23	24

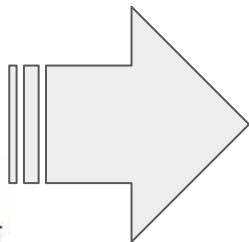
- Il più semplice
- Metodo della classe DF
- `df1.append(df2)`
- Parametro importante:
`ignore_index =`
False/True



Unire in verticale: **concat**

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

	0	1	2	3
0	13	14	15	16
1	17	18	19	20
2	21	22	23	24




	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
0	13	14	15	16
1	17	18	19	20
2	21	22	23	24

- Più completo di append
- Metodo della classe pandas
- `pd.concat([df1, df2...])`
- Presente il parametro `ignore_index`



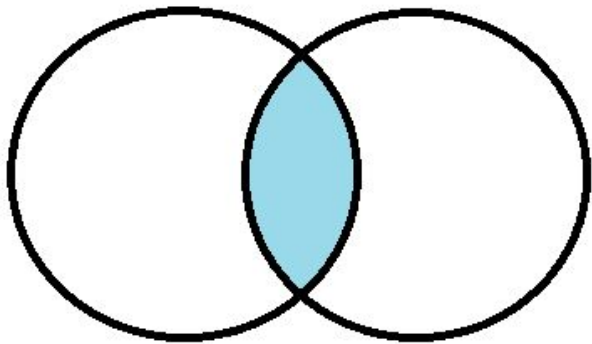
Peculiarità di concat

- Permette di scegliere l'asse su cui "concatenare" i DataFrame
axis=0 => verticale (default)
axis=1 => orizzontale 
- Si può scegliere tra due tipi di join:
 - inner
 - outer

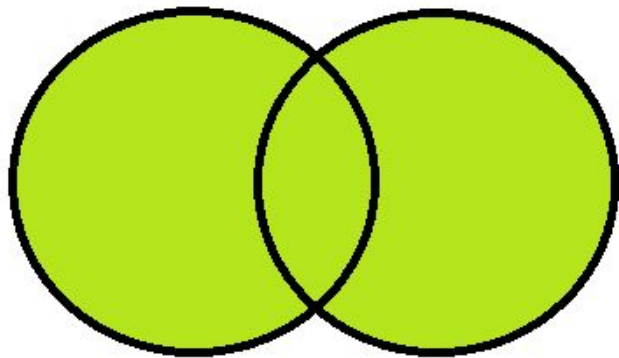
	0	1	2	3	0	1	2	3
0	1	2	3	4	13	14	15	16
1	5	6	7	8	17	18	19	20
2	9	10	11	12	21	22	23	24



Join: inner e outer



INNER JOIN



(FULL) OUTER JOIN



Unire in orizzontale: **join**

0

1

2

3

0

1

2

3

4

1

5

6

7

8

2

9

10

11

12

0

1

2

3

0

13

14

15

16

1

17

18

19

20

2

21

22

23

24

0

1

2

3

0

1

2

3

0

1

2

3

4

13

14

15

16

1

5

6

7

8

17

18

19

20

2

9

10

11

12

21

22

23

24

- Simile a `concat(axis=1)`
- Metodo della classe DataFrame
- `df1.join(df2)`
- Si può scegliere la colonna “on=”
- Presenti anche “left” e “right” join



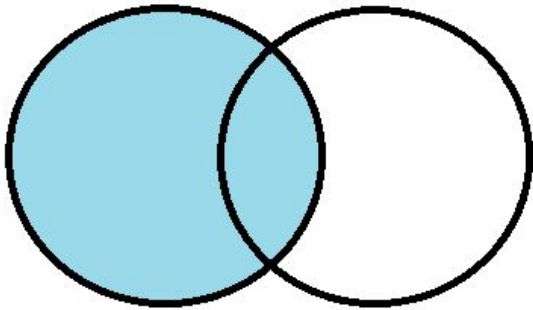
Peculiarità di join

- Permette di scegliere la colonna su cui fare il join delle tabelle che farà da chiave (prima si utilizzavano indici e colonne)
- Si può scegliere tra quattro tipi di join:
 - inner
 - outer
 - left
 - right

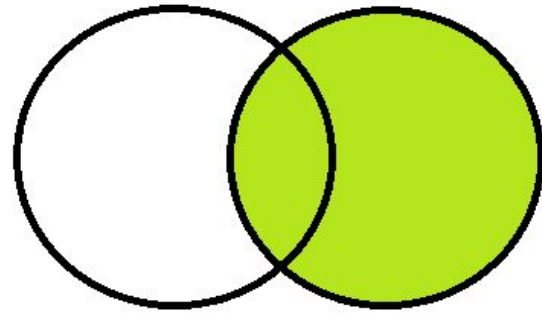
	0	1	2	3		0	1	2	3
0	1	2	3	4	13	14	15	16	
1	5	6	7	8	17	18	19	20	
2	9	10	11	12	21	22	23	24	



Join: left e right



LEFT JOIN



RIGHT JOIN



Peculiarità di **join** (2)

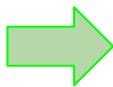
- Si può scegliere un suffisso per la colonna che si è utilizzata come chiave

```
>>> df
```

	key	A
0	K0	A0
1	K1	A1
2	K2	A2
3	K3	A3
4	K4	A4
5	K5	A5

```
>>> other
```

	key	B
0	K0	B0
1	K1	B1
2	K2	B2



```
>>> df.join(other, lsuffix='_caller', rsuffix='_other')
```

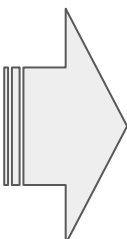
	key_caller	A	key_other	B
0	K0	A0	K0	B0
1	K1	A1	K1	B1
2	K2	A2	K2	B2
3	K3	A3	NaN	NaN
4	K4	A4	NaN	NaN
5	K5	A5	NaN	NaN

Default: “left join”



Unire in orizzontale: **merge**

0	1	2	3
0	1	2	3
1	5	6	7
2	9	10	11



0	1	2	3
0	13	14	15
1	17	18	19
2	21	22	23

0	1	2	3	0	1	2	3
0	1	2	3	4	13	14	15
1	5	6	7	8	17	18	19
2	9	10	11	12	21	22	23

- Simile a **join**
- Metodo sia di pandas che di DataFrame
- `df1.merge(df2)`
- `pd.merge(df1, df2)`



Peculiarità di merge

- Permette di scegliere le colonne su cui fare il join delle tabelle
- Come la **join** ha i suffissi e i quattro tipo di join (default “inner”)

	0	1	2	3		0	1	2	3
0	1	2	3	4	13	14	15	16	
1	5	6	7	8	17	18	19	20	
2	9	10	11	12	21	22	23	24	

```
>>> df1
   lkey value
0    foo     1
1    bar     2
2    baz     3
3    foo     5
>>> df2
   rkey value
0    foo     5
1    bar     6
2    baz     7
3    foo     8
```



```
>>> df1.merge(df2, left_on='lkey', right_on='rkey',
...           suffixes=('_left', '_right'))
   lkey  value_left rkey  value_right
0    foo           1    foo           5
1    foo           1    foo           8
2    foo           5    foo           5
3    foo           5    foo           8
4    bar           2    bar           6
5    baz           3    baz           7
```



Nella prossima lezione...

Ordinare, raggruppare e aggregare i dati

