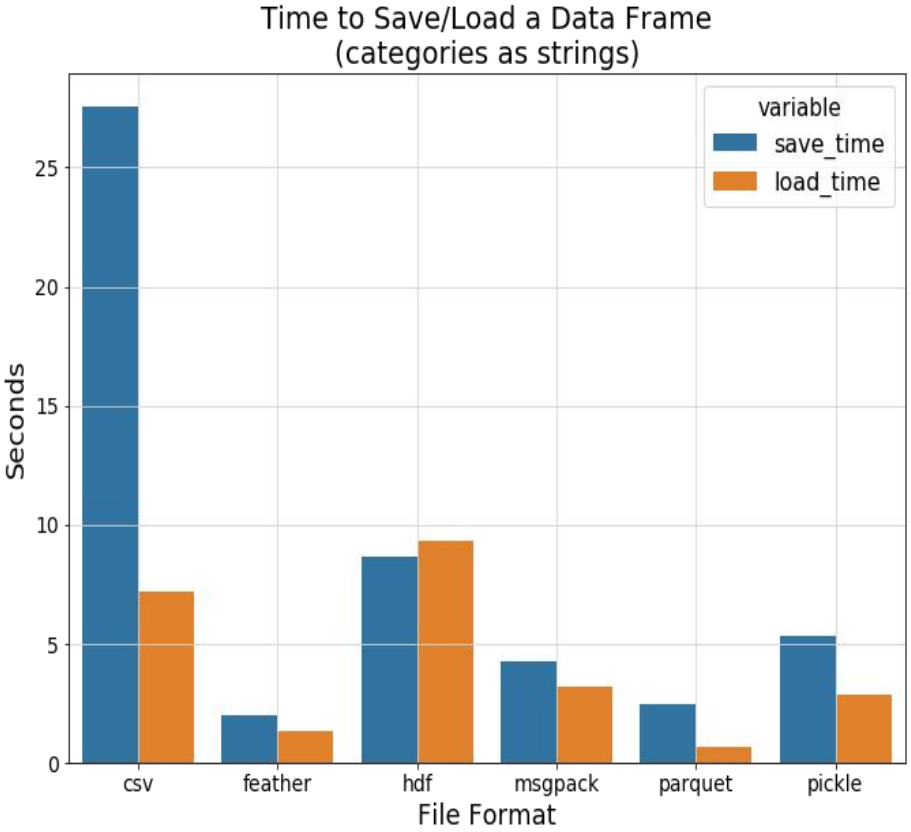
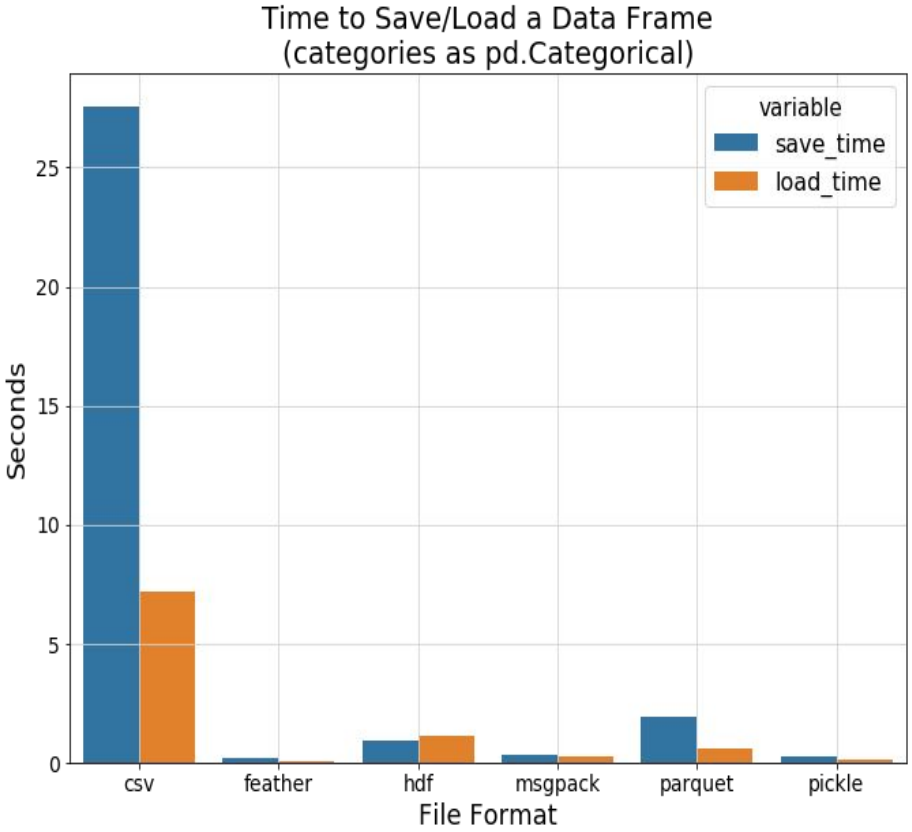
Форматы хранения данные

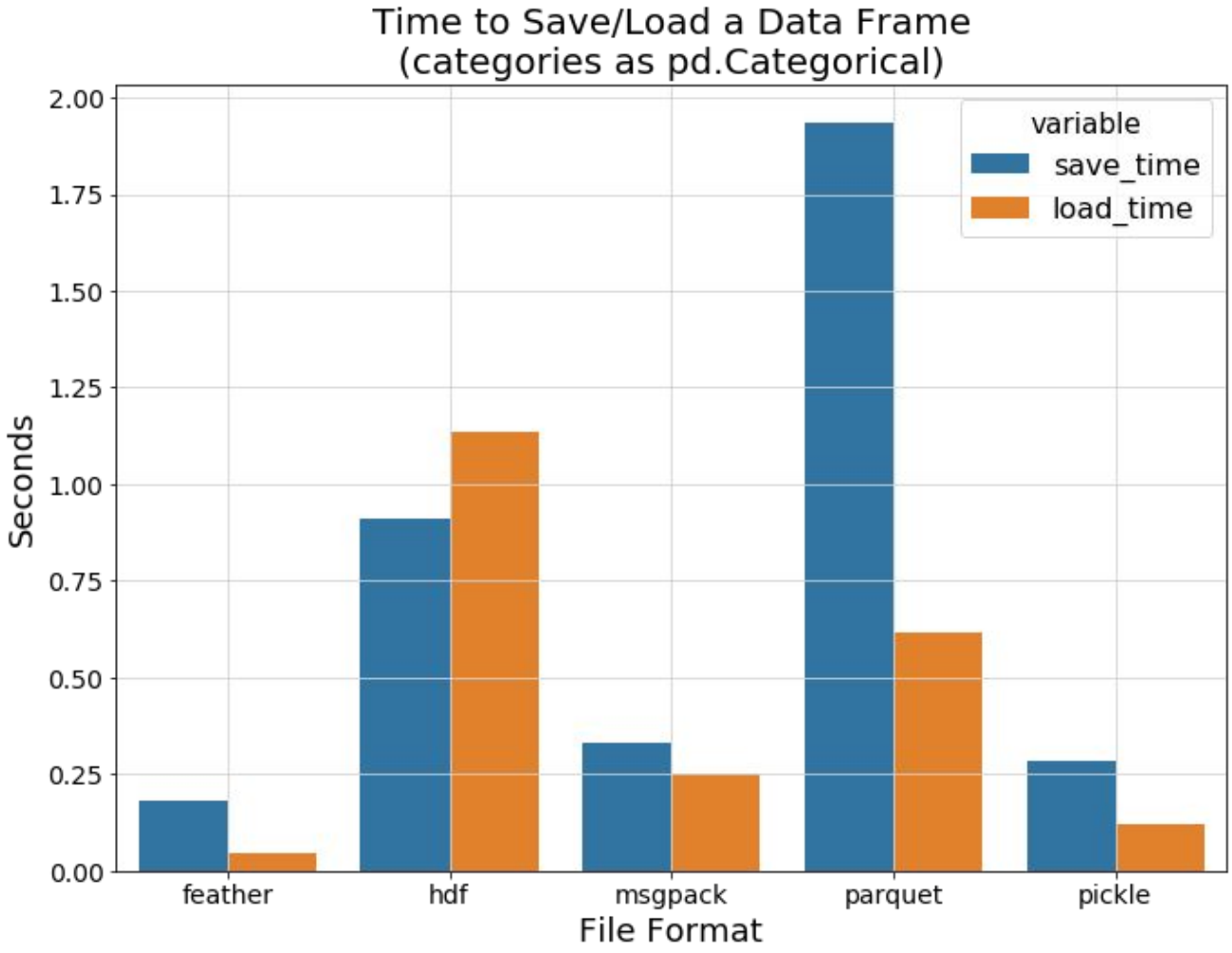
Сравнение основных форматов для хранения файлов

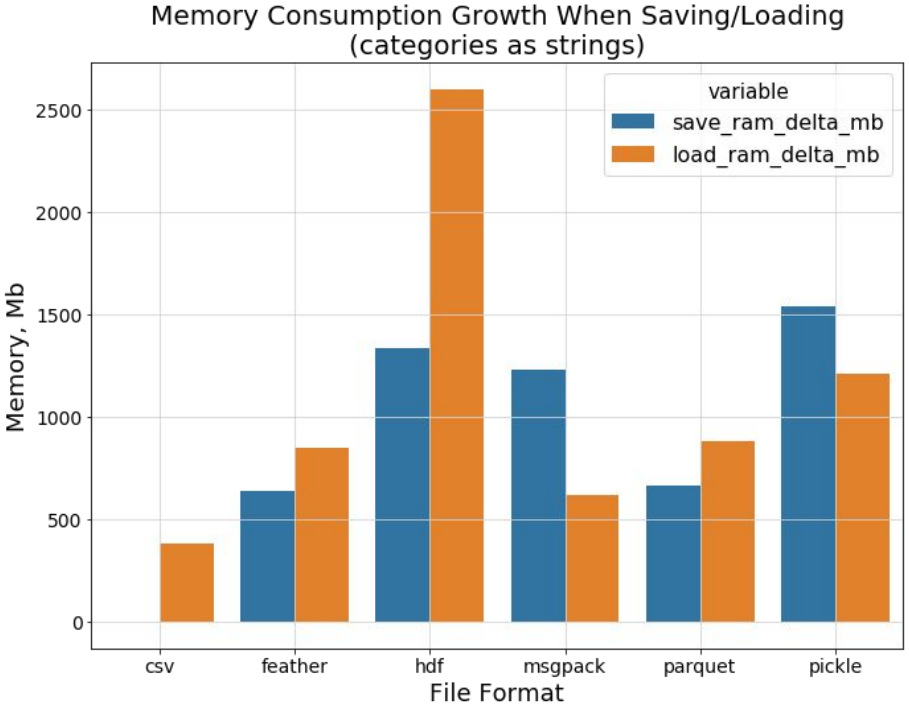
Определение основных типов данных

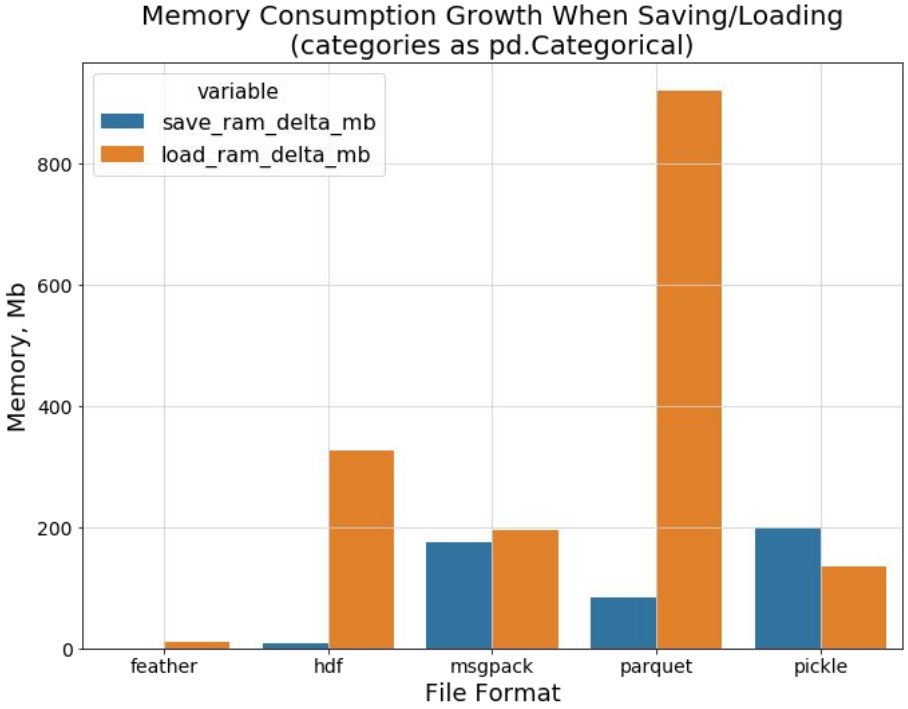
1. Plain-text CSV
2. Pickle
3. MessagePack (похож на json)
4. HDF5
5. Feather
6. Parquet

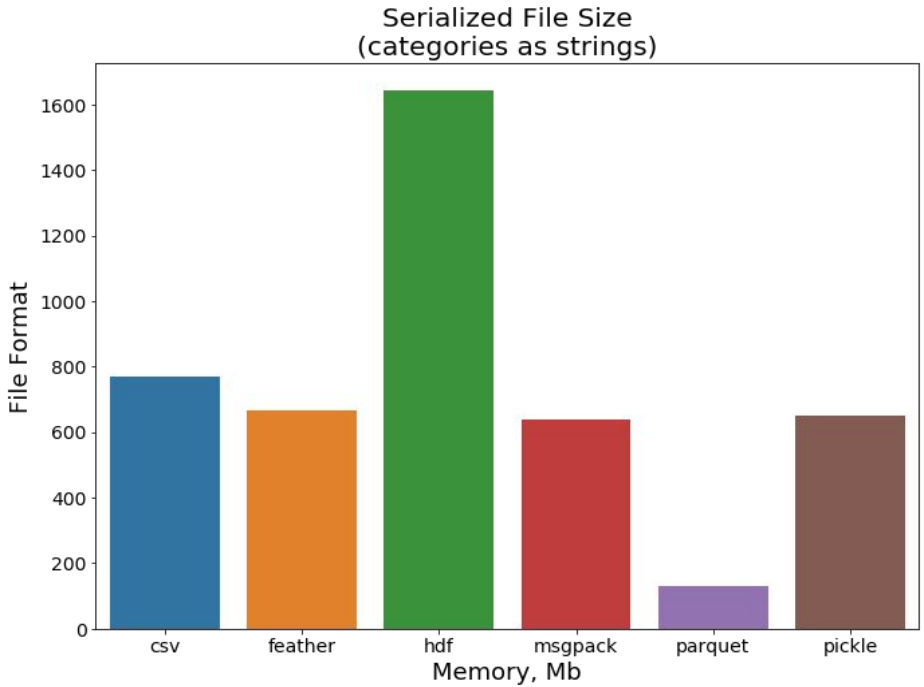

Time to Save/Load a Data Frame
(categories as strings)
variable
save_time
load_time
Seconds
0
5
10
15
20
25
csv
feather
hdf
msgpack
parquet
pickle
File Format

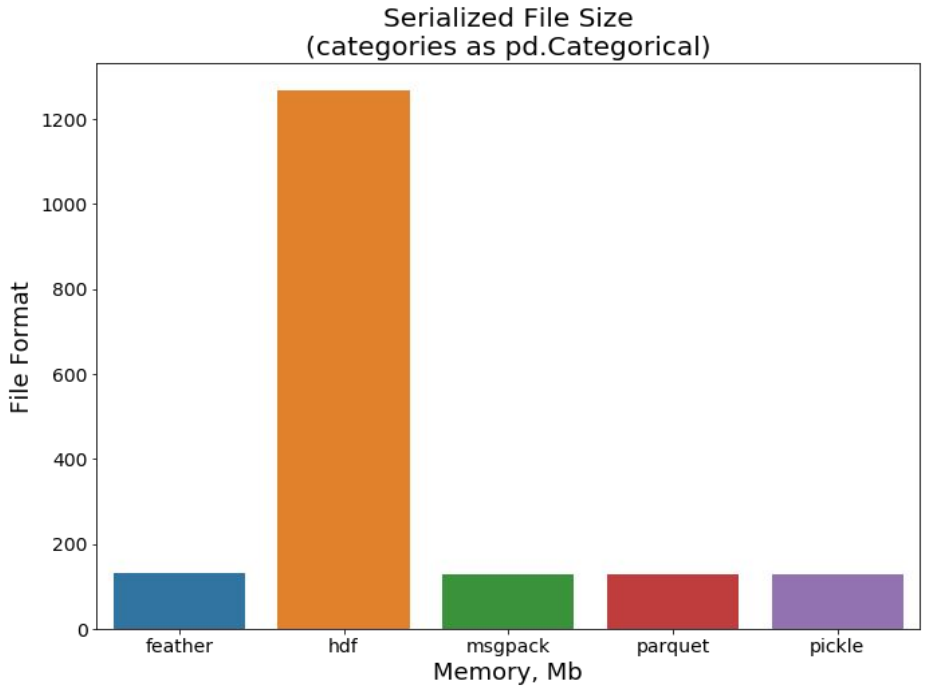

Time to Save/Load a Data Frame
(categories as pd.Categorical)
variable
save_time
load_time
Seconds
0
5
10
15
20
25
csv
feather
hdf
msgpack
parquet
pickle
File Format


Time to Save/Load a Data Frame
(categories as pd.Categorical)
Seconds
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
variable
save_time
load_time
feather
hdf
msgpack
parquet
pickle
File Format


Memory Consumption Growth When Saving/Loading
(categories as strings)
variable
save_ram_delta_mb
load_ram_delta_mb
Memory, Mb
2500
2000
1500
1000
500
0
csv
feather
hdf
msgpack
parquet
pickle
File Format


Memory Consumption Growth When Saving/Loading
(categories as pd.Categorical)
variable
save_ram_delta_mb
load_ram_delta_mb
Memory, Mb
800
600
400
200
0
feather
hdf
msgpack
parquet
pickle
File Format

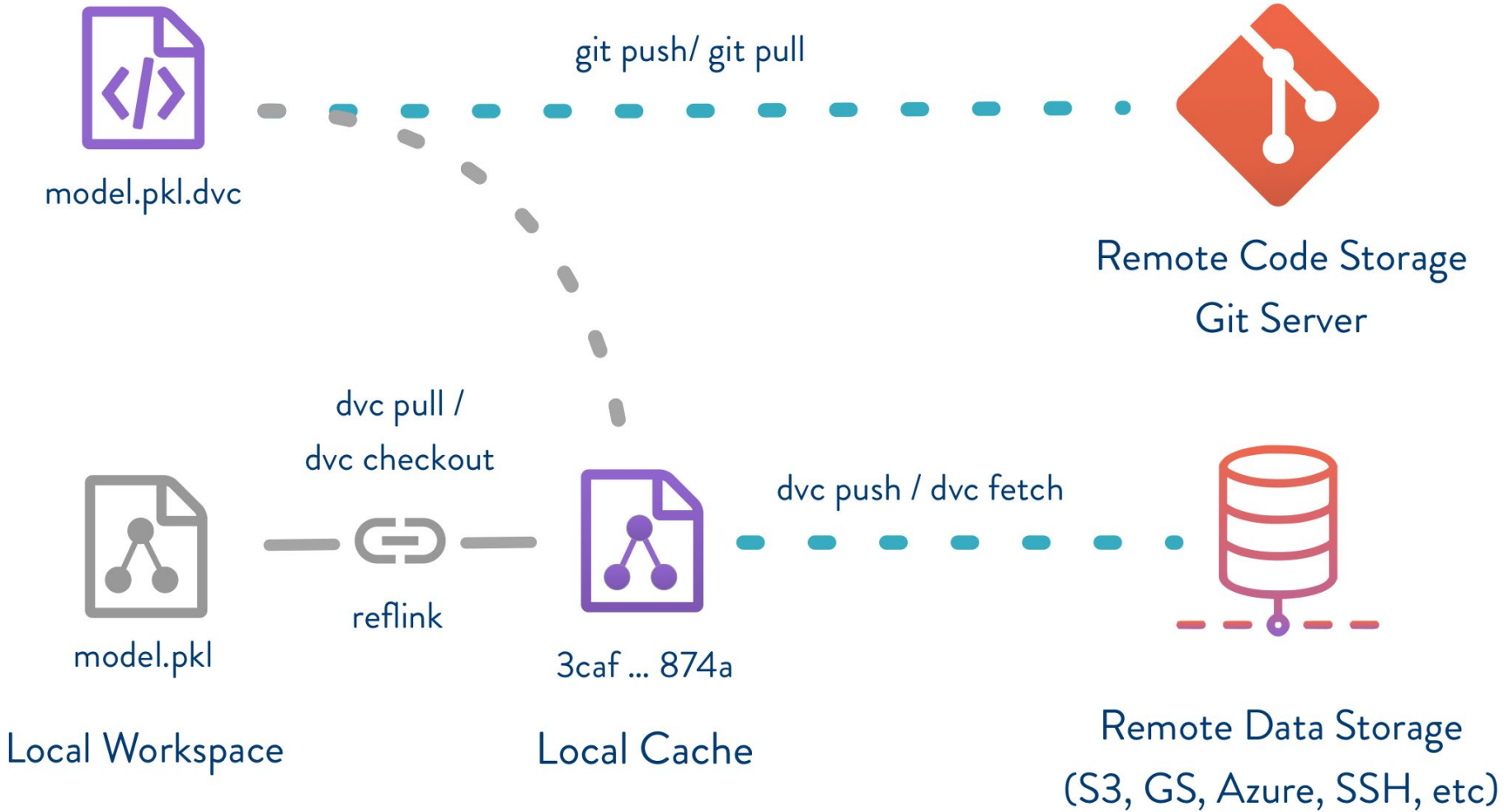




https://towardsdatascience.com/the-best-format-to-save-pandas-data-414dca023e0d

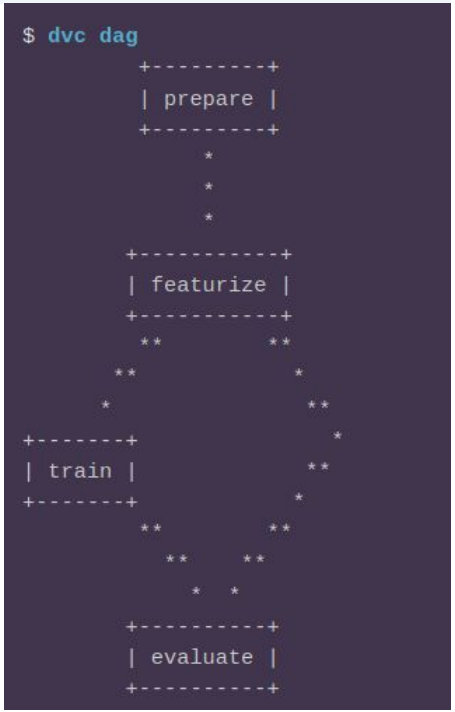
Data Version Control

Версионирование данных


git push/ git pull
model.pkl.dvc
Remote Code Storage
Git Server
dvc pull /
dvc checkout
dvc push / dvc fetch
reflink
model.pkl
3caf ... 874a
Local Workspace
Local Cache
Remote Data Storage
(S3, GS, Azure, SSH, etc)

Режимы хранения данных

1. ML Pipeline
   a. dvc run
   b. dvc repro
2. Данные из внешних источников
   a. dvc add



-d specify dependencies

-f specifies name for .dvc file to store stage metadata

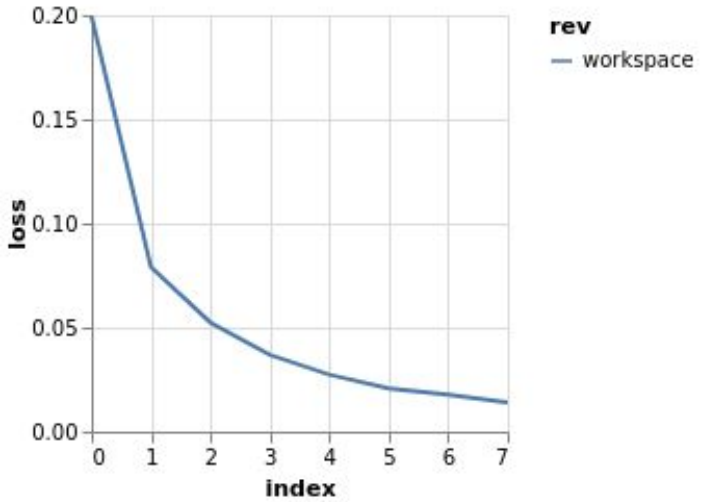
```
dvc run -f stage_feature_extraction.dvc \
    -d src/featurization.py \
    -d data/iris.csv \
    -o data/iris_featurized.csv \
    python src/featurization.py
```

python command with arguments

-o specifies outputs (data files)

```
$ dvc metrics show -a
workspace:
        eval.json:
                AUC: 0.66729
                error: 0.16982
                TP: 516
master:
        eval.json:
                AUC: 0.65115
                error: 0.17304
                TP: 528
increase_bow:
        eval.json:
                AUC: 0.66524
                error: 0.17074
                TP: 521
```

```
$ dvc plots show logs.csv -y loss
file:///Users/usr/src/plots/logs.csv.html
```



Relational Database

Теоретический минимум

Что такое реляционная база данных?

1. База данные => Таблицы => Строки и столбцы;
2. Основной интерфейс управления SQL-запросы;
3. Транзакции.

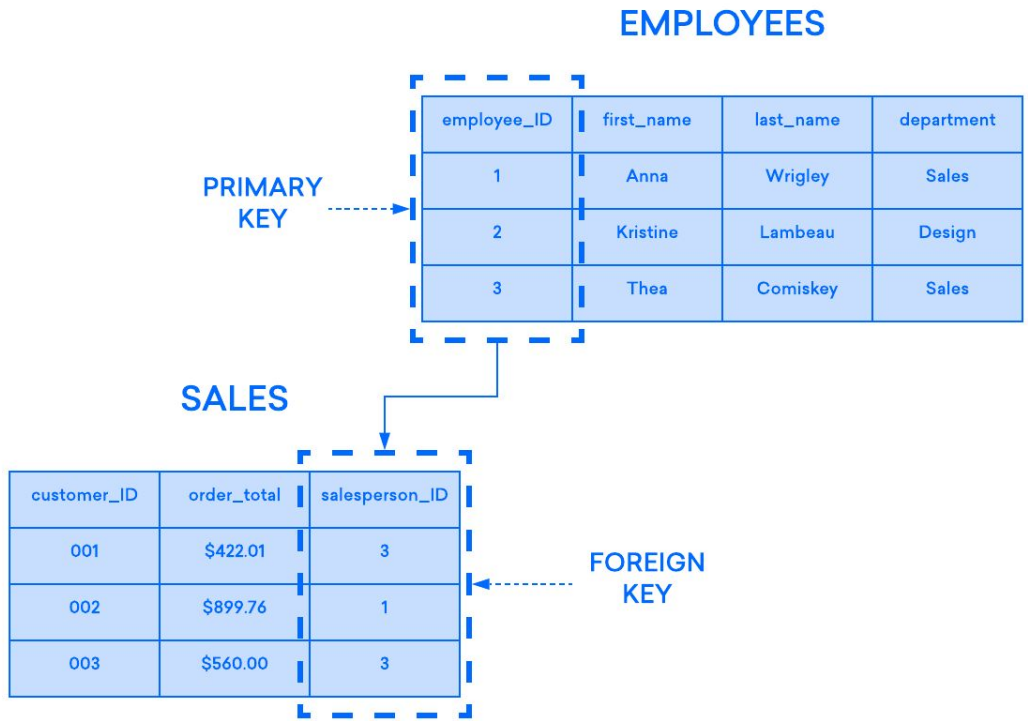
Группа анализа данных чаще всего использует из RD:

1. Oracle Database
2. PostgreSQL

Сравнение продкутов: https://db-engines.com/en/system/Oracle%3BPostgreSQL

Целостность данных – это полнота, точность и единообразие данных

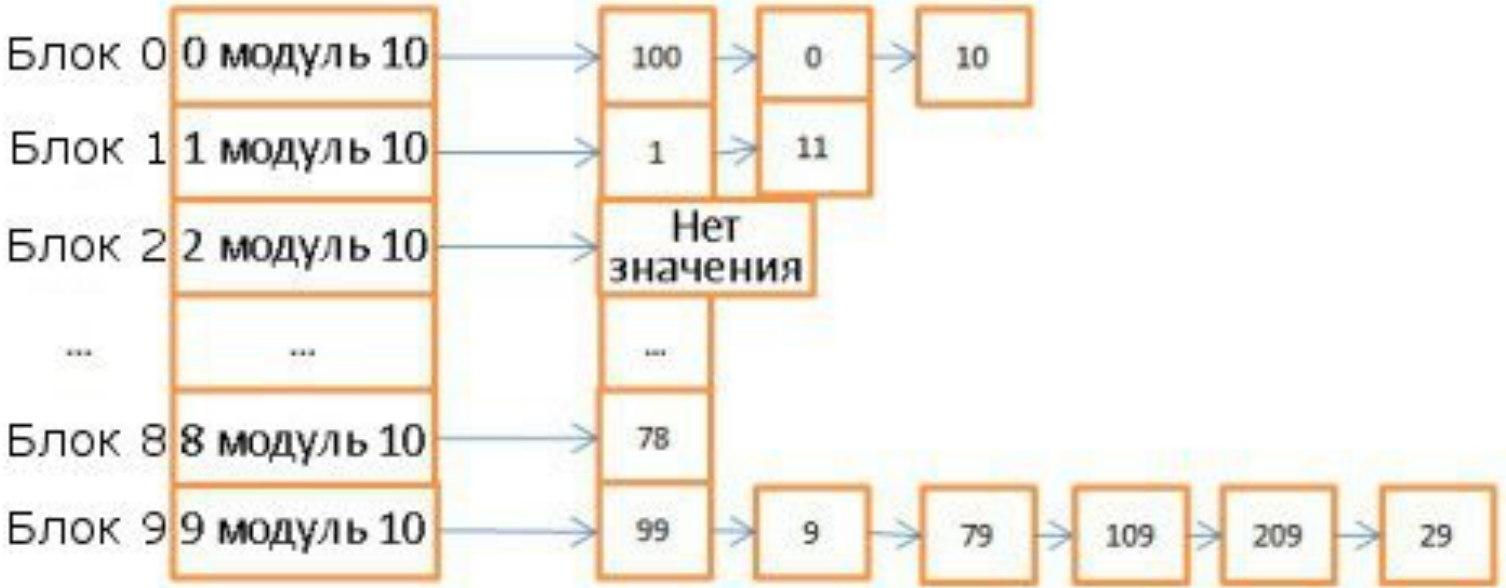
1. Первичные ключи;
2. Внешние ключи;
3. Ограничения:
   a. Not NULL
   b. Unique
   c. Default
   d. Check

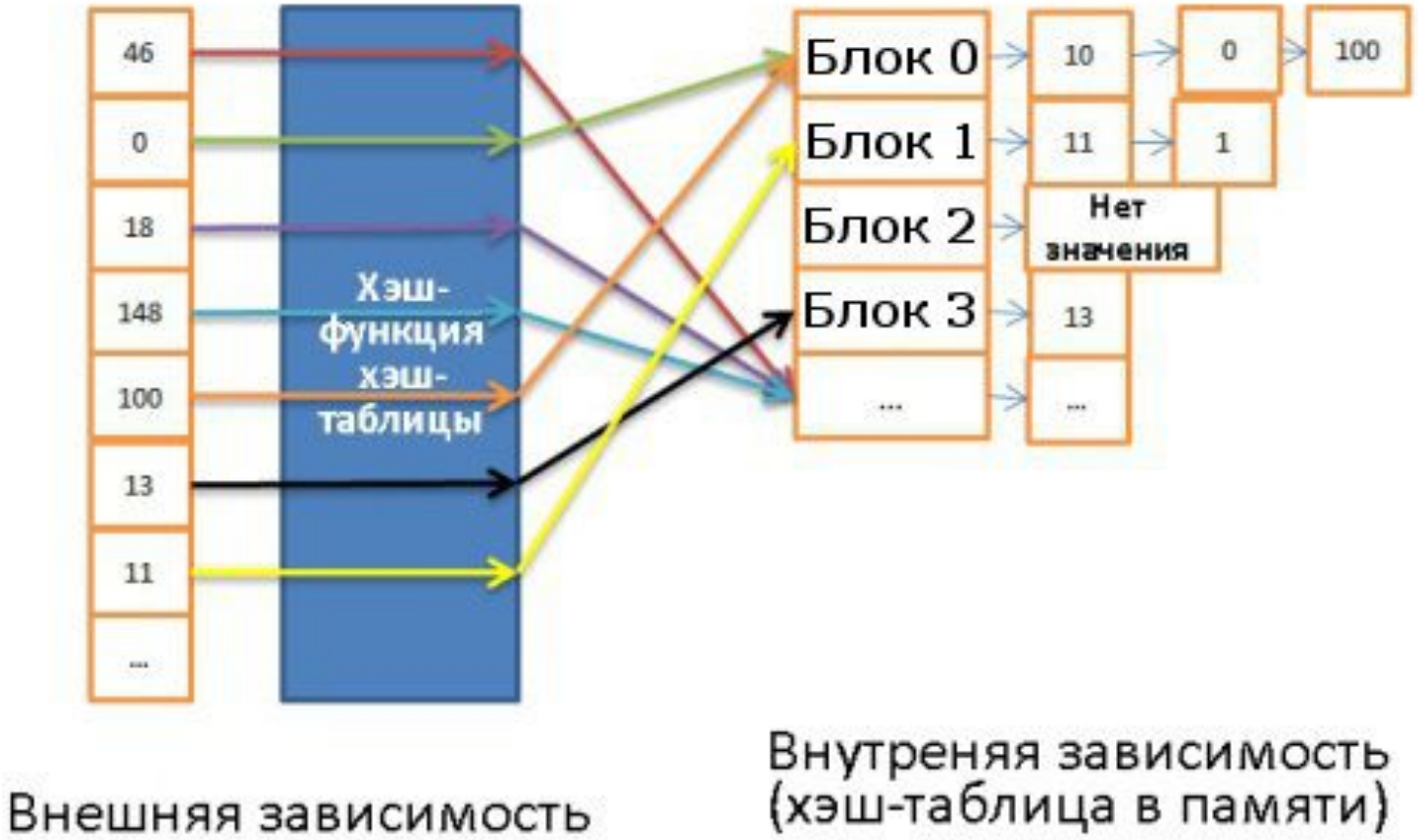


https://aws.amazon.com/ru/relational-database/

Индекс B+дерева

Хэш-таблица


Блок 0
0 модуль 10
100
0
10
Блок 1
1 модуль 10
1
11
Блок 2
2 модуль 10
Нет значения
...
...
...
Блок 8
8 модуль 10
78
Блок 9
9 модуль 10
99
9
79
109
209
29

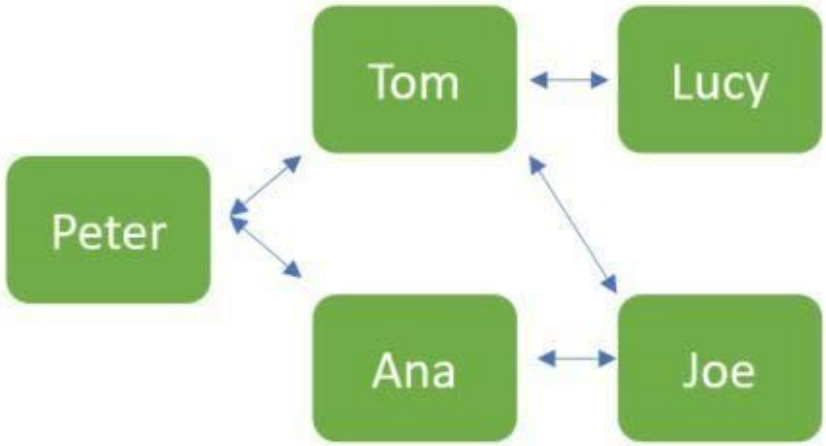


Подробнее: https://habr.com/ru/company/mailru/blog/266811/

Non-Relational Database

Теоретический минимум

SQL и NoSQL



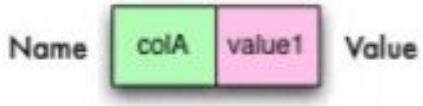
(a)

NODE	FRIENDS
Peter	{Tom, Ana}
Tom	{Joe, Lucy, Peter}
Ana	{Joe, Lucy, Peter}
Lucy	{Tom}
Joe	{Tom, Ana}

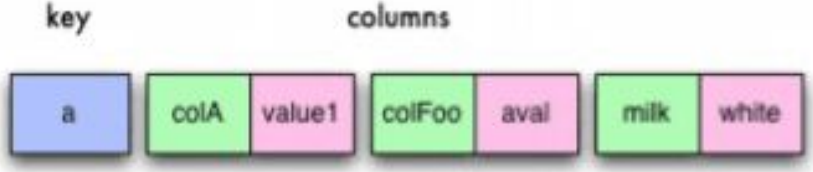
(b)

Структура данных Apache Cassandra

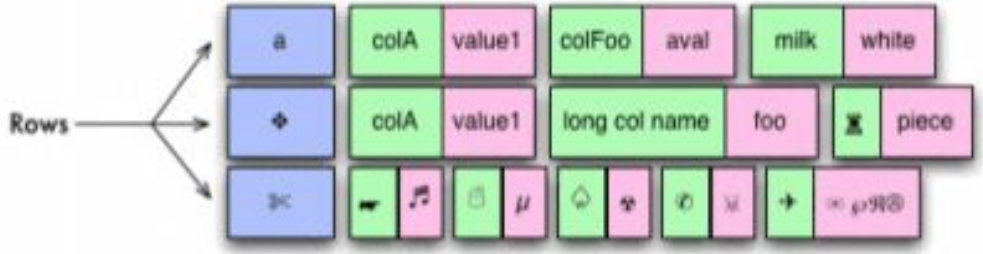
Ячейка



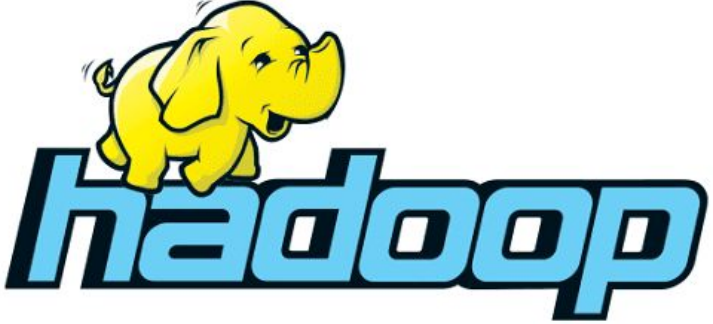
Строка



Column Family



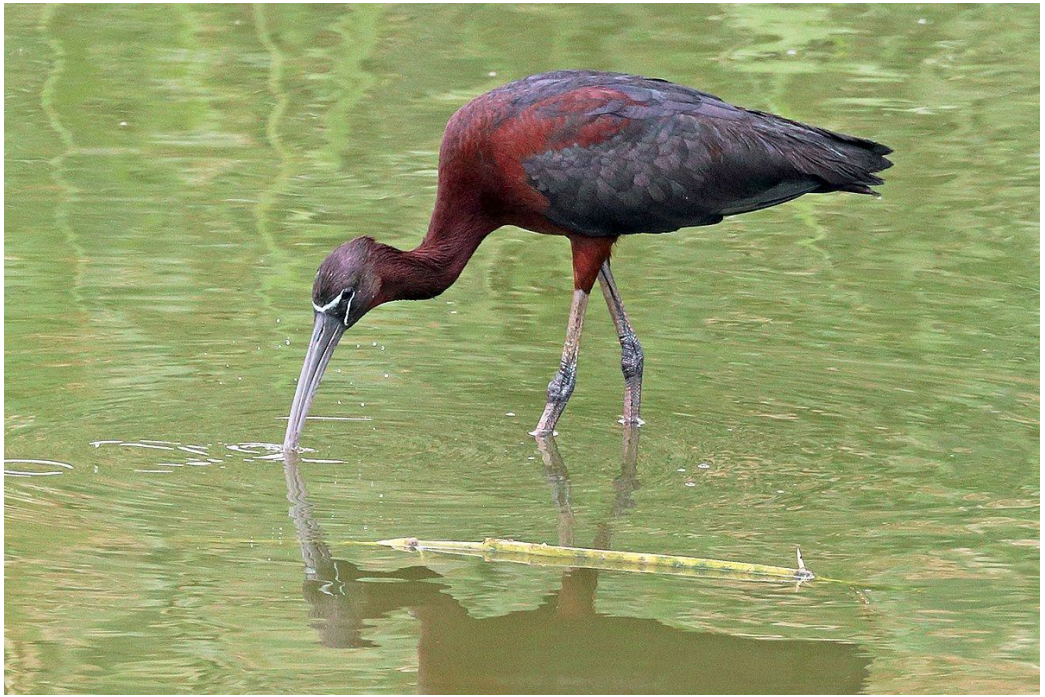
Hadoop - что там внутри?



1. **Hadoop Common** – набор инфраструктурных программных библиотек и утилит;
2. **HDFS** – распределенная файловая система, Hadoop Distributed File System;
3. **YARN** – система планирования заданий и управления кластером (Yet Another Resource Negotiator);
4. **Hadoop MapReduce** – платформа программирования и выполнения распределенных MapReduce-вычислений.

Apache Hive and Apache Impala

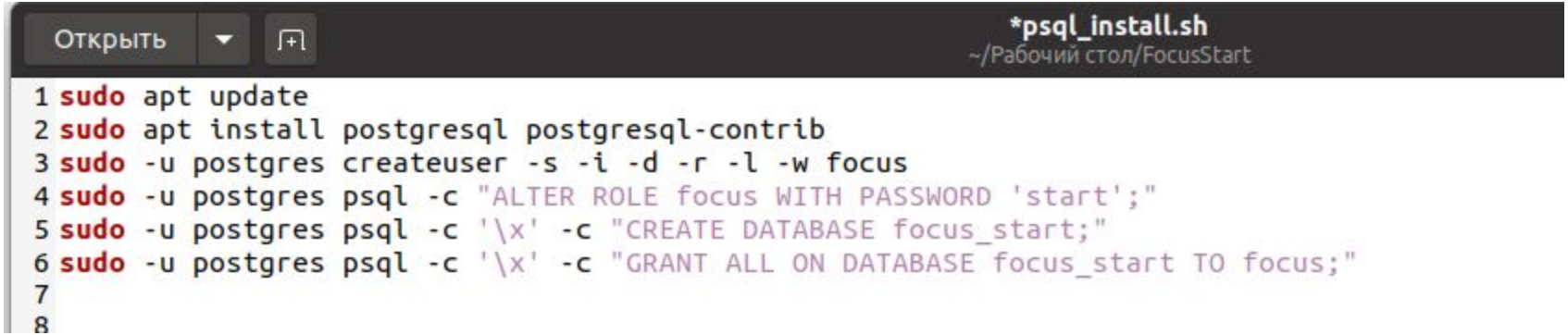
Apache Spark and Ibis



Structured Query Language

Основные моменты на примере PSQL

Установка Postgres и создание новой базы данных



ASHRAE - Great Energy Predictor III

weather_[train/test].csv

Weather data from a meteorological station as close as possible to the site.

- `site_id`
- `air_temperature` - Degrees Celsius
- `cloud_coverage` - Portion of the sky covered in clouds, in oktas
- `dew_temperature` - Degrees Celsius
- `precip_depth_1_hr` - Millimeters
- `sea_level_pressure` - Millibar/hectopascals
- `wind_direction` - Compass direction (0-360)
- `wind_speed` - Meters per second

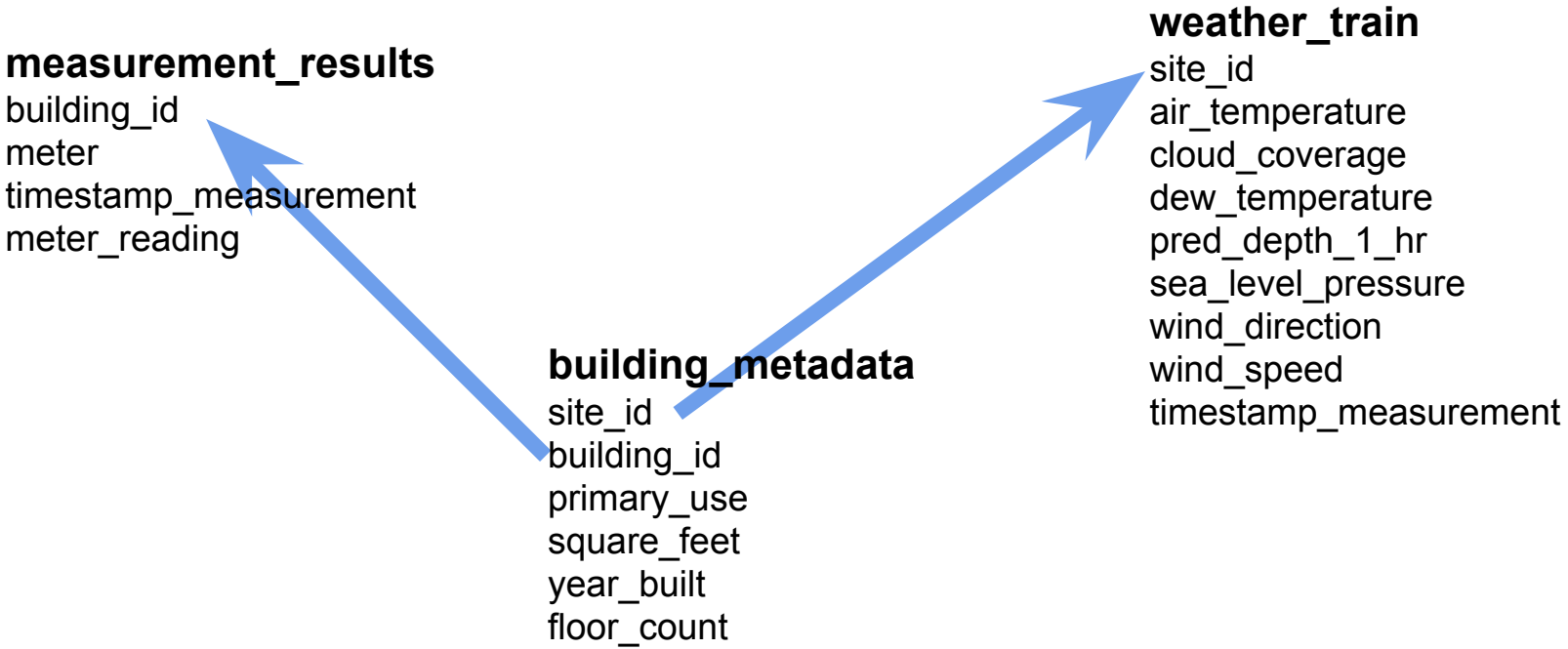
train.csv

- `building_id` - Foreign key for the building metadata.
- `meter` - The meter id code. Read as `{0: electricity, 1: chilledwater, 2: steam, 3: hotwater}`. Not every building has all meter types.
- `timestamp` - When the measurement was taken
- `meter_reading` - The target variable. Energy consumption in kWh (or equivalent). Note that this is real data with measurement error, which we expect will impose a baseline level of modeling error. UPDATE: as discussed here, the site 0 electric meter readings are in kBTU.

building_meta.csv

- `site_id` - Foreign key for the weather files.
- `building_id` - Foreign key for `training.csv`
- `primary_use` - Indicator of the primary category of activities for the building based on EnergyStar property type definitions
- `square_feet` - Gross floor area of the building
- `year_built` - Year building was opened
- `floor_count` - Number of floors of the building

Схема взаимодействия между таблицами



Создадим таблицу из файла

```
create table weather_train (
        site_id SERIAL NOT NULL,
        timestamp_measurement timestamp without time zone not null DEFAULT NULLIF('0000-00-00 00:00:00','0000-00-00
00:00:00')::timestamp,
        air_temperature float(8),
        cloud_coverage float(8),
        dew_temperature float(8),
        precip_depth_1_hr float(8),
        sea_level_pressure float(8),
        wind_direction float(8),
        wind_speed float(8)
);

COPY weather_train(
        site_id,
        timestamp_measurement,
        air_temperature,
        cloud_coverage,
        dew_temperature,
        precip_depth_1_hr,
        sea_level_pressure,
        wind_direction,
        wind_speed
)
FROM '/home/lapltop/Рабочий стол/FocusStart/weather_train.csv' DELIMITER ',' CSV HEADER;
```

Подготовка рабочей среды

1. Создаем окружение.
2. Устанавливаем туда
   - ***psycopg2-binary*** - для соединения с базой данных
   - ***pandas*** есть метод в котором реализован функционал отправки запросов к базе.
   - ***jupyter notebook*** просто для удобства =)
3. Поднимаем jupyter из окружения.

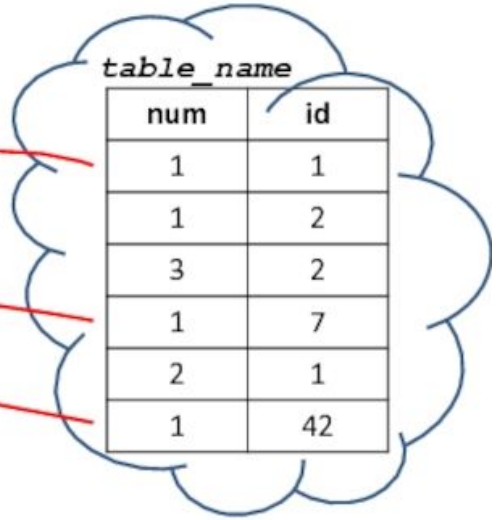
Напишем вспомогательную функцию для отправки запросов

```
def send_sql_query(query):
    params = {
        'database': 'focus_start',
        'host': 'localhost',
        'user': 'focus',
        'password': 'start'
    }
    conn = psycopg2.connect(**params)
    raw_data = pd.read_sql_query(query, conn)
    conn.close()
    return raw_data
```

Основные команды SQL

```
SELECT * FROM table_name
  WHERE num = 1 AND (id > 5 OR id
 < 2 );
```

```
+------+------+
| num  | id   |
+------+------+
|    1 |    1 |
|    1 |    7 |
|    1 |   42 |
+------+------+
```

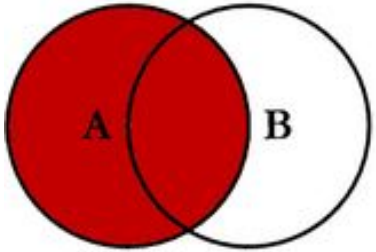


Давайте испытаем нашу функцию!

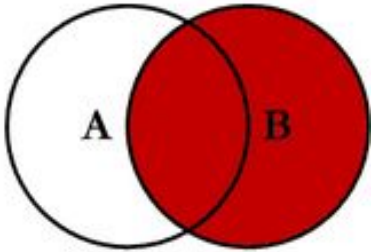
Задача

Посчитайте количество зданий, где установлено 2 и более типа счетчиков.

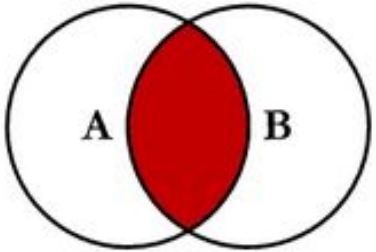
SQL JOINS



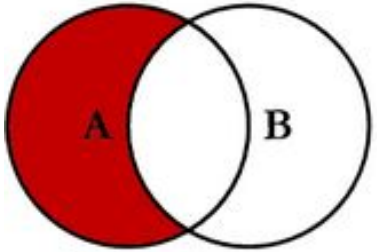

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



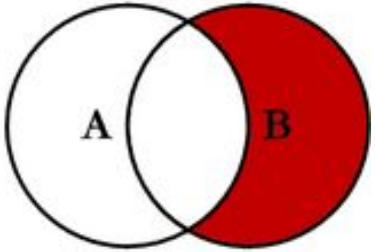

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



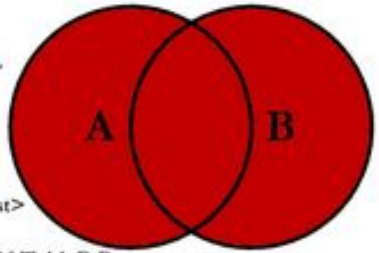

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



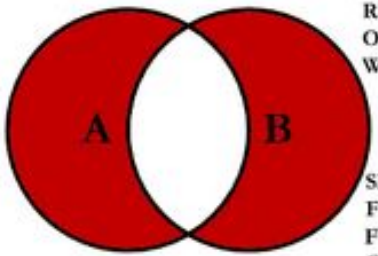

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Задача:

Представим, что мы решили построить модель на этих данных и хотим собрать все признаки в кучу за последние день 2016 года. (если менее 5гб свободной оперативки, можете добавить ещё фильтры к основной таблице)

Соединить между собой все три таблицы (***measurement_results, building_metadata, weather_train***), с фильтром по дате от 30 декабря 2016 года.

*Учитывайте, что **measurement_results** является основной таблицей, где находится целевая переменная.*

Задача с WITH

Задача

1. *Давайте по генерируем гипотезы на основе наших данных! (не более 2ух - **не нужно придумывать, те которые сложно посчитать**).*
2. *Попробуем используя конструкцию WITH в одном запросе получить:*
   2.1 Основную часть данных, которая состоит из:
      - идентификатор здания
      - тип счетчика
      - целевая переменная (показания счетчика)
   2.2 Запрос насчитывающий признак №1
   2.3 Запрос насчитывающий признак №2
3. *Объединить все в один набор данных.*

Оконные функции бывают...

- Агрегатные функции;
- Ранжирующие функции;
- Функции смещения;
- Аналитические функции.

Обычный запрос

Запрос с оконной функцией

http://thisisdata.ru/blog/uchimsya-primenyat-okonnyye-funktsii/?utm_source=telegram.org&utm_medium=referral

Агрегатные функции

- **SUM** – возвращает сумму значений в столбце;
- **COUNT** — вычисляет количество значений в столбце (*значения NULL не учитываются*);
- **AVG** — определяет среднее значение в столбце;
- **MAX** — определяет максимальное значение в столбце;
- **MIN** — определяет минимальное значение в столбце.

Кумулятивная сумма

Задание

Посчитайте накопительную сумму(используя оконную функцию) для каждого здания и типа счетчика.

Ранжирующие функции

- **ROW_NUMBER** – функция возвращает номер строки и используется для нумерации;
- **RANK** — функция возвращает ранг каждой строки. В данном случае значения уже анализируются и, в случае нахождения одинаковых, возвращает одинаковый ранг с пропуском следующего значения;
- **DENSE_RANK** — функция возвращает ранг каждой строки. Но в отличие от функции RANK, она для одинаковых значений возвращает ранг, не пропуская следующий;
- **NTILE** – это функция, которая позволяет определить к какой группе относится текущая строка. Количество групп задается в скобках.

Drop duplicates and keep = last

Задание

*Напишите запрос(используя оконные функции), который уберет все дублирующие записи по полям **building_id, meter** и вернет самое **новое** значение.*

Функции смещения

- **LAG или LEAD** – функция LAG обращается к данным из предыдущей строки окна, а LEAD к данным из следующей строки. Функцию можно использовать для того, чтобы сравнивать текущее значение строки с предыдущим или следующим. Имеет три параметра: столбец, значение которого необходимо вернуть, количество строк для смещения (*по умолчанию 1*), значение, которое необходимо вернуть если после смещения возвращается значение NULL;
- **FIRST_VALUE** или **LAST_VALUE** — с помощью функции можно получить первое и последнее значение в окне. В качестве параметра принимает столбец, значение которого необходимо вернуть.

Аналитические функции

- **CUME_DIST** — вычисляет интегральное распределение (относительное положение) значений в окне;
- **PERCENT_RANK** — вычисляет относительный ранг строки в окне;
- **PERCENTILE_CONT** — вычисляет процентиль на основе постоянного распределения значения столбца. В качестве параметра принимает процентиль, который необходимо вычислить;
- **PERCENTILE_DISC** — вычисляет определенный процентиль для отсортированных значений в наборе данных. В качестве параметра принимает процентиль, который необходимо вычислить.

Кастомные функции

```
CREATE FUNCTION is_more_average(building bigint, meter_type bigint, meter_reading_float float(8))
RETURNS boolean AS $$
DECLARE average_meter_reading float(8);
DECLARE answer boolean;

BEGIN
        SELECT
            AVG(mr.meter_reading) INTO average_meter_reading
        FROM measurement_results as mr
        WHERE mr.meter = $1 AND mr.building_id = $2;

        if average_meter_reading <= $3 then
                SELECT TRUE INTO answer;
        else
                SELECT FALSE INTO answer;
        end if;
        RETURN answer;
END;
$$ LANGUAGE PLpgSQL;
```