Объединение наборов данных: слияние и соединение

https://dfedorov.spb.ru

Реляционная алгебра

Реализованное в методе pd.merge поведение представляет собой подмножество того, что известно под названием «реляционная алгебра» (relational algebra).

Реляционная алгебра — формальный набор правил манипуляции реляционными данными, формирующий теоретические основания для имеющихся в большинстве баз данных операций. Сила реляционного подхода состоит в предоставлении им нескольких простейших операций — своеобразных «кирпичиков» для построения более сложных операций над любым набором данных. При наличии эффективно реализованного в базе данных или другой программе подобного базового набора операций можно выполнять широкий диапазон весьма сложных составных операций.

Библиотека Pandas реализует несколько из этих базовых «кирпичиков» в функции pd.merge() и родственном ей методе join() объектов Series и DataFrame. Они обеспечивают возможность эффективно связывать данные из различных источников.

Виды соединений

Функция pd.merge() реализует множество типов соединений: **«один-к-одному»**, **«многие-к-одному»** и **«многие-ко-многим»**. Все эти три типа соединений доступны через один и тот же вызов pd.merge() , тип выполняемого соединения зависит от формы входных данных.

Соединения «один-к-одному»

Простейший тип выражения слияния — соединение «один-к-одному», во многом напоминающее конкатенацию по столбцам. В качестве примера рассмотрим два объекта DataFrame, содержащие информацию о нескольких служащих компании:

```
>>> df1 = pd.DataFrame({'employee': ['Bob', 'Jake', 'Lisa', 'Sue'],
                        'group': ['Accounting', 'Engineering', 'Engineering', 'HR']})
>>> df1

>>> df2 = pd.DataFrame({'employee': ['Lisa', 'Bob', 'Jake', 'Sue'],
                        'hire_date': [2004, 2008, 2012, 2014]})
>>> df2

>>> df3 = pd.merge(df1, df2)
>>> df3
```

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering
3	Sue	HR

	employee	hire_date
0	Lisa	2004
1	Bob	2008
2	Jake	2012
3	Sue	2014

	employee	group	hire_date
0	Bob	Accounting	2008
1	Jake	Engineering	2012
2	Lisa	Engineering	2004
3	Sue	HR	2014

Функция pd.merge() распознает, что в обоих объектах DataFrame имеется столбец employee, и автоматически выполняет соединение, используя этот столбец в качестве ключа. Результатом слияния становится новый объект DataFrame, объединяющий информацию из двух входных объектов.

Не забывайте, что слияние игнорирует индекс, за исключением особого случая слияния по индексу.

Соединения «многие-к-одному»

«Многие-к-одному» — соединения, при которых один из двух ключевых столбцов содержит дублирующиеся значения. В случае соединения «многие-к-одному» в итоговом объекте DataFrame эти дублирующиеся записи будут сохранены.

```
>>> df3

>>> df4 = pd.DataFrame({'group': ['Accounting', 'Engineering', 'HR'],
                        'supervisor': ['Carly', 'Guido', 'Steve']})
>>> df4

>>> pd.merge(df3, df4)
```

	employee	group	hire_date
0	Bob	Accounting	2008
1	Jake	Engineering	2012
2	Lisa	Engineering	2004
3	Sue	HR	2014

	group	supervisor
0	Accounting	Carly
1	Engineering	Guido
2	HR	Steve

В итоговом объекте DataFrame имеется дополнительный столбец с информацией о руководителе (supervisor) с повторением информации в одном или нескольких местах в соответствии с вводимыми данными.

	employee	group	hire_date	supervisor
0	Bob	Accounting	2008	Carly
1	Jake	Engineering	2012	Guido
2	Lisa	Engineering	2004	Guido
3	Sue	HR	2014	Steve

Соединения «многие-ко-многим»

Если столбец ключа как в левом, так и в правом массивах содержит повторяющиеся значения, результат окажется слиянием типа «многие-ко-многим».

```
>>> df1

>>> df5 = pd.DataFrame({'group': ['Accounting', 'Accounting', 'Engineering',
                                  'Engineering','HR', 'HR'],
                        'skills': ['math', 'spreadsheets', 'coding',
                                   'linux', 'spreadsheets','organization']})
>>> df5

>>> pd.merge(df1, df5)
```

Выполнив соединение «многие-ко-многим», можно выяснить навыки каждого конкретного человека.

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering
3	Sue	HR

	group	skills
0	Accounting	math
1	Accounting	spreadsheets
2	Engineering	coding
3	Engineering	linux
4	HR	spreadsheets
5	HR	organization

	employee	group	skills
0	Bob	Accounting	math
1	Bob	Accounting	spreadsheets
2	Jake	Engineering	coding
3	Jake	Engineering	linux
4	Lisa	Engineering	coding
5	Lisa	Engineering	linux
6	Sue	HR	spreadsheets
7	Sue	HR	organization

Задание ключа слияния

Мы рассмотрели поведение метода pd.merge() по умолчанию: он выполняет поиск в двух входных объектах соответствующих названий столбцов и использует найденное в качестве ключа. Однако зачастую имена столбцов не совпадают добуквенно точно, и в методе pd.merge() имеется немало параметров для такой ситуации.

https://stackoverflow.com/questions/53645882/pandas-merging-101

Ключевое слово on

Проще всего указать название ключевого столбца с помощью ключевого слова **on** , в котором указывается название или список названий столбцов:

```
>>> df1

>>> df2

>>> pd.merge(df1, df2, on='employee')
```

Этот параметр работает только в том случае, когда в левом и правом объектах DataFrame имеется указанное название столбца.

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering
3	Sue	HR

	employee	hire_date
0	Lisa	2004
1	Bob	2008
2	Jake	2012
3	Sue	2014

	employee	group	hire_date
0	Bob	Accounting	2008
1	Jake	Engineering	2012
2	Lisa	Engineering	2004
3	Sue	HR	2014

Ключевые слова left_on и right_on

Иногда приходится выполнять слияние двух наборов данных с различными именами столбцов.
Например, у нас может быть набор данных, в котором столбец для имени служащего называется Name , а не Employee . В этом случае можно воспользоваться ключевыми словами **left_on** и **right_on** для указания названий двух нужных столбцов:

```
>>> df1

>>> df3 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'],
                        'salary': [70000, 80000, 120000, 90000]})
>>> df3

>>> pd.merge(df1, df3, left_on="employee", right_on="name")
```

Результат этой операции содержит избыточный столбец, который можно при желании удалить. Например, с помощью имеющегося в объектах DataFrame метода drop().

	employee	group
0	Bob	Accounting
1	Jake	Engineering
2	Lisa	Engineering
3	Sue	HR

	name	salary
0	Bob	70000
1	Jake	80000
2	Lisa	120000
3	Sue	90000

	employee	group	name	salary
0	Bob	Accounting	Bob	70000
1	Jake	Engineering	Jake	80000
2	Lisa	Engineering	Lisa	120000
3	Sue	HR	Sue	90000

```
>>> pd.merge(df1, df3, left_on="employee", right_on="name").drop('name', axis=1)
```

Ключевые слова left_index и right_index

https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html

Иногда удобнее вместо слияния по столбцу выполнить слияние по индексу.

```
>>> df1a = df1.set_index('employee')
>>> df1a
```

employee	group
Bob	Accounting
Jake	Engineering
Lisa	Engineering
Sue	HR

```
>>> df2a = df2.set_index('employee')
>>> df2a
```

employee	hire_date
Lisa	2004
Bob	2008
Jake	2012
Sue	2014

```
>>> pd.merge(df1a, df2a, left_index=True, right_index=True)
```

employee	group	hire_date
Bob	Accounting	2008
Jake	Engineering	2012
Lisa	Engineering	2004
Sue	HR	2014

Можно проще:

```
>>> df1a.join(df2a)
```

Если требуется комбинация слияния по столбцам и индексам, можно для достижения нужного поведения воспользоваться сочетанием флага **left_index** с параметром **right_on** или параметра **left_on** с флагом **right_index** :

```
>>> df1a

>>> df3

>>> pd.merge(df1a, df3, left_index=True, right_on='name')
```

employee	group
Bob	Accounting
Jake	Engineering
Lisa	Engineering
Sue	HR

	name	salary
0	Bob	70000
1	Jake	80000
2	Lisa	120000
3	Sue	90000

	group	name	salary
0	Accounting	Bob	70000
1	Engineering	Jake	80000
2	Engineering	Lisa	120000
3	HR	Sue	90000

Задание операций над множествами для соединений

```
>>> df6 = pd.DataFrame({'name': ['Peter', 'Paul', 'Mary'],
                        'food': ['fish', 'beans', 'bread']},
                         columns=['name', 'food'])
>>> df6

>>> df7 = pd.DataFrame({'name': ['Mary', 'Joseph'],
                        'drink': ['wine', 'beer']},
                         columns=['name', 'drink'])
>>> df7

>>> pd.merge(df6, df7)
```

Здесь мы слили воедино два набора данных, у которых совпадает только одна запись name : Mary.
По умолчанию результат будет содержать пересечение двух входных множеств — внутреннее соединение (**inner join**).
Можно указать это явным образом, с помощью ключевого слова **how** , имеющего по умолчанию значение **'inner'**:

	name	food
0	Peter	fish
1	Paul	beans
2	Mary	bread

	name	drink
0	Mary	wine
1	Joseph	beer

	name	food	drink
0	Mary	bread	wine

```
>>> pd.merge(df6, df7, how='inner')
```

Другие возможные значения ключевого слова how : **'outer'** , **'left'** и **'right'** .

Внешнее соединение (**outer join**) означает соединение по объединению входных столбцов и заполняет значениями NA все пропуски значений:

```
>>> df6
>>> df7
>>> pd.merge(df6, df7, how='outer')
```

	name	food
0	Peter	fish
1	Paul	beans
2	Mary	bread

	name	drink
0	Mary	wine
1	Joseph	beer

	name	food	drink
0	Peter	fish	NaN
1	Paul	beans	NaN
2	Mary	bread	wine
3	Joseph	NaN	beer

Левое соединение (left join) и правое соединение (right join) выполняют соединение по записям слева и справа соответственно.

```
>>> df6
>>> df7
>>> pd.merge(df6, df7, how='left')
```

	name	food
0	Peter	fish
1	Paul	beans
2	Mary	bread

Использует ключи из левого df

	name	drink
0	Mary	wine
1	Joseph	beer

	name	food	drink
0	Peter	fish	NaN
1	Paul	beans	NaN
2	Mary	bread	wine

```
>>> df6
>>> df7
>>> pd.merge(df6, df7, how='right')
```

	name	food
0	Peter	fish
1	Paul	beans
2	Mary	bread

Использует ключи из правого df

	name	drink
0	Mary	wine
1	Joseph	beer

	name	food	drink
0	Mary	bread	wine
1	Joseph	NaN	beer

https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html#brief-primer-on-merge-methods-relational-algebra

Пересекающиеся названия столбцов: ключевое слово suffixes

Вам может встретиться случай, когда в двух входных объектах присутствуют конфликтующие названия столбцов.

```
>>> df8 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'],
                        'rank': [1, 2, 3, 4]})
>>> df8

>>> df9 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'],
                        'rank': [3, 1, 4, 2]})
>>> df9

>>> pd.merge(df8, df9, on="name")
```

	name	rank
0	Bob	1
1	Jake	2
2	Lisa	3
3	Sue	4

	name	rank
0	Bob	3
1	Jake	1
2	Lisa	4
3	Sue	2

Поскольку в результате должны были быть два конфликтующих названия столбцов, функция слияния автоматически добавила в названия суффиксы _x и _y , чтобы обеспечить уникальность названий столбцов результата.

	name	rank_x	rank_y
0	Bob	1	3
1	Jake	2	1
2	Lisa	3	4
3	Sue	4	2

```
>>> pd.merge(df8, df9, on="name", suffixes=["_L", "_R"])
```