# **Лабораторная работа №31 (2 часа)**

**Тема работы: «Разработка, отладка и испытание программ анализа**

**данных посредством «реляционной алгебры»»**

**1 Цель работы**

Закрепить навык анализа данных посредством «реляционной алгебры»

**2 Задание**

Создайте два объекта Series с данными. Необходимо с помощью библиотеки Pandas выполнить объединение двух наборами данных следующим образом:

* inner;
* outer;

**3 Оснащение работы**

Задание по варианту, ЭВМ, среда разработки **Python 3.7, IDLE**.

**4 Основные теоретические сведения**

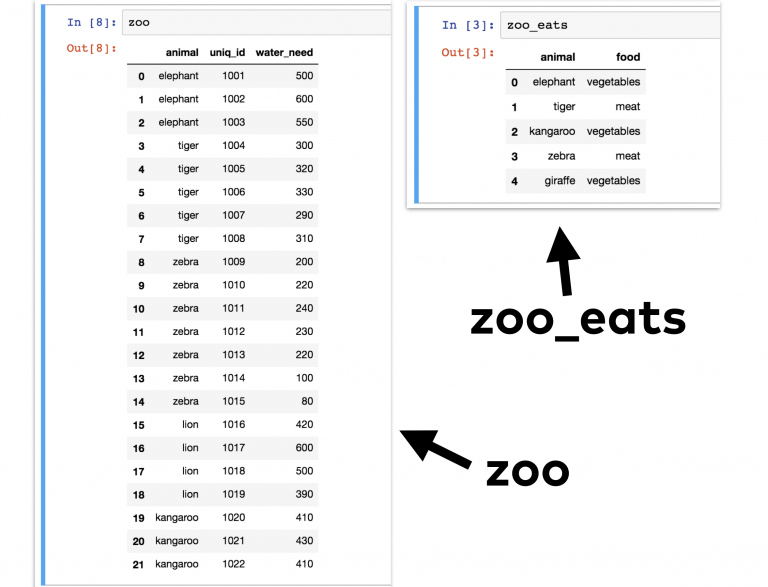
**Merge в pandas («объединение» Data Frames)**

В реальных проектах данные обычно не хранятся в одной таблице. Вместо нее используется много маленьких. И на то есть несколько причин. С помощью нескольких таблиц данными легче управлять, проще избегать «многословия», можно экономить место на диске, а запросы к таблицам обрабатываются быстрее.

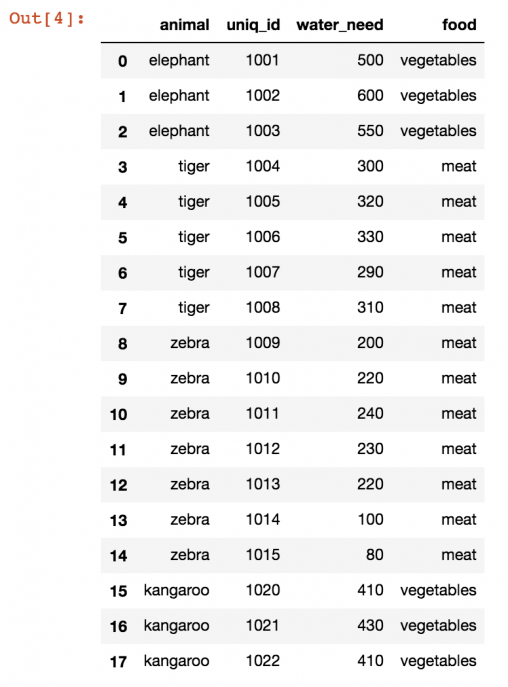
Суть в том, что при работе с данными довольно часто придется вытаскивать данные из двух и более разных страниц. Это делается с помощью merge.

Примечание: хотя в pandas это называется merge, метод почти не отличается от JOIN в SQL.

Рассмотрим пример. Для этого можно взять DataFrame zoo (из [предыдущих частей руководства](https://pythonru.com/uroki/osnovy-pandas-1-chtenie-fajlov-dataframe-otbor-dannyh)), в котором есть разные животные. Но в этот раз нужен еще один DataFrame — zoo\_eats, в котором будет описаны пищевые требования каждого вида.



Теперь нужно объединить два эти Data Frames в один. Чтобы получилось нечто подобное:



В этой таблице можно проанализировать, например, сколько животных в зоопарке едят мясо или овощи.

**Как делается merge?**

В первую очередь нужно создать DataFrame zoo\_eats, потому что zoo уже имеется из прошлых частей. Для упрощения задачи вот исходные данные:

animal;food

elephant;vegetables

tiger;meat

kangaroo;vegetables

zebra;vegetables

giraffe;vegetables

О том, как превратить этот набор в DataFrame, написано в первом уроке по pandas. Но есть способ для ленивых. Нужно лишь скопировать эту длинную строку в Jupyter Notebook pandas\_tutorial\_1, который был создан еще в первой части руководства.

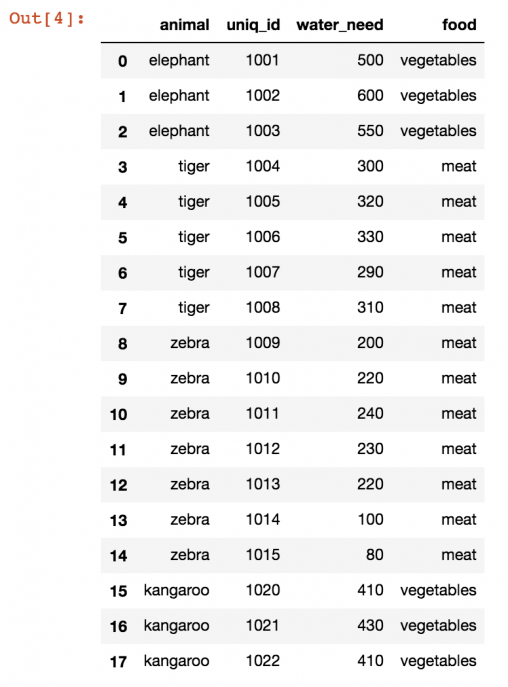
zoo\_eats = pd.DataFrame([['elephant','vegetables'], ['tiger','meat'], ['kangaroo','vegetables'], ['zebra','vegetables'], ['giraffe','vegetables']], columns=['animal', 'food'])

И вот готов DataFrame zoo\_eats.



Теперь пришло время метода merge:

zoo.merge(zoo\_eats)



(А где же все львы? К этому вернемся чуть позже).

Это было просто, не так ли? Но стоит разобрать, что сейчас произошло:

Сначала был указан первый DataFrame (zoo). Потом к нему применен метод .merge(). В качестве его параметра выступает новый DataFrame (zoo\_eats). Можно было сделать и наоборот:

zoo\_eats.merge(zoo)

Это то же самое, что и:

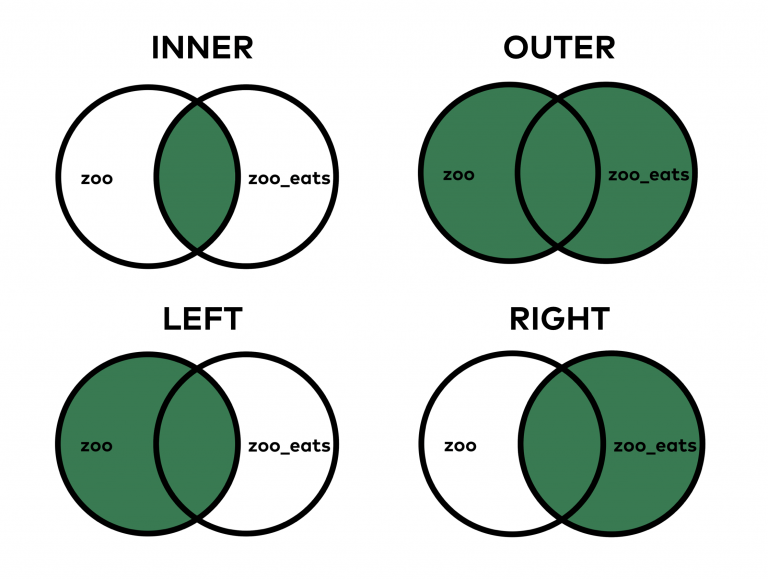
zoo.merge(zoo\_eats)

Разница будет лишь в порядке колонок в финальной таблице.

Способы объединения: inner, outer, left, right

Базовый метод merge довольно прост. Но иногда к нему нужно добавить несколько параметров.

Один из самых важных вопросов — как именно нужно объединять эти таблицы. В SQL есть 4 типа JOIN.

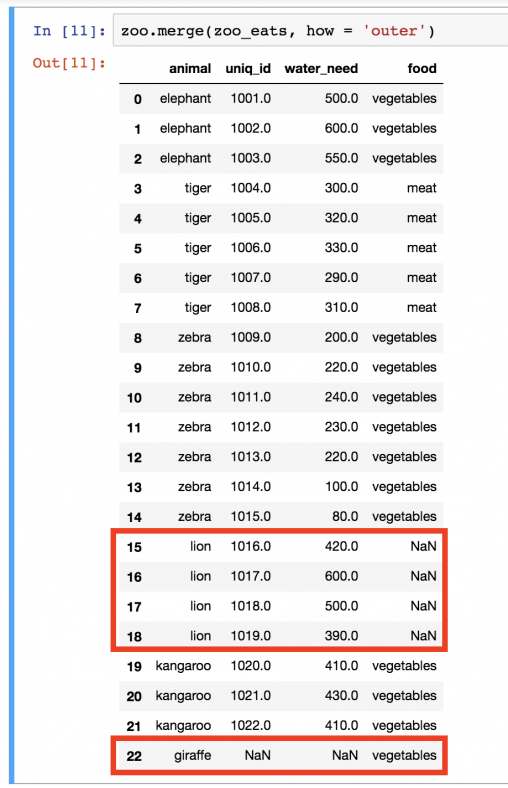


В случае с merge в pandas в теории это работает аналогичным образом.

При выборе INNER JOIN (вид по умолчанию в SQL и pandas) объединяются только те значения, которые можно найти в обеих таблицах. В случае же с OUTER JOIN объединяются все значения, даже если некоторые из них есть только в одной таблице.

Конкретный пример: в zoo\_eats нет значения lion. А в zoo нет значения giraffe. По умолчанию использовался метод INNER, поэтому и львы, и жирафы пропали из таблицы. Но бывают случаи, когда нужно, чтобы все значения оставались в объединенном DataFrame. Этого можно добиться следующим образом:

zoo.merge(zoo\_eats, how='outer')



В этот раз львы и жирафы вернулись. Но поскольку вторая таблица не предоставила конкретных данных, то вместо значения ставится пропуск (NaN).

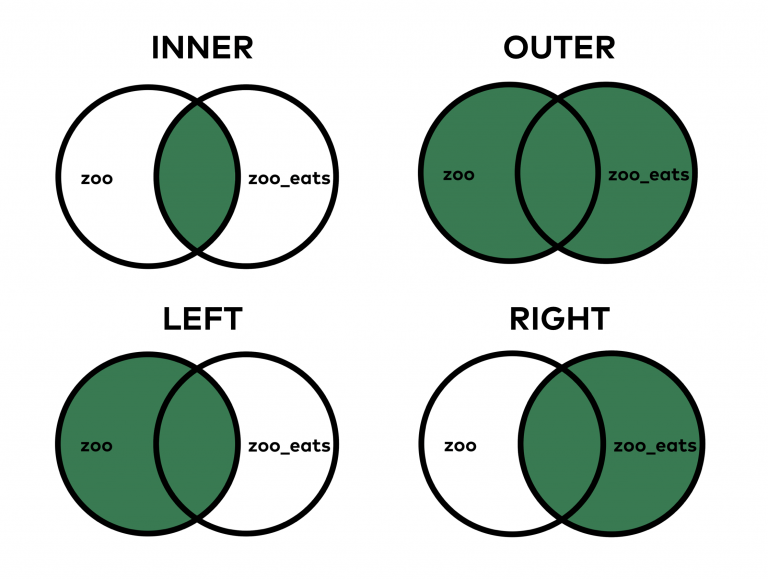
Логичнее всего было бы оставить в таблице львов, но не жирафов. В таком случае будет три типа еды: vegetables, meat и NaN (что, фактически, значит, «информации нет»). Если же в таблице останутся жирафы, это может запутать, потому что в зоопарке-то этого вида животных все равно нет. Поэтому следует воспользоваться параметром how='left' при объединении.

Вот так:

zoo.merge(zoo\_eats, how='left')

Теперь в таблице есть вся необходимая информация, и ничего лишнего. how = 'left' заберет все значения из левой таблицы (zoo), но из правой (zoo\_eats) использует только те значения, которые есть в левой.

Еще раз взглянем на типы объединения:



Примечание: «Какой метод merge является самым безопасным?» — самый распространенный вопрос. Но на него нет однозначного ответа. Нужно решать в зависимости от конкретной задачи.

**5 Порядок выполнения работы**

1. Выделить ключевые моменты задачи.

2. Построить алгоритм решения задачи.

3. Запрограммировать полученный алгоритм.

4. Провести тестирование полученной программы.

**6 Форма отчета о работе**

*Лабораторная работа № \_\_\_*

*Номер учебной группы \_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия, инициалы учащегося: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Тема работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Цель работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Оснащение работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Результат выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**7 Контрольные вопросы и задания**

1. Что такое объединение наборов данных?
2. Какие бывают виды объединения наборов данных?
3. Для чего применяется объединение данных?

**8 Рекомендуемая литература**

**Плас, Дж. В.** Python для сложных задач. Наука о данных и машинное обучение / Дж.В. Плас. – СПб: Питер, 2018.

**Прохоренок, Н.А.** Python 3. Самое необходимое / Н.А Прохоренок, В.А. Дронов – СПб.: БВХ-Петербург, 2016.