УДК 004.423 **Информационные технологии**

Дрянкова Дарья Александровна, *студент факультета информатики и вычислительной техники, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

Замулин Иван Сергеевич, *заведующий кафедрой ПОВТиАС, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия*

ОБЪЕДИНЕНИЕ ТАБЛИЦ ДАННЫХ С ПОМОЩЬЮ БИБЛИОТЕКИ PANDAS ДЛЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Аннотация: В статье рассматриваются способы объединения таблиц библиотеки Pandas языка программирования.

Ключевые слова: Python, Pandas, таблицы данных, столбцы, строки, DataFrame, объединение, методы, функции.

Annotation: The article discusses ways to combine the tables of the Pandas library of the programming language.

Keywords: Python, Pandas, data tables, columns, rows, DataFrame, union, methods, functions.

Библиотека Pandas - это библиотека для работы с данными в формате таблиц на языке программирования Python. Она позволяет считывать, записывать и обрабатывать данные из различных источников, таких как файлы CSV, Excel, базы данных SQL и другие.

Помимо создания таблиц, их фильтрации и выборки, Pandas также предоставляет инструменты для объединения нескольких таблиц данных, а также для преобразования таблиц данных в соответствии с определенными правилами. Это позволяет объединять данные из разных источников и проводить более сложный анализ данных.

Конкатенация и объединение таблиц - важные операции при работе с данными. Pandas предоставляет множество методов для выполнения этих задач, включая метод join(), который объединяет две таблицы на основе значений в индексе или столбцах [1].

Рассмотрим пример, где мы будем объединять две таблицы по общему столбцу с помощью метода join(). Допустим, у нас есть две таблицы df1 и df2, содержащие информацию о продажах компании по месяцам, они изображены на рисунке 1.

```
>>> import pandas as pd
>>>
>>> # Создаем первую таблицу
... data1 = {'Месяц': ['Январь', 'Февраль', 'Март', 'Апрель'],
...          'Продажи': [100, 200, 150, 300]}
>>> df1 = pd.DataFrame(data1)
>>>
>>> # Создаем вторую таблицу
... data2 = {'Месяц': ['Апрель', 'Май', 'Июнь', 'Июль'],
...          'Продажи': [250, 150, 200, 350]}
>>> df2 = pd.DataFrame(data2)
```

Рисунок 1 – Таблицы с информацией о продажах компании

На рисунке 2 изображено содержание таблицы данных df1 и df2.

```
>>> df1
     Месяц  Продажи
0   Январь      100
1  Февраль      200
2     Март      150
3   Апрель      300
>>>
>>> df2
    Месяц  Продажи
0  Апрель      250
1     Май      150
2    Июнь      200
3    Июль      350
```

Рисунок 2 – Содержание таблиц данных df1 и df2

Мы хотим объединить эти таблицы по столбцу "Месяц". Для этого мы можем использовать метод join() способом, изображенном на рисунке 3.

```
>>> df3 = df1.join(df2.set_index('Месяц'), on='Месяц', rsuffix='_2')
>>> df3
     Месяц  Продажи  Продажи_2
0   Январь      100        NaN
1  Февраль      200        NaN
2    Март       150        NaN
3   Апрель      300      250.0
```

Рисунок 3 – Объединение таблиц данных с помощью метода join()

В этом примере мы сначала установили индекс таблицы df2 на столбец "Месяц", используя метод set_index(). Затем мы объединили таблицы с помощью метода join(), указав "Месяц" как ключ объединения. Мы также использовали аргумент rsuffix='_2', чтобы указать, что столбец "Продажи" из таблицы df2 должен быть переименован в "Продажи_2", чтобы избежать конфликтов имен.

Помимо метода join() существует ещё несколько способов объединения таблиц данных. К примеру метод merge(), объединяющий две таблицы в одну на основе заданных столбцов. Аналогично предыдущей функции merge также объединяет все столбцы из двух таблиц с общими столбцами. Но в отличие от join, merge уже предлагает три способа организации построчного выравнивания:

1) on = 'название столбца' – объединяющий таблицы по общему столбцу;

2) left_on on = 'название столбца', right_on on = 'название столбца' – позволяющий объединить две таблицы, используя два разных столбца;

3) left_index = True, right_index = True – объединяющий таблицы по индексам;

Мы рассмотрим способ объединения с использованием первого из перечисленных выше способов.

На рисунке 4 изображено использование метода merge() с объединением таблиц по столбцу on = 'id'.

```
>>> import pandas as pd
>>>
>>> df1 = pd.DataFrame({
...         'id': [1, 2, 3, 4, 5],
...         'name': ['Alice', 'Bob', 'Charlie', 'David', 'Emily'],
...         'age': [25, 32, 18, 47, 22],
...     })
>>>
>>> df2 = pd.DataFrame({
...         'id': [1, 2, 3, 4, 5],
...         'score': [80, 90, 70, 85, 95],
...     })
>>>
>>> df3 = pd.merge(df1, df2, on ='id')
```

Рисунок 4 – Использование метода merge() для объединения двух таблиц данных

На рисунке 5 изображено содержание таблиц df1 и df2, а также результат их соединения с помощью merge в таблице данных df3.

```
>>> df1
   id     name  age
0   1    Alice   25
1   2      Bob   32
2   3  Charlie   18
3   4    David   47
4   5    Emily   22
>>> df2
   id  score
0   1     80
1   2     90
2   3     70
3   4     85
4   5     95
>>> df3
   id     name  age  score
0   1    Alice   25     80
1   2      Bob   32     90
2   3  Charlie   18     70
3   4    David   47     85
4   5    Emily   22     95
```

Рисунок 5 – Содержание таблиц df1, df2 и df3

Следующим методом объединения является метод concat(), позволяющий объединить таблицы вертикально или горизонтально. На рисунке 6 приведён пример использования данного метода.

```
>>> import pandas as pd

>>> # Создание первой таблицы
... df1 = pd.DataFrame({
...         'A': ['A0', 'A1', 'A2', 'A3'],
...         'B': ['B0', 'B1', 'B2', 'B3'],
...         'C': ['C0', 'C1', 'C2', 'C3'],
...         'D': ['D0', 'D1', 'D2', 'D3']
...     })
>>>
>>> # Создание второй таблицы
... df2 = pd.DataFrame({
...         'A': ['A4', 'A5', 'A6', 'A7'],
...         'B': ['B4', 'B5', 'B6', 'B7'],
...         'C': ['C4', 'C5', 'C6', 'C7'],
...         'D': ['D4', 'D5', 'D6', 'D7']
...     })
>>>
>>> res1 = pd.concat([df1, df2], axis=0)
>>> res2 = pd.concat([df1, df2], axis=1)
```

Рисунок 6 – Объединение таблиц с помощью метода concat()

В параметре функции необходимо прописать axis = 0 или axis = 1, соответственно вписанному значению в res1 происходит вертикальное объединение таблиц, а в res2 происходит горизонтальное объединение.

При выводе таблиц res1 и res2 получится результат, изображенный на рисунке 7.

```
>>> res1
    A   B   C   D
0  A0  B0  C0  D0
1  A1  B1  C1  D1
2  A2  B2  C2  D2
3  A3  B3  C3  D3
0  A4  B4  C4  D4
1  A5  B5  C5  D5
2  A6  B6  C6  D6
3  A7  B7  C7  D7
>>> res2
    A   B   C   D   A   B   C   D
0  A0  B0  C0  D0  A4  B4  C4  D4
1  A1  B1  C1  D1  A5  B5  C5  D5
2  A2  B2  C2  D2  A6  B6  C6  D6
3  A3  B3  C3  D3  A7  B7  C7  D7
```

Рисунок 7 – Содержание таблиц объединенных в res1 и res2

Также нужно заметить, что отличии от join и merge, concat позволяет работать как со столбцами, так и со строками, но не дает переименовывать

строки или столбцы.

Последним методом объединения является append(), котрый используется для объединения строк одной таблицы с другой таблицей или DataFrame. Этот метод создает новый объект DataFrame, содержащий соединенные строки.

На рисунке 8 приведён пример использования метода append() для объединения двух таблиц.

```
>>> import pandas as pd
>>>
>>> # Создание первой таблицы
... df1 = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
>>>
>>> # Создание второй таблицы
... df2 = pd.DataFrame({'A': [7, 8, 9], 'B': [10, 11, 12]})
>>>
>>> # Объединение таблиц с помощью метода append()
... df3 = df1._append(df2)
>>>
>>> df3
   A   B
0  1   4
1  2   5
2  3   6
0  7  10
1  8  11
2  9  12
```

Рисунок 8 – Объединение таблиц с помощью метода append()

Также, столбцы, не входящие в исходный датафрейм, добавляются как новые столбцы, а новые ячейки заполняются значениями NaN.

На рисунке 9 представлен пример того, как метод append() также может использоваться для добавления новых строк в существующую таблицу.

```
>>> import pandas as pd

>>> # Создание пустой таблицы с заданными столбцами
... df = pd.DataFrame(columns=['A', 'B'])
>>>
>>> # Добавление новой строки в таблицу
... df = df._append({'A': 1, 'B': 4}, ignore_index=True)
>>>
>>> df
   A  B
0  1  4
```

Рисунок 9 – Добавление строк в существующую таблицу с помощью метода append()

Обратите внимание на параметр ignore_index, который указывает Pandas на то, что индексы строк должны быть пересчитаны с 0 после добавления новой строки в таблицу. Если не указать этот параметр, то индексы будут продолжаться от предыдущих строк таблицы.

Заключение

Библиотека Pandas предоставляет мощные инструменты для объединения DataFrames. Но бывает сложно решить, когда что использовать. Хотя в большинстве случаев функции merge достаточно, в некоторых случаях вы можете использовать concat для слияния по строкам, или использовать join. Также существует возможность добавления строк данных с помощью append() [2].

Библиографический список:

1. Яворски М., Зиаде Т. Python. Лучшие практики и инструменты [Текст] / Яворски М., Зиаде Т. – СПб.: Питер, 2021. – 560 с. ISBN: 978-5-4461-1589-1.

2. Марк Саммерфилд. Python на практике [Текст] / Марк Саммерфилд. – М.: ДМК-Пресс, 2016. – 338 с. ISBN: 978-5-97060-095-5.