Раздел дисциплины/темы
Раздел 1. Алгоритмы и технологии анализа больших данных
Введение в науку о данных
Алгоритмы первичной обработки больших данных
Структуры данных в Python и библиотеки для работы с данными в Python
Визуализация Big Data
Анализ и преобразование больших данных, библиотеки Python
Работа с временными рядами в библиотеках Python
Раздел 2. Хранилища данных и технологии работы с ними
Системы управления базами данных
Использование языка SQL (DDL) для создания структур данных
Запросы на языке SQL (DML)
Объекты баз данных
NoSQL хранилища данных
Хранилища больших данных класса ключ-значение
Документоориентированные хранилища больших данных
Графовые хранилища больших данных
Раздел 3. Интеллектуальные платформы анализа больших данных
Облачные платформы интеллектуального анализа данных
Разработка методов, алгоритмов и программных средств интеллектуального анализа больших данных: Hadoop, Spark и IoT
Эффективность и отказоустойчивость платформ интеллектуального анализа больших данных

РАЗДЕЛ 1
АЛГОРИТМЫ И ТЕХНОЛОГИИ АНАЛИЗА БОЛЬШИХ ДАННЫХ

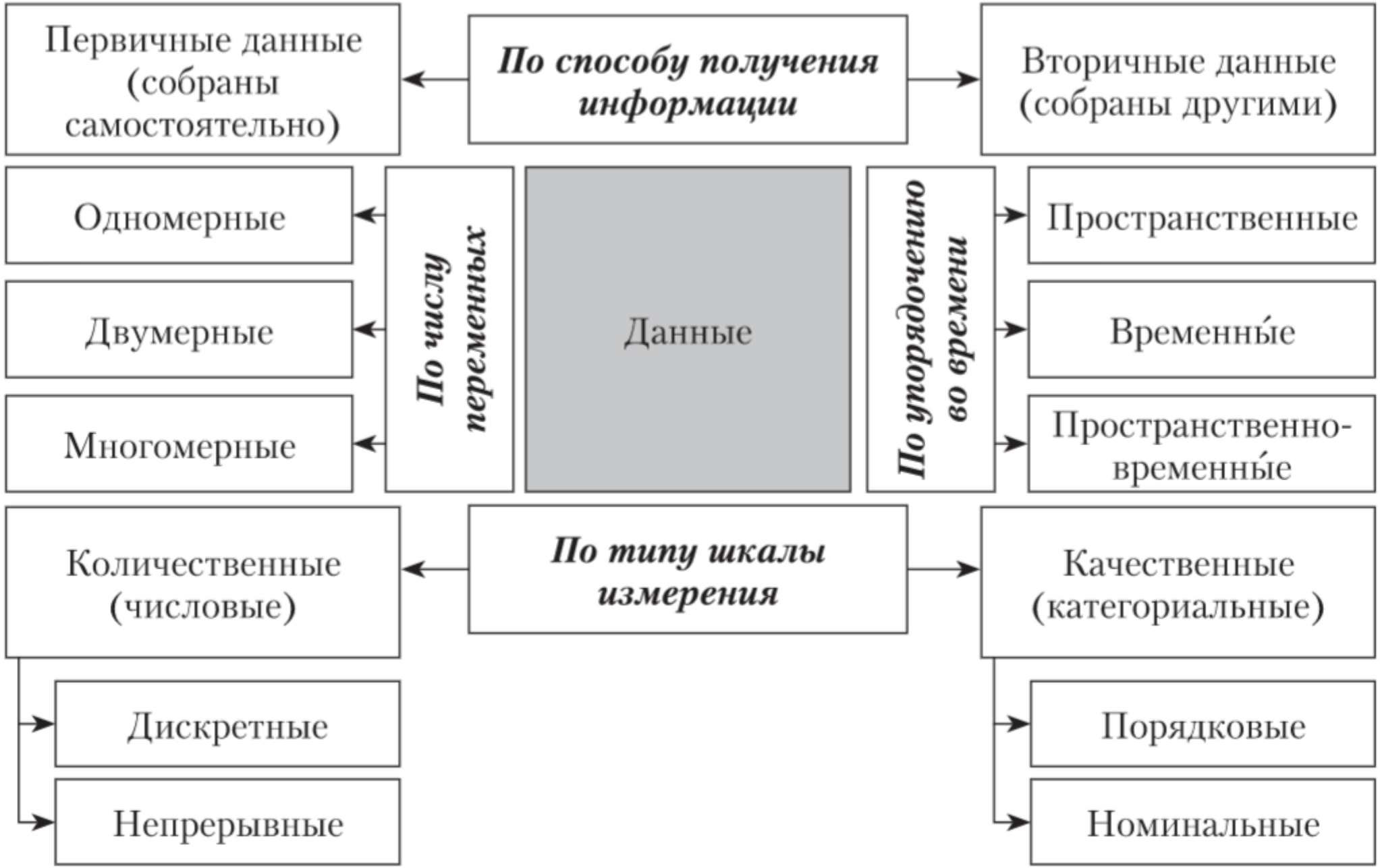
Лекция 1. Введение в науку о данных

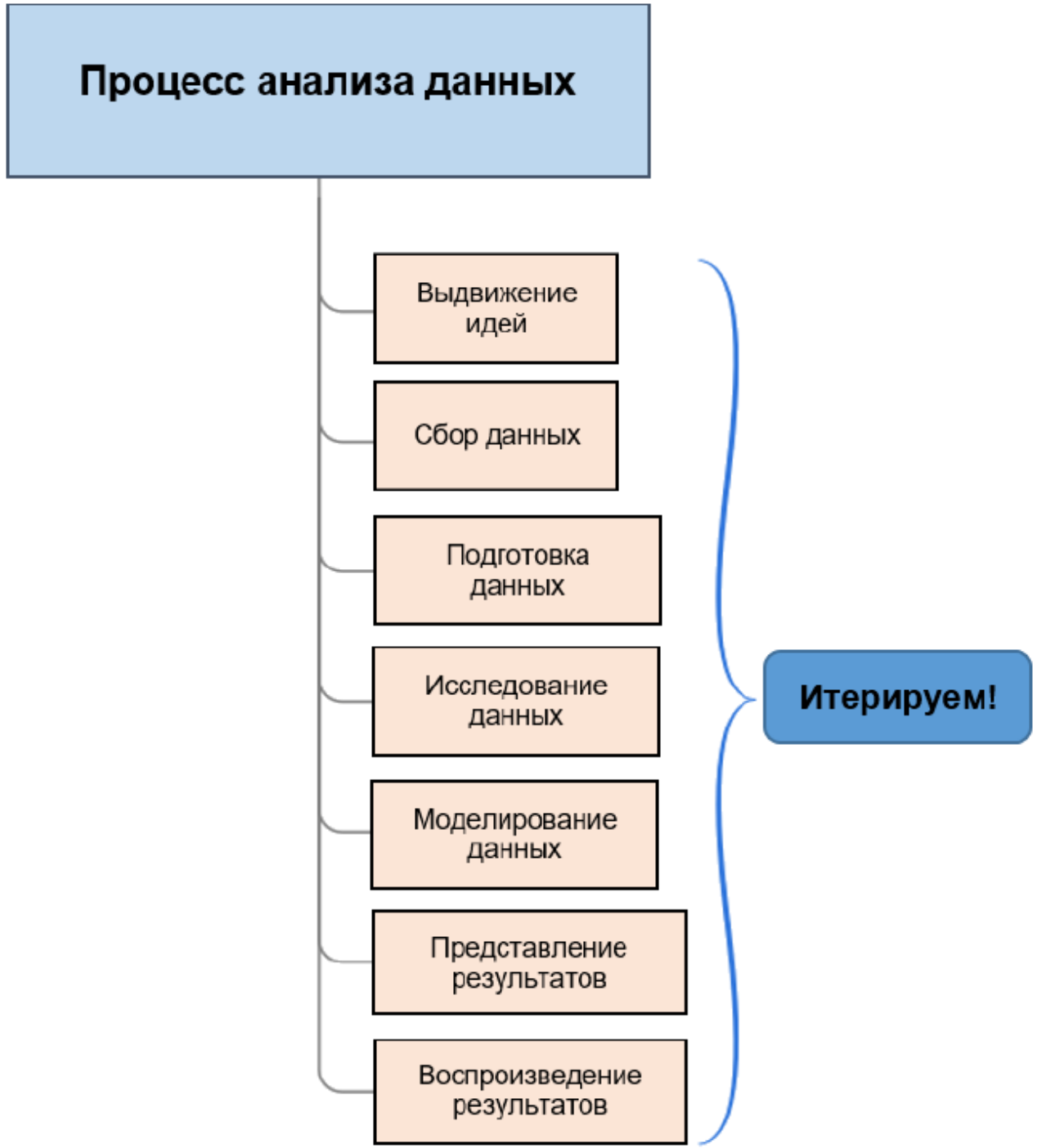
Анализ данных

Анализ данных – это процесс извлечения смысла из данных. Данные, представленные в количественном виде, часто называют **информацией (information)**. Анализ данных – это процесс получения информации из данных путем создания моделей и применения математического аппарата для поиска закономерностей. Он часто перекликается с обработкой данных и не всегда можно четко провести различие между ними. Многие инструменты обработки данных также содержат аналитические функции, а инструменты анализа данных часто предлагают возможности обработки данных.

Наука о данных

Наука о данных – это процесс использования статистики и анализа данных для понимания **явлений (phenomena)**, скрытых в данных. Наука о данных обычно начинается с информации и применяет к ней более сложный анализ на основе знаний, относящихся к разным предметным областям. К этим предметным областям относятся математика, статистика, информатика, компьютерные науки, машинное обучение, классификация, кластерный анализ, интеллектуальный анализ данных, базы данных и визуализация. Наука о данных носит многодисциплинарный характер. Ее методы анализа могут сильно отличаться друг от друга и зависеть от конкретной предметной области.


Данные
По способу получения информации
Первичные данные (собраны самостоятельно)
Вторичные данные (собраны другими)
По числу переменных
Одномерные
Двумерные
Многомерные
По упорядочению во времени
Пространственные
Временны́е
Пространственно-временны́е
По типу шкалы измерения
Количественные (числовые)
Дискретные
Непрерывные
Качественные (категориальные)
Порядковые
Номинальные


Процесс анализа данных
Выдвижение идей
Сбор данных
Подготовка данных
Исследование данных
Моделирование данных
Представление результатов
Воспроизведение результатов
Итерируем!

Структурированные данные

Структурированные данные – это тип данных, при котором последние упорядочены в вертикальные столбцы (поля) и горизонтальные строки (записи или наблюдения). Примером таких данных являются данные в реляционных базах и электронных таблицах. Структурированные данные зависят от модели данных, которая представляет собой определенную структуру, содержательного смысла данных и часто от способа обработки данных. Сюда входит идентификация типа данных (целое число, число с плавающей точкой, строка и т.д.) и ограничения, накладываемые на данные, например, количество символов, максимальное и минимальное значения, или ограничение на определенный набор значений.

Неструктурированные данные

Неструктурированные данные – это данные, которые не имеют определенной структуры, не предполагают наличия заранее определенных столбцов определенного типа. Примером неструктурированных данных могут быть такие виды информации, как фотографии и графические изображения, видеоролики, потоковые данные датчиков, веб-страницы, PDF-файлы, презентации PowerPoint, электронные письма, записи в блогах, страницы Википедии и документы Word.

Одномерный и многомерный анализ

В каком-то смысле статистика представляет собой практику изучения переменных и, в частности, наблюдение за этими переменными. Статистика преимущественно опирается на анализ одной переменной, которая называется одномерным анализом. **Одномерный анализ (univariate analysis)** – это простейшая форма анализа данных. Он не имеет отношения к анализу причин или взаимосвязей и обычно используется для описания или подытоживания данных, а также поиска закономерностей в данных.

Многомерный анализ (multivariate analysis) – это метод моделирования, в котором участвуют две или более переменных, которые влияют на результат эксперимента. Многомерный анализ часто связан с такими понятиями, как корреляция и регрессия, которые помогают нам понять взаимосвязь между несколькими переменными, а также влияние этой взаимосвязи на результат.

Описательные статистики

Описательные статистики – это показатели, которые подытоживают данные, обычно в тех случаях, где набор данных представляет собой генеральную совокупность или выборку из одной переменной (одномерные данные). Эти показатели описывают набор данных и являются мерами центральной тенденции, а также мерами изменчивости и дисперсии.

Например, следующие показатели являются описательными статистиками:

- Распределение (например, нормальное, пуассоновское)
- Центральная тенденция (например, среднее, медиана и мода)
- Дисперсия (например, дисперсия, стандартное отклонение)

Индуктивные статистики

Индуктивные статистики отличаются от описательных тем, что с их помощью мы пытаемся сделать выводы о данных, а не просто подытожить данные. Примерами индуктивных статистик являются:

- t-тест
- хи-квадрат
- ANOVA
- бутстреп

Стохастические модели

Стохастические модели представляют собой вид статистического моделирования, который включает в себя одну или несколько случайных величин, а также подразумевает использование временных рядов. Цель стохастической модели – оценить вероятность того, что результат будет находиться в пределах определенного интервала, и спрогнозировать условия для разных ситуаций.

Лекция 2. Алгоритмы обработки первичной обработки больших данных

Знания – это совокупность классифицированных сведений о мире, объектах, процессах и явлениях

Концептуальные знания получают путем анализа, обобщения и рассуждений

Фиксируются в виде текстовых описаний

Эмпирические знания получают путем наблюдения и измерения

Фиксируются в виде числовых и графических описаний

Знания в символическом искусственном интеллекте получают с помощью математико-логических рассуждений

Достоверные рассуждения используют строгие математико-логические аксиомы и правила вывода

Результат: строгое принятие (или непринятие) решений

Правдоподобные рассуждения используют математические подходы, допускающие оценку уверенности вывода

Результат: принятие (или непринятие) решений с долей неуверенности

В искусственном интеллекте используются правдоподобные рассуждения

Схема правдоподобного рассуждения с вероятностной оценкой уверенности:

Вероятность, что A→B больше q
Вероятность, что A истинно больше r

Вероятность, что B истинно больше max (0, q+r-1)

Пример:

Вероятность, что A→B больше 0.9
Вероятность, что A истинно больше 0.5

Вероятность, что B истинно больше 0.4

Знания в вычислительном искусственном интеллекте извлекаются из БОЛЬШИХ ДАННЫХ

Закономерности в БОЛЬШИХ ДАННЫХ изучаются НАУКОЙ О ДАННЫХ

Разнообразие

Скорость

Объем

НАУКА О ДАННЫХ

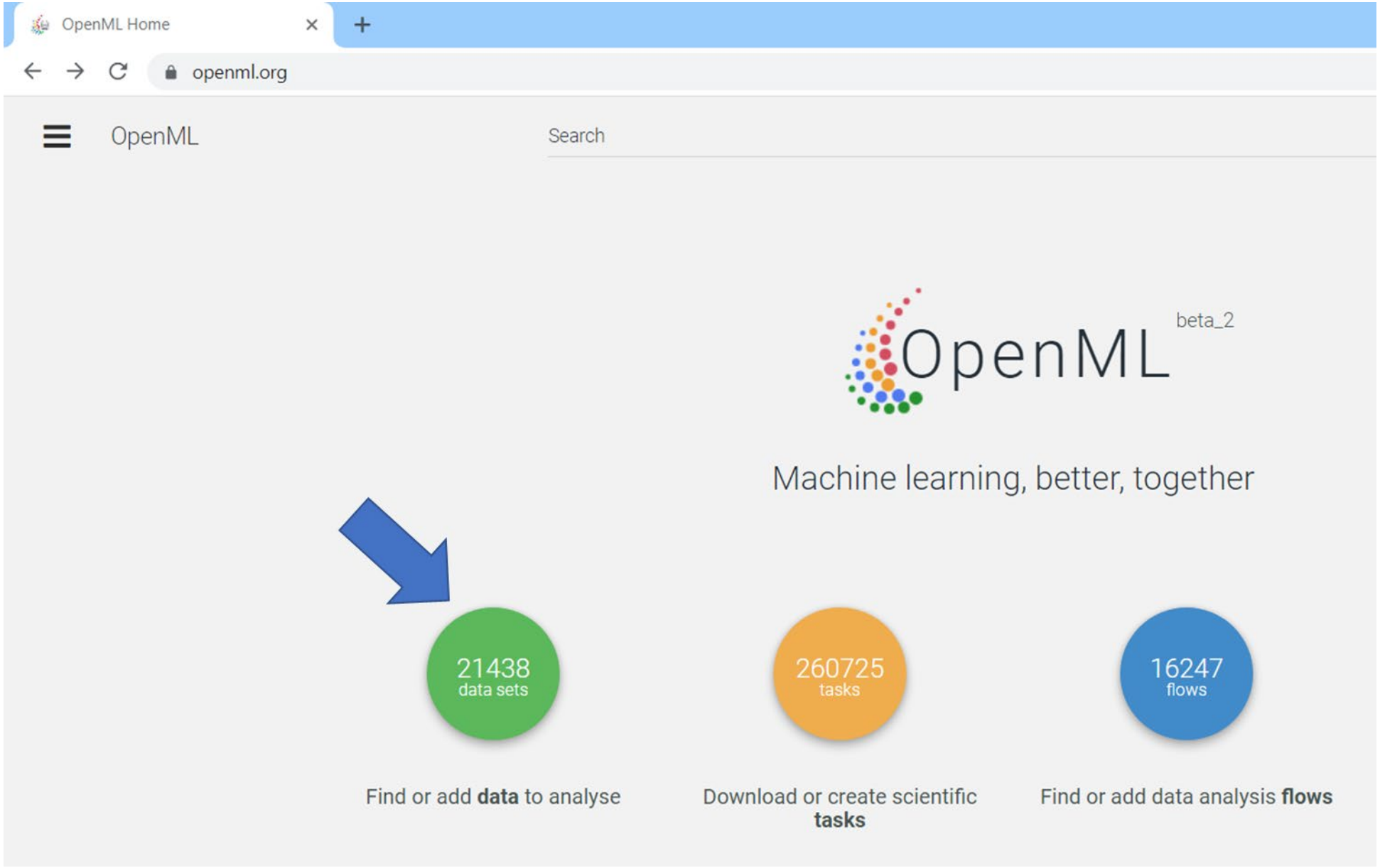
ДОБЫЧА ДАННЫХ
(DATA MINING)

АНАЛИТИКА ДАННЫХ
(DATA ANALYTICS)

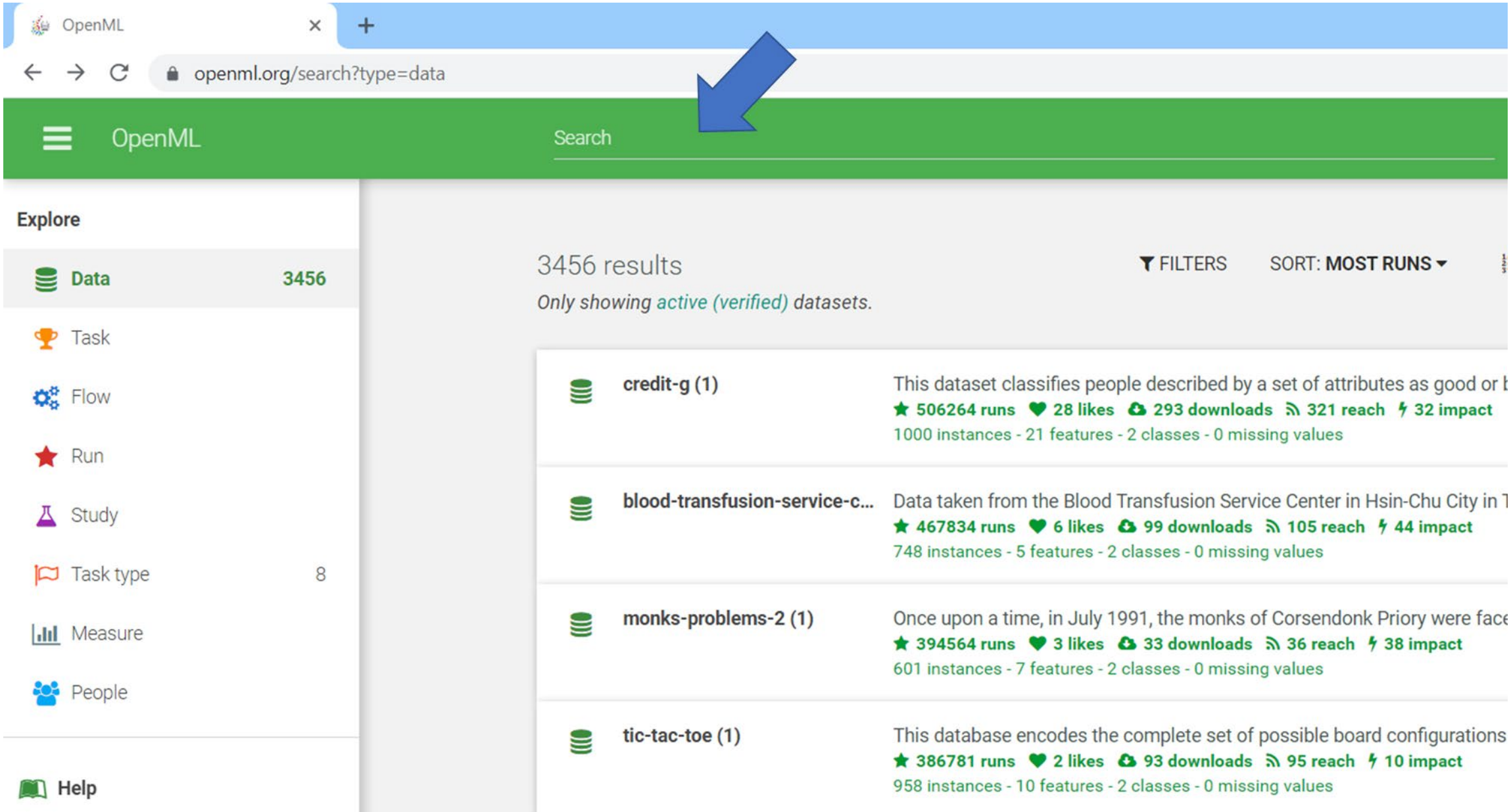
ОТКРЫТЫЕ РЕПОЗИТОРИИ ДАННЫХ ДЛЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Реестр исследовательских репозиториев	*https://www.re3data.org*
Репозиторий Open Machine Learning	*https://www.openml.org*
Репозиторий UCI Machine Learning	*https://archive.ics.uci.edu/ml/index.php*
Репозиторий Scientific Data Repository	*https://www.mlvis.com/*

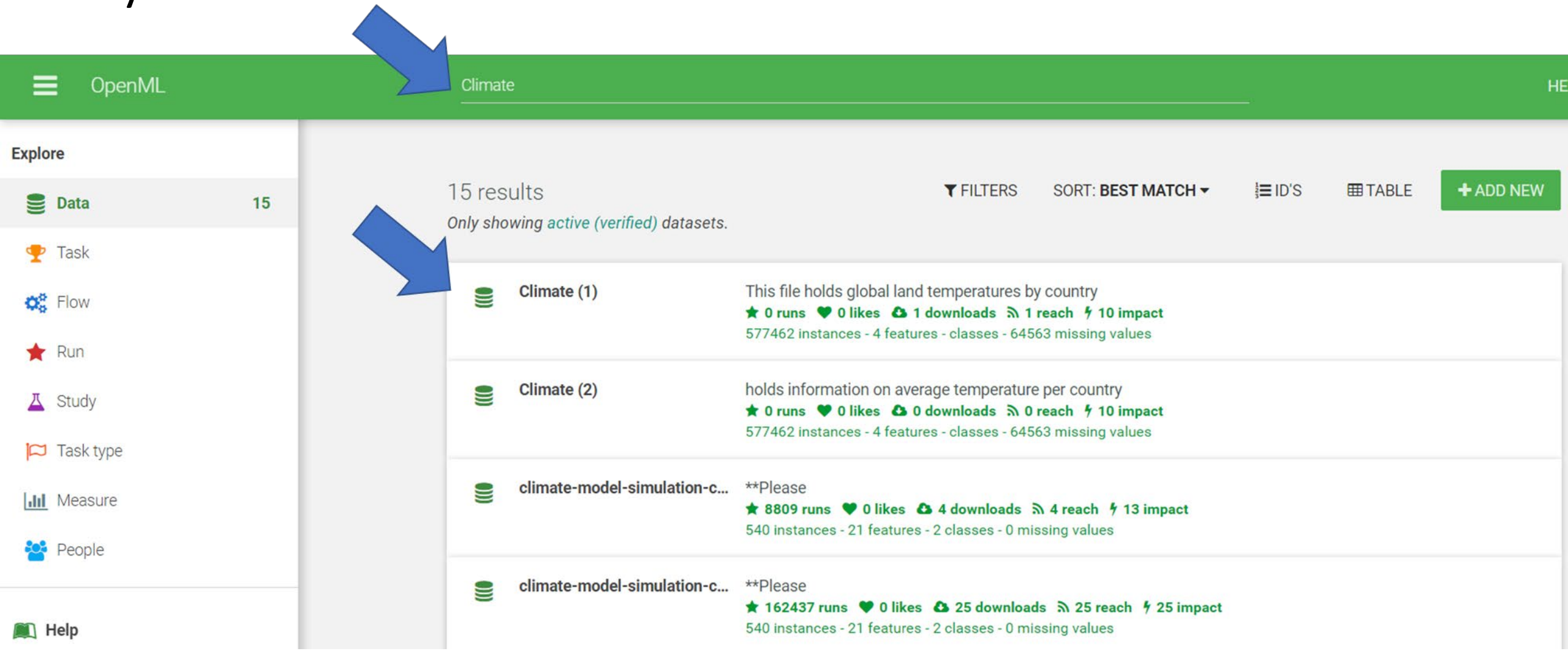
РЕПОЗИТОРИЙ OPENML.ORG



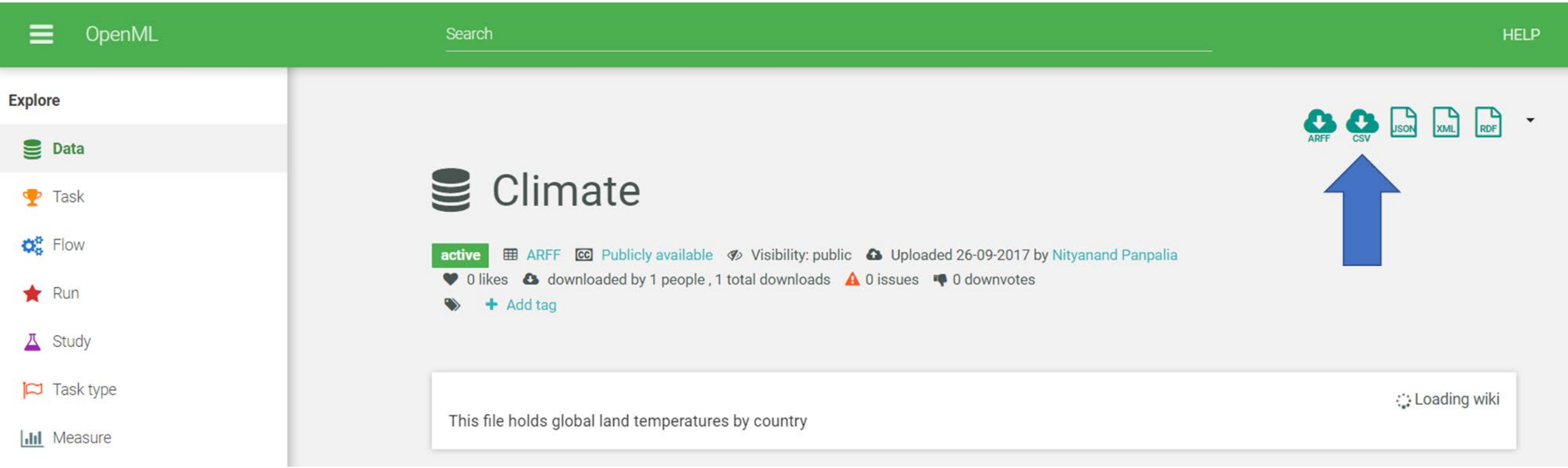
Поиск наборов данных в репозитории



Результат поиска тематики Climate

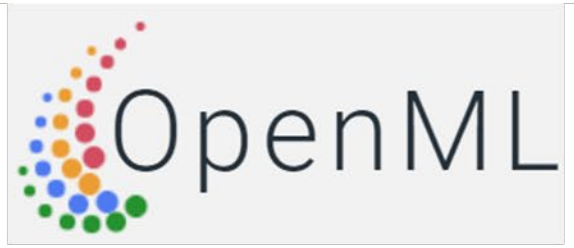


Выбор формата данных Climate



Получение данных

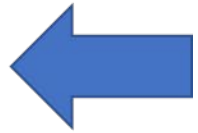
Язык и среда разработки



Программные библиотеки

```
!pip install openml
```

```
import openml as oml
import pandas as pd

dataset = oml.datasets.get_dataset(40918)
X, y, category, attributes = dataset.get_data(target = dataset.default_target_attribute,
                                              dataset_format="dataframe")
```



Получение информации о наборе данных

```
print(X.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 577462 entries, 0 to 577461
Data columns (total 4 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   dt                             577462 non-null  object 
 1   AverageTemperature             544811 non-null  float64
 2   AverageTemperatureUncertainty  545550 non-null  float64
 3   Country                        577462 non-null  object 
dtypes: float64(2), object(2)
memory usage: 17.6+ MB
None
```

Получение информации о наборе данных

```
print(X.head(10))
```

	dt	AverageTemperature	AverageTemperatureUncertainty	Country
0	1743-11-01	4.384	2.294	Åland
1	1743-12-01	NaN	NaN	Åland
2	1744-01-01	NaN	NaN	Åland
3	1744-02-01	NaN	NaN	Åland
4	1744-03-01	NaN	NaN	Åland
5	1744-04-01	1.530	4.680	Åland
6	1744-05-01	6.702	1.789	Åland
7	1744-06-01	11.609	1.577	Åland
8	1744-07-01	15.342	1.410	Åland
9	1744-08-01	NaN	NaN	Åland

Лекция 3. Алгоритмы обработки первичной обработки больших данных

Знания – это совокупность классифицированных сведений о мире, объектах, процессах и явлениях

Концептуальные знания получают путем анализа, обобщения и рассуждений

Фиксируются в виде текстовых описаний

Эмпирические знания получают путем наблюдения и измерения

Фиксируются в виде числовых и графических описаний

Знания в символическом искусственном интеллекте получают с помощью математико-логических рассуждений

Достоверные рассуждения используют строгие математико-логические аксиомы и правила вывода

Результат: строгое принятие (или непринятие) решений

Правдоподобные рассуждения используют математические подходы, допускающие оценку уверенности вывода

Результат: принятие (или непринятие) решений с долей неуверенности

В искусственном интеллекте используются правдоподобные рассуждения

Схема правдоподобного рассуждения с вероятностной оценкой уверенности:

Вероятность, что A→B больше q
Вероятность, что A истинно больше r

Вероятность, что B истинно больше max (0, q+r-1)

Пример:

Вероятность, что A→B больше 0.9
Вероятность, что A истинно больше 0.5

Вероятность, что B истинно больше 0.4

Знания в вычислительном искусственном интеллекте извлекаются из БОЛЬШИХ ДАННЫХ

Закономерности в БОЛЬШИХ ДАННЫХ изучаются НАУКОЙ О ДАННЫХ

Разнообразие

Скорость

Объем

НАУКА О ДАННЫХ

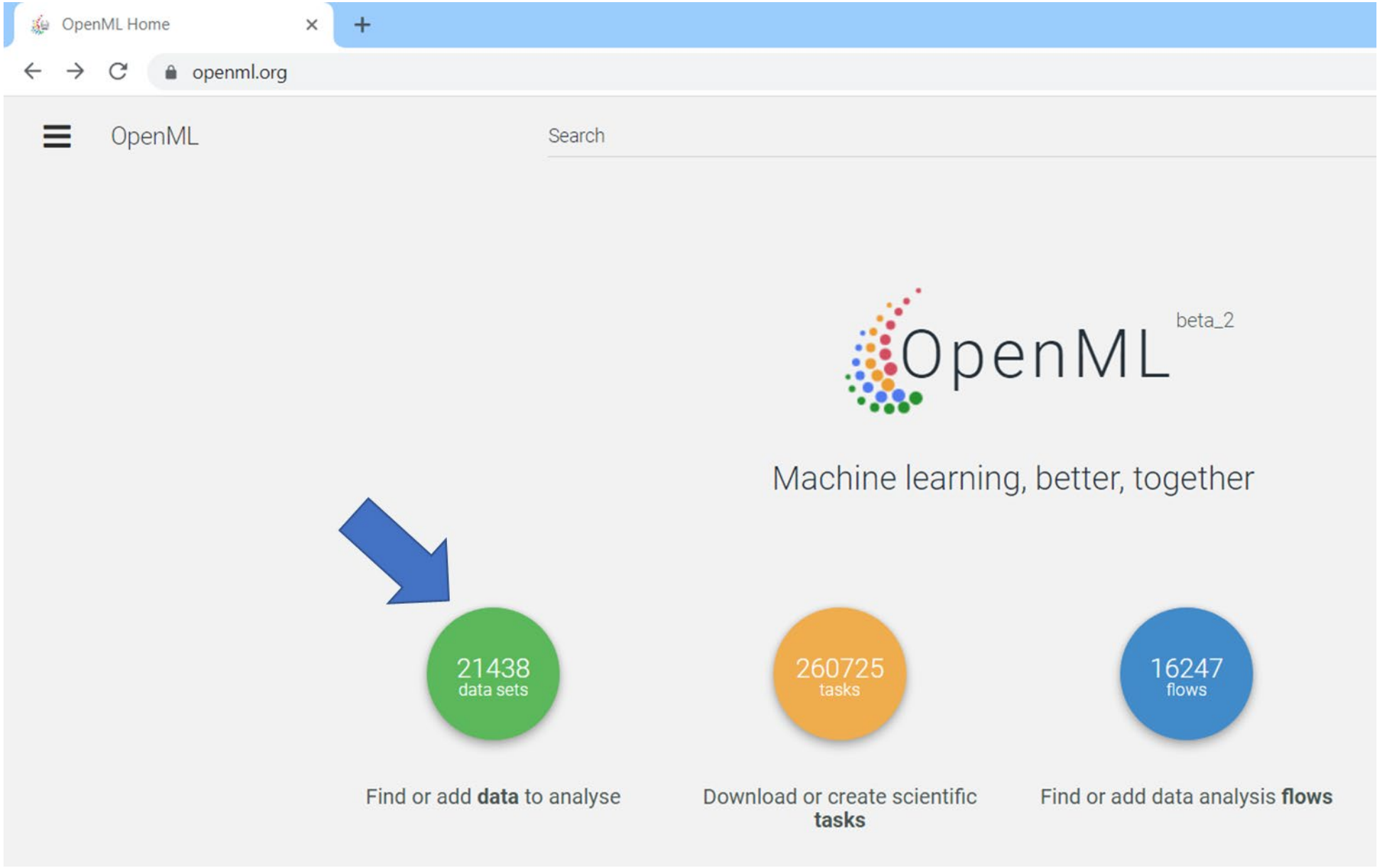
ДОБЫЧА ДАННЫХ
(DATA MINING)

АНАЛИТИКА ДАННЫХ
(DATA ANALYTICS)

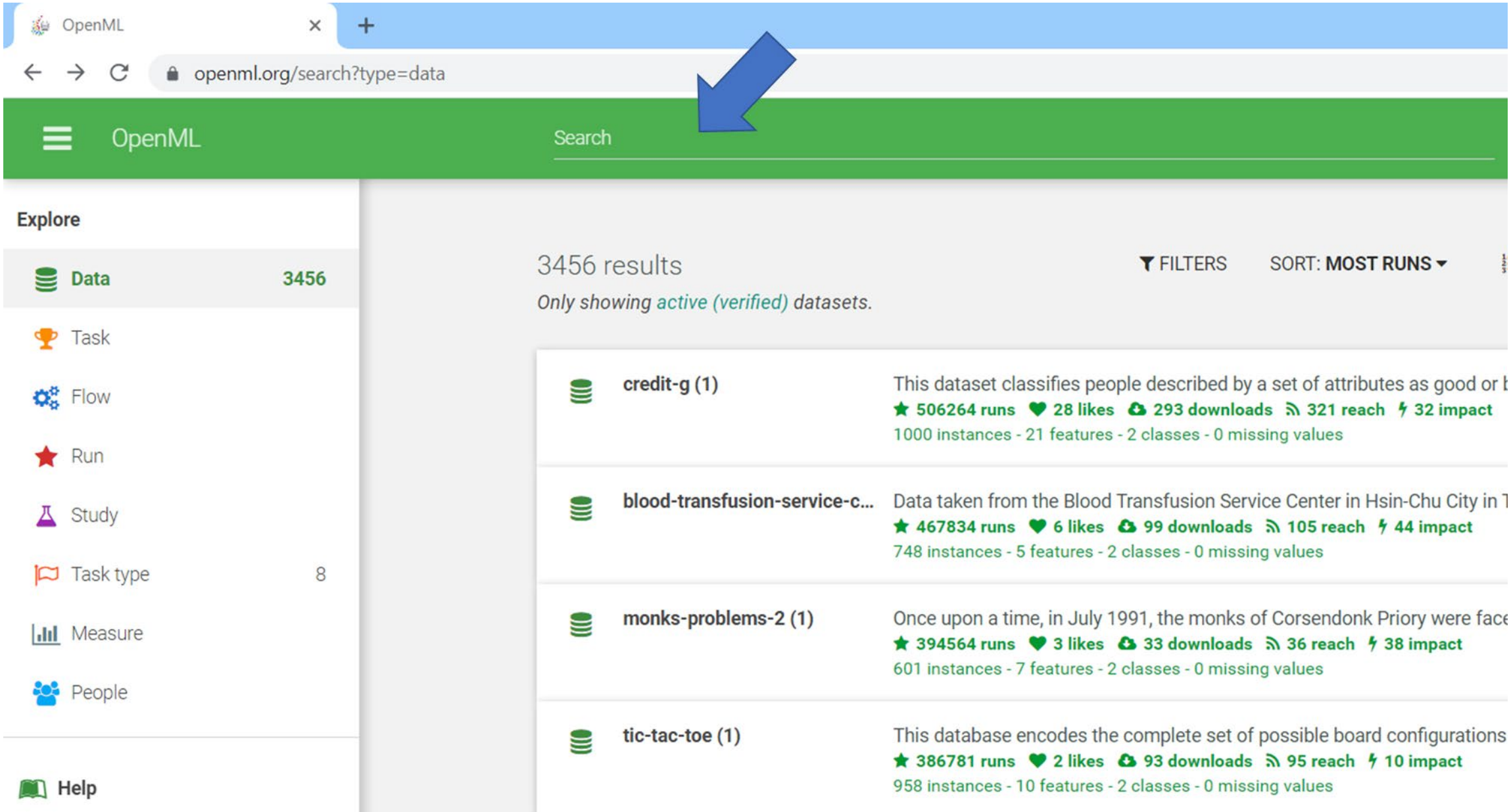
ОТКРЫТЫЕ РЕПОЗИТОРИИ ДАННЫХ ДЛЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Реестр исследовательских репозиториев	*https://www.re3data.org*
Репозиторий Open Machine Learning	*https://www.openml.org*
Репозиторий UCI Machine Learning	*https://archive.ics.uci.edu/ml/index.php*
Репозиторий Scientific Data Repository	*https://www.mlvis.com/*

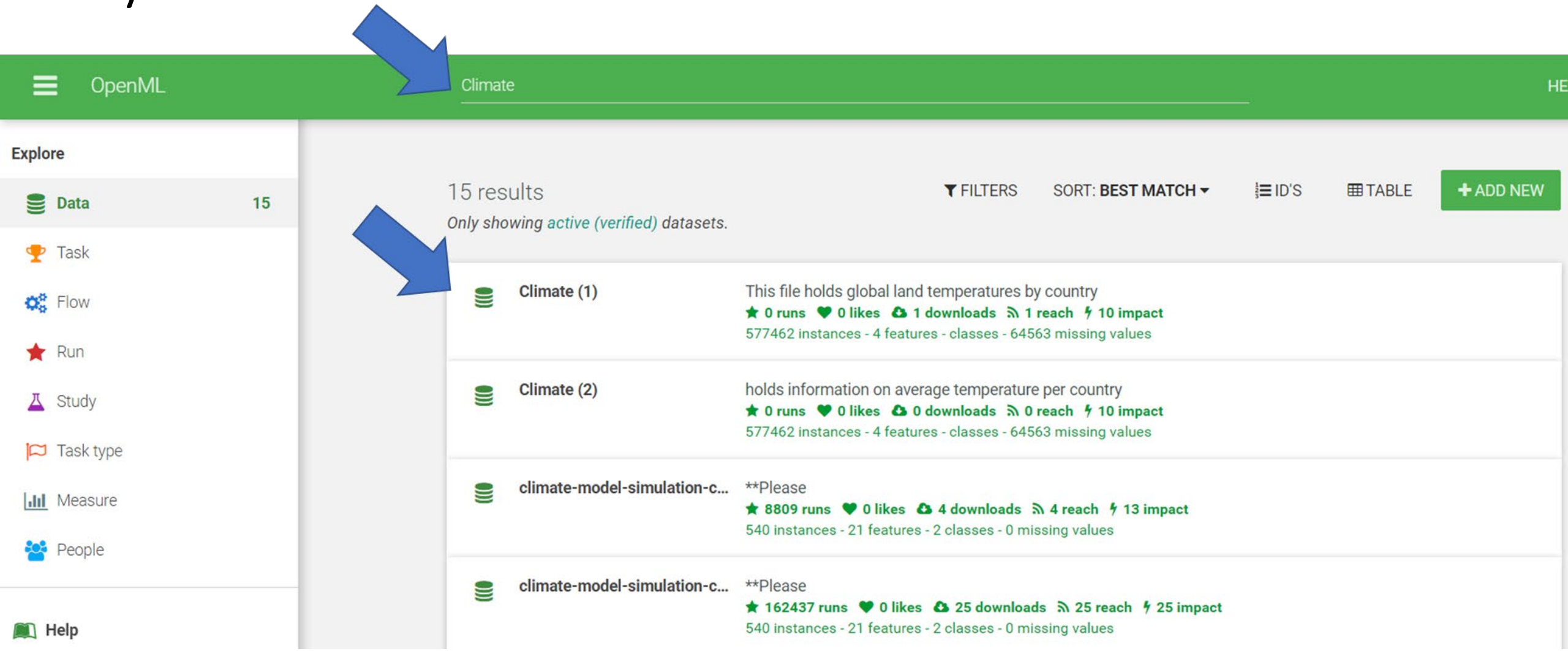
РЕПОЗИТОРИЙ OPENML.ORG



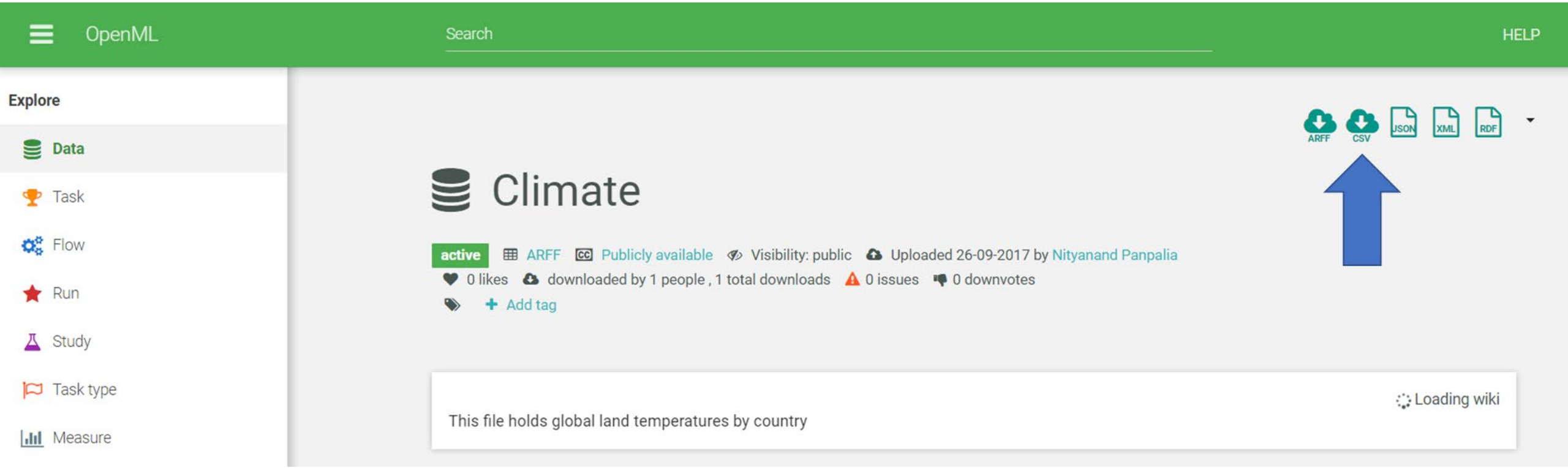
Поиск наборов данных в репозитории



Результат поиска тематики Climate

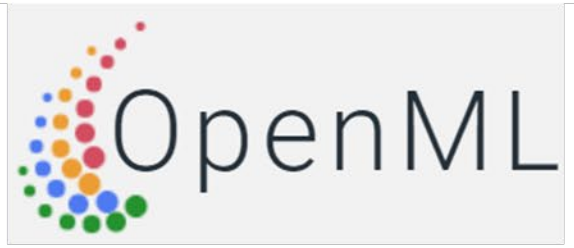


Выбор формата данных Climate



Получение данных

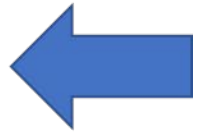
Язык и среда разработки



Программные библиотеки

```
!pip install openml
```

```
import openml as oml
import pandas as pd

dataset = oml.datasets.get_dataset(40918)
X, y, category, attributes = dataset.get_data(target = dataset.default_target_attribute,
                                              dataset_format="dataframe")
```



Получение информации о наборе данных

```
print(X.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 577462 entries, 0 to 577461
Data columns (total 4 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   dt                             577462 non-null  object 
 1   AverageTemperature             544811 non-null  float64
 2   AverageTemperatureUncertainty  545550 non-null  float64
 3   Country                        577462 non-null  object 
dtypes: float64(2), object(2)
memory usage: 17.6+ MB
None
```

Получение информации о наборе данных

```
print(X.head(10))
```

	dt	AverageTemperature	AverageTemperatureUncertainty	Country
0	1743-11-01	4.384	2.294	Åland
1	1743-12-01	NaN	NaN	Åland
2	1744-01-01	NaN	NaN	Åland
3	1744-02-01	NaN	NaN	Åland
4	1744-03-01	NaN	NaN	Åland
5	1744-04-01	1.530	4.680	Åland
6	1744-05-01	6.702	1.789	Åland
7	1744-06-01	11.609	1.577	Åland
8	1744-07-01	15.342	1.410	Åland
9	1744-08-01	NaN	NaN	Åland

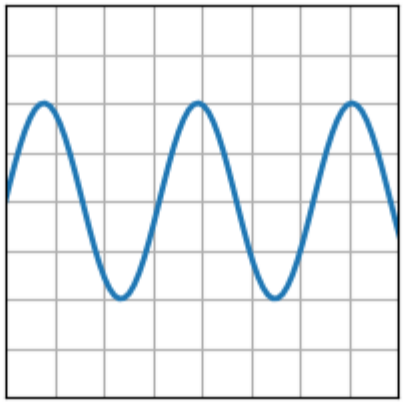
Лекция 4. Визуализация Big Data

Pandas - это пакет Python с открытым исходным кодом, который предоставляет высокоэффективные, простые в использовании структуры данных и инструменты анализа для помеченных данных на языке программирования Python.

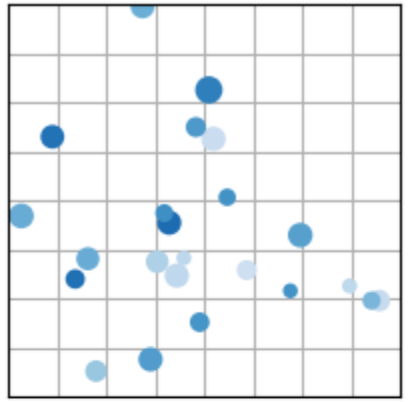
NumPy - один из самых фундаментальных пакетов в Python - универсальный пакет для обработки массивов. Он предоставляет высокопроизводительные объекты многомерных массивов и инструменты для работы с массивами. NumPy - это эффективный контейнер универсальных многомерных данных.

Matplotlib — это обширная библиотека для создания статических, анимированных и интерактивных визуализаций на Python. Matplotlib делает простые вещи простыми, а сложные возможными.
Создание графиков качества публикации.
Создавайте интерактивные фигуры, которые можно масштабировать, панорамировать, обновлять.
Настройка визуального стиля и макета.
Экспорт во многие форматы файлов.
Встраивание в JupyterLab и графические пользовательские интерфейсы.

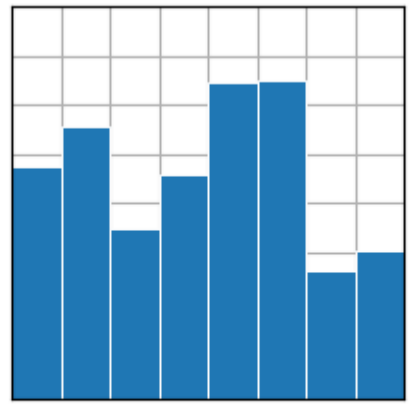
Seaborn — это библиотека визуализации данных Python, основанная на matplotlib. Она предоставляет высокоуровневый интерфейс для рисования привлекательных и информативных статистических графиков.



```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('_mpl-gallery')

# make data
x = np.linspace(0, 10, 100)
y = 4 + 2 * np.sin(2 * x)

# plot
fig, ax = plt.subplots()

ax.plot(x, y, linewidth=2.0)

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```

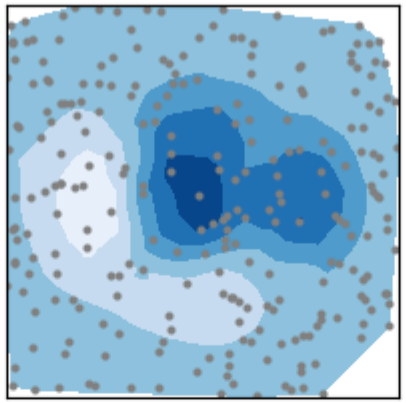


```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('_mpl-gallery')

# make the data
np.random.seed(3)
x = 4 + np.random.normal(0, 2, 24)
y = 4 + np.random.normal(0, 2, len(x))
# size and color:
sizes = np.random.uniform(15, 80, len(x))
colors = np.random.uniform(15, 80, len(x))

# plot
fig, ax = plt.subplots()

ax.scatter(x, y, s=sizes, c=colors, vmin=0, vmax=100)

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```

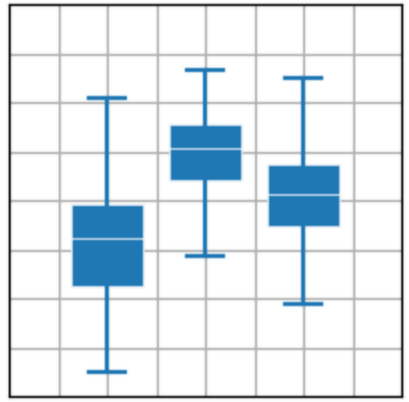


```
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('_mpl-gallery')

# make data:
np.random.seed(3)
x = 0.5 + np.arange(8)
y = np.random.uniform(2, 7, len(x))

# plot
fig, ax = plt.subplots()

ax.bar(x, y, width=1, edgecolor="white", linewidth=0.7)

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```

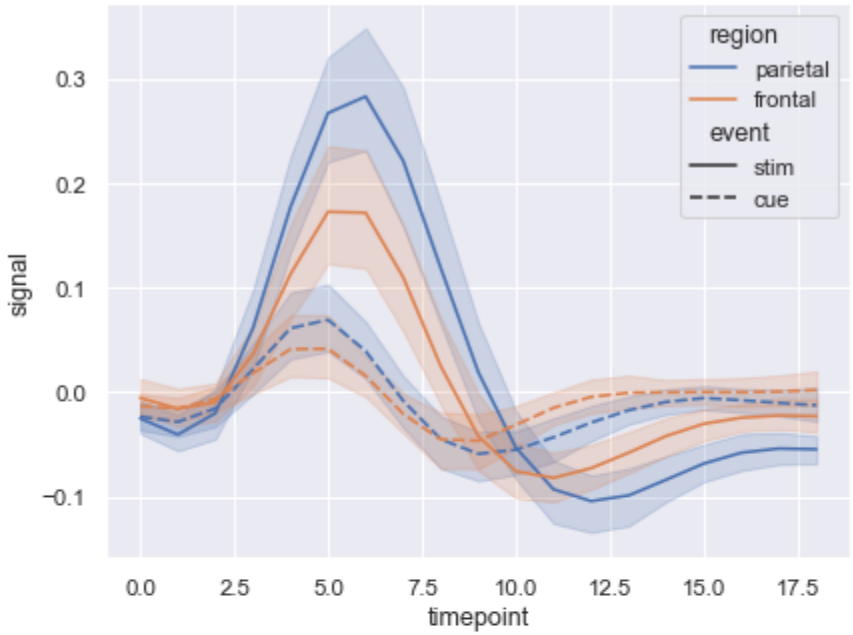
```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('_mpl-gallery-nogrid')

# make a stream function:
X, Y = np.meshgrid(np.linspace(-3, 3, 256), np.linspace(-3, 3, 256))
Z = (1 - X/2 + X**5 + Y**3) * np.exp(-X**2 - Y**2)
# make U and V out of the streamfunction:
V = np.diff(Z[1:, :], axis=1)
U = -np.diff(Z[:, 1:], axis=0)

# plot:
fig, ax = plt.subplots()

ax.streamplot(X[1:, 1:], Y[1:, 1:], U, V)

plt.show()
```



```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('_mpl-gallery-nogrid')

# make data:
np.random.seed(1)
x = np.random.uniform(-3, 3, 256)
y = np.random.uniform(-3, 3, 256)
z = (1 - x/2 + x**5 + y**3) * np.exp(-x**2 - y**2)
levels = np.linspace(z.min(), z.max(), 7)

# plot:
fig, ax = plt.subplots()

ax.plot(x, y, 'o', markersize=2, color='grey')
ax.tricontourf(x, y, z, levels=levels)

ax.set(xlim=(-3, 3), ylim=(-3, 3))

plt.show()
```

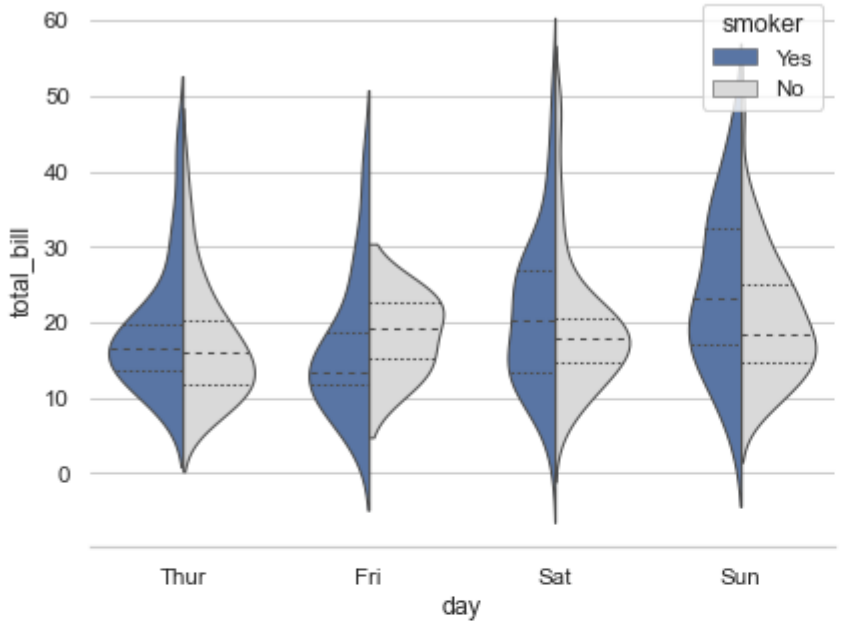


```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('_mpl-gallery')

# make data:
np.random.seed(10)
D = np.random.normal((3, 5, 4), (1.25, 1.00, 1.25), (100, 3))

# plot
fig, ax = plt.subplots()
VP = ax.boxplot(D, positions=[2, 4, 6], widths=1.5, patch_artist=True,
                showmeans=False, showfliers=False,
                medianprops={"color": "white", "linewidth": 0.5},
                boxprops={"facecolor": "C0", "edgecolor": "white",
                          "linewidth": 0.5},
                whiskerprops={"color": "C0", "linewidth": 1.5},
                capprops={"color": "C0", "linewidth": 1.5})

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```



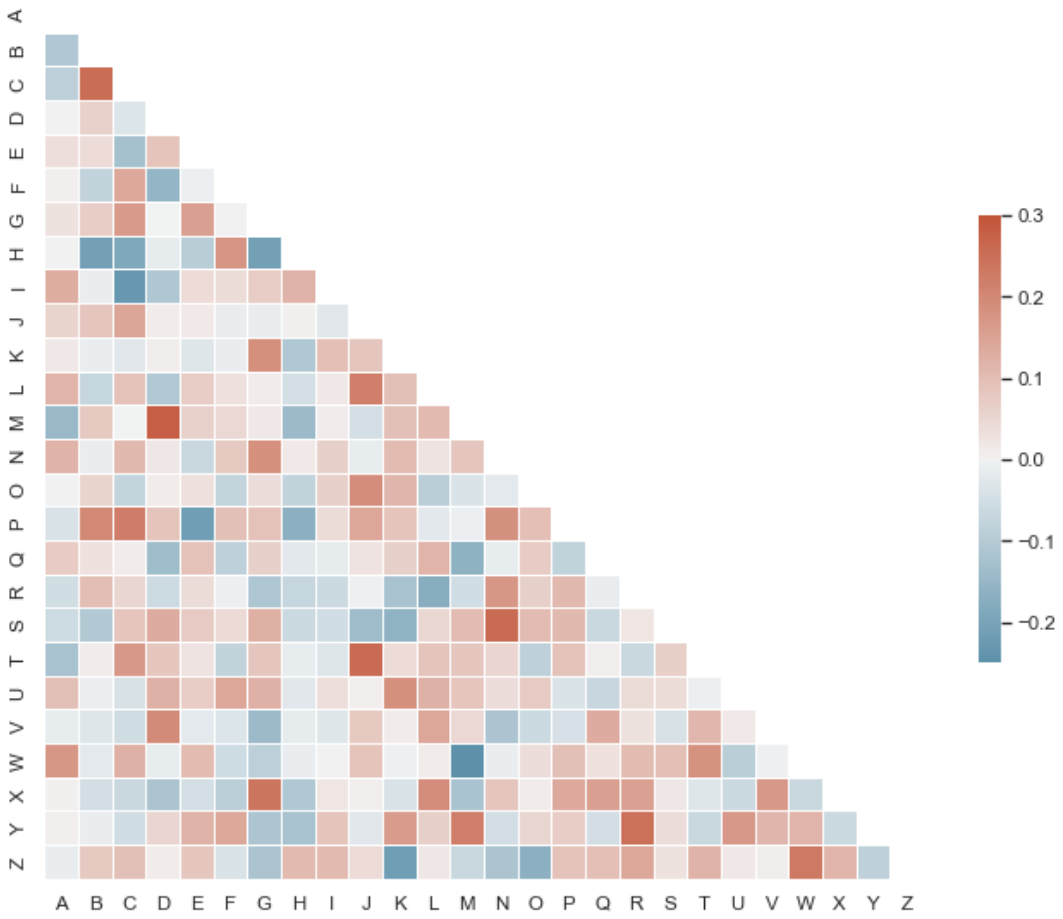
seaborn components used: `set_theme()`, `load_dataset()`, `lineplot()`

```
import seaborn as sns
sns.set_theme(style="darkgrid")

# Load an example dataset with long-form data
fmri = sns.load_dataset("fmri")

# Plot the responses for different events and regions
sns.lineplot(x="timepoint", y="signal",
             hue="region", style="event",
             data=fmri)
```



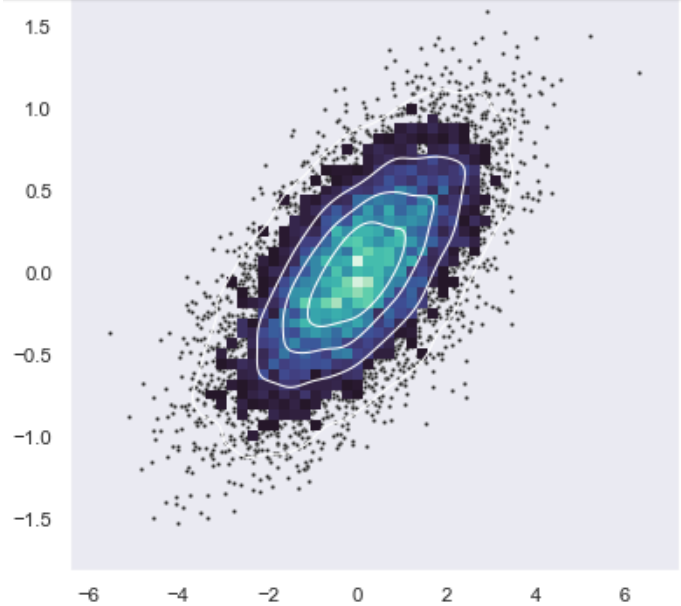
seaborn components used: `set_theme()`, `load_dataset()`, `violinplot()`, `despine()`

```
import seaborn as sns
sns.set_theme(style="whitegrid")

# Load the example tips dataset
tips = sns.load_dataset("tips")

# Draw a nested violinplot and split the violins for easier comparison
sns.violinplot(data=tips, x="day", y="total_bill", hue="smoker",
               split=True, inner="quart", linewidth=1,
               palette={"Yes": "b", "No": ".85"})
sns.despine(left=True)
```



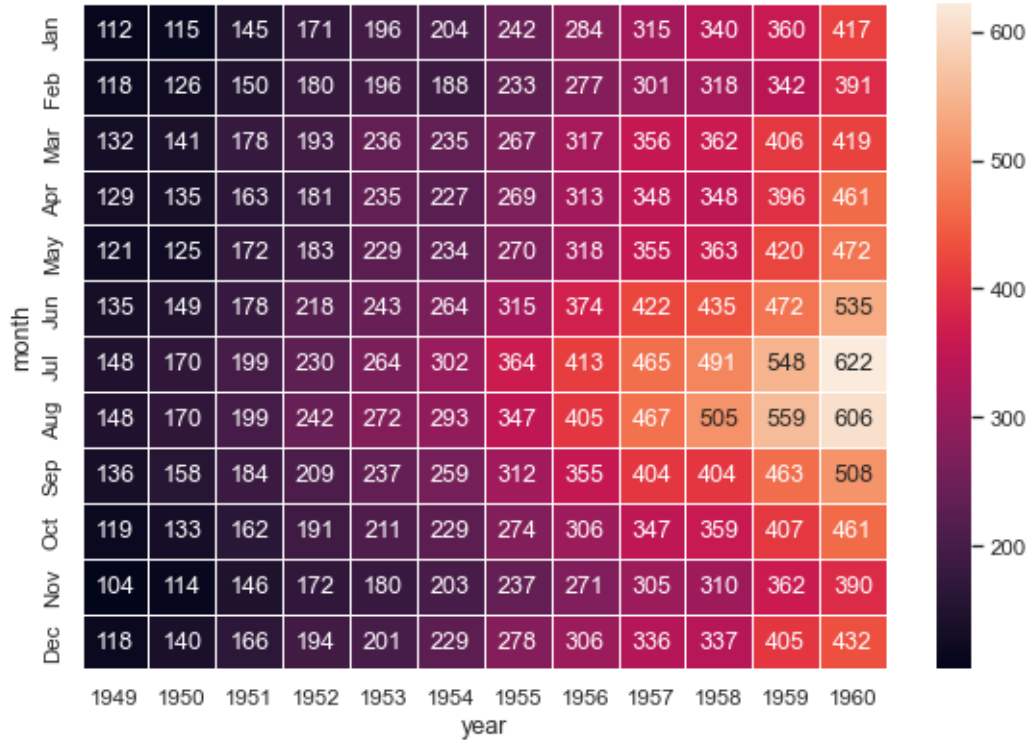
seaborn components used: `set_theme()`, `diverging_palette()`, `heatmap()`

```
from string import ascii_letters
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_theme(style="white")

# Generate a large random dataset
rs = np.random.RandomState(33)
d = pd.DataFrame(data=rs.normal(size=(100, 26)),
                 columns=list(ascii_letters[26:]))

# Compute the correlation matrix
corr = d.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```



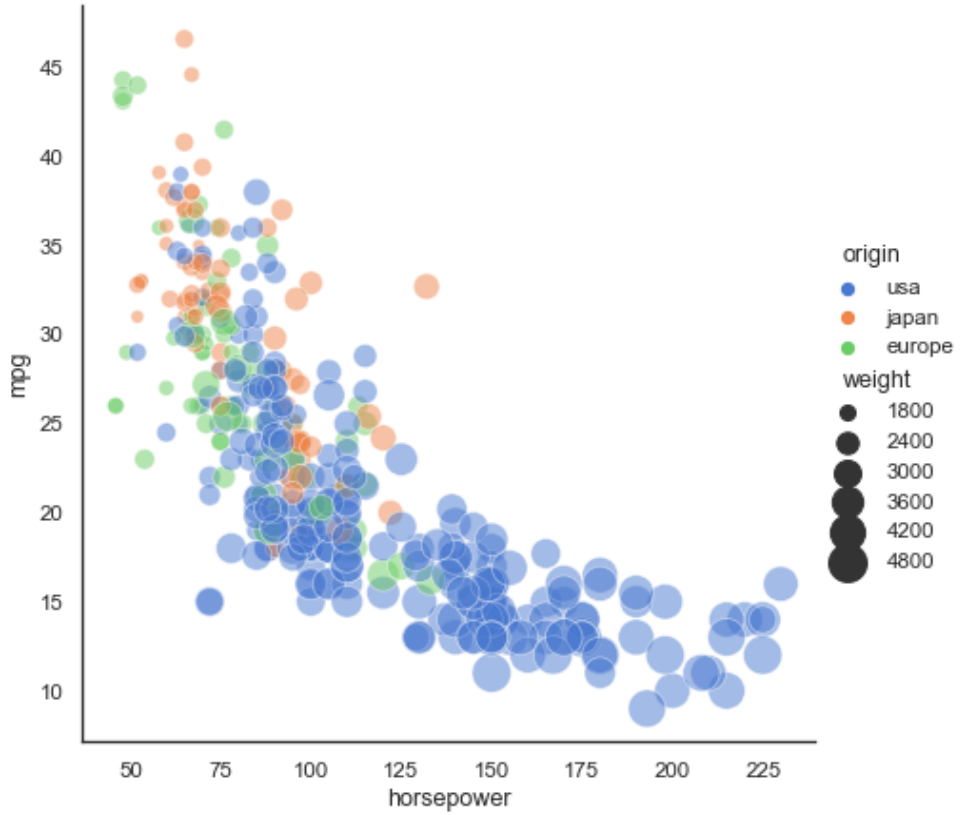
seaborn components used: `set_theme()`, `scatterplot()`, `histplot()`, `kdeplot()`

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="dark")

# Simulate data from a bivariate Gaussian
n = 10000
mean = [0, 0]
cov = [(2, .4), (.4, .2)]
rng = np.random.RandomState(0)
x, y = rng.multivariate_normal(mean, cov, n).T

# Draw a combo histogram and scatterplot with density contours
f, ax = plt.subplots(figsize=(6, 6))
sns.scatterplot(x=x, y=y, s=5, color=".15")
sns.histplot(x=x, y=y, bins=50, pthresh=.1, cmap="mako")
sns.kdeplot(x=x, y=y, levels=5, color="w", linewidths=1)
```



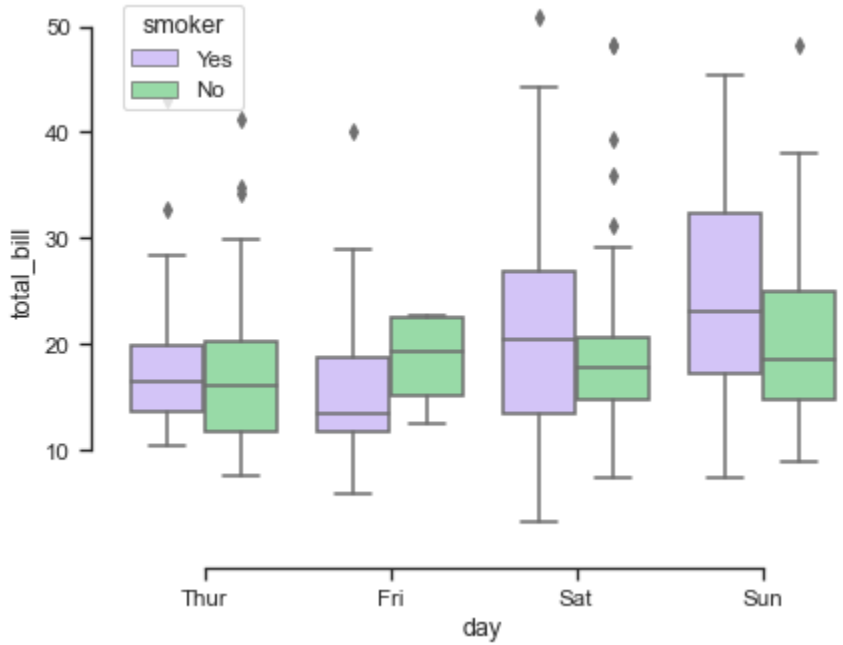
seaborn components used: `set_theme()`, `load_dataset()`, `heatmap()`

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()

# Load the example flights dataset and convert to long-form
flights_long = sns.load_dataset("flights")
flights = flights_long.pivot("month", "year", "passengers")

# Draw a heatmap with the numeric values in each cell
f, ax = plt.subplots(figsize=(9, 6))
sns.heatmap(flights, annot=True, fmt="d", linewidths=.5, ax=ax)
```



seaborn components used: `set_theme()`, `load_dataset()`, `relplot()`

```
import seaborn as sns
sns.set_theme(style="white")

# Load the example mpg dataset
mpg = sns.load_dataset("mpg")

# Plot miles per gallon against horsepower with other semantics
sns.relplot(x="horsepower", y="mpg", hue="origin", size="weight",
            sizes=(40, 400), alpha=.5, palette="muted",
            height=6, data=mpg)
```



seaborn components used: `set_theme()` , `load_dataset()` , `boxplot()` , `despine()`

```
import seaborn as sns
sns.set_theme(style="ticks", palette="pastel")

# Load the example tips dataset
tips = sns.load_dataset("tips")

# Draw a nested boxplot to show bills by day and time
sns.boxplot(x="day", y="total_bill",
            hue="smoker", palette=["m", "g"],
            data=tips)
sns.despine(offset=10, trim=True)
```

Лекция 5. Анализ и преобразование больших данных, библиотеки Python

Почему необходим предварительный анализ данных?

Наборы данных могут иметь:

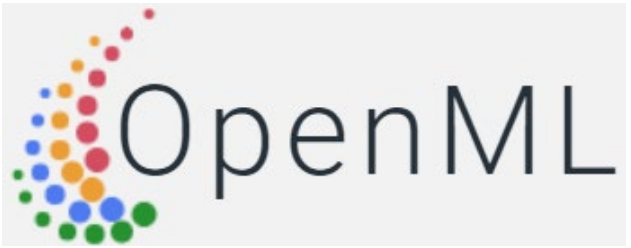
Пропущенные значения – NaN

Экстремальные значения – Выбросы

Полностью совпадающие – избыточные значения

Несогласованные значения

Этапы предварительного анализа данных



Этапы предварительного анализа данных

```
import openml as oml
import pandas as pd
import numpy as np
import seaborn as sbn

dataset = oml.datasets.get_dataset(40918)
X, y, category, attributes = dataset.get_data(target = dataset.default_target_attribute,
                                              dataset_format="dataframe")
```

Переименование набора данных и его столбцов

```
df = X.rename(columns={'dt':'Date','AverageTemperature':'Temperature',
                       'AverageTemperatureUncertainty':'Scatter'}, )
```

Вывод DataFrame на экран

```
df
```

:

	Date	Temperature	Scatter	Country
0	1743-11-01	4.384	2.294	Åland
1	1743-12-01	NaN	NaN	Åland
2	1744-01-01	NaN	NaN	Åland
3	1744-02-01	NaN	NaN	Åland
4	1744-03-01	NaN	NaN	Åland
...	...	...	...	...
577457	2013-05-01	19.059	1.022	Zimbabwe
577458	2013-06-01	17.613	0.473	Zimbabwe
577459	2013-07-01	17.000	0.453	Zimbabwe
577460	2013-08-01	19.759	0.717	Zimbabwe
577461	2013-09-01	NaN	NaN	Zimbabwe

577462 rows × 4 columns

NaN – пропущенные значения, в CSV они обозначены знаком ?

```
"dt","AverageTemperature","AverageTemperatureUncertainty","Country"
1743-11-01,4.384,2.294,Åland
1743-12-01,?,?,Åland
1744-01-01,?,?,Åland
1744-02-01,?,?,Åland
1744-03-01,?,?,Åland
1744-04-01,1.53,4.68,Åland
1744-05-01,6.702000000000001,1.7890000000000001,Åland
1744-06-01,11.609000000000002,1.577,Åland
1744-07-01,15.342,1.41,Åland
1744-08-01,?,?,Åland
1744-09-01,11.702,1.517,Åland
1744-10-01,5.477,1.8619999999999999,Åland
1744-11-01,3.407,1.425,Åland
1744-12-01,-2.181,1.641,Åland
1745-01-01,-3.85,1.841,Åland
1745-02-01,-6.5749999999999975,1.36,Åland
1745-03-01,-4.195,1.213,Åland
1745-04-01,-0.9660000000000002,1.172,Åland
```

Числовые и категориальные типы данных

```
print(df.dtypes)
```

```
Date            object
Temperature    float64
Scatter        float64
Country         object
dtype: object
```

Вывод типов данных командой dtypes

```
df_numeric = df.select_dtypes(include=[np.number])
numeric_cols = df_numeric.columns.values
print(numeric_cols)
```

```
['Temperature' 'Scatter']
```

Вывод названий столбцов с числовыми данными

```
df_non_numeric = df.select_dtypes(exclude=[np.number])
non_numeric_cols = df_non_numeric.columns.values
print(non_numeric_cols)
```

```
['Date' 'Country']
```

Вывод названий столбцов с категориальными данными

```
cols = df.columns
sbn.heatmap(df[cols].isna(), cmap='Greens')
```

Вывод тепловой карты таблицы
Команда isna определяет NaN значения

Пропущенные значения имеют более темный цвет

```
df.shape
```

```
(577462, 4)
```

Число строк в исходном наборе, вместе с NaN значениями

```
df.isna().sum()
```

```
Date              0
Temperature    32651
Scatter        31912
Country            0
dtype: int64
```

Число пропущенных NaN значений

```
for col in cols:
    data_missing = np.mean(df[col].isna())
    print('{} - {}%'.format(col, round(data_missing*100)))
```

```
Date - 0%
Temperature - 6%
Scatter - 6%
Country - 0%
```

Процентное соотношение пропущенных NaN значений в наборе

```
df = df.dropna()
df.shape
```

```
(544811, 4)
```

```
for col in cols:
    data_missing = np.mean(df[col].isna())
    print('{} - {}%'.format(col, round(data_missing*100)))
```

```
Date - 0%
Temperature - 0%
Scatter - 0%
Country - 0%
```

```
df
```

	Date	Temperature	Scatter	Country
0	1743-11-01	4.384	2.294	Åland
5	1744-04-01	1.530	4.680	Åland
6	1744-05-01	6.702	1.789	Åland
7	1744-06-01	11.609	1.577	Åland
8	1744-07-01	15.342	1.410	Åland
...	...	...	...	...
577456	2013-04-01	21.142	0.495	Zimbabwe
577457	2013-05-01	19.059	1.022	Zimbabwe
577458	2013-06-01	17.613	0.473	Zimbabwe
577459	2013-07-01	17.000	0.453	Zimbabwe
577460	2013-08-01	19.759	0.717	Zimbabwe

544811 rows × 4 columns

Вывод набора после выполнения команды dropna – удаления пропущенных значений

```
df['Temperature'].describe()
```

```
count    544811.000000
mean         17.193354
std          10.953966
min         -37.658000
25%          10.025000
50%          20.901000
75%          25.814000
max          38.842000
Name: Temperature, dtype: float64
```

Число строк

Средняя температура

Стандартное отклонение

Минимальная температура

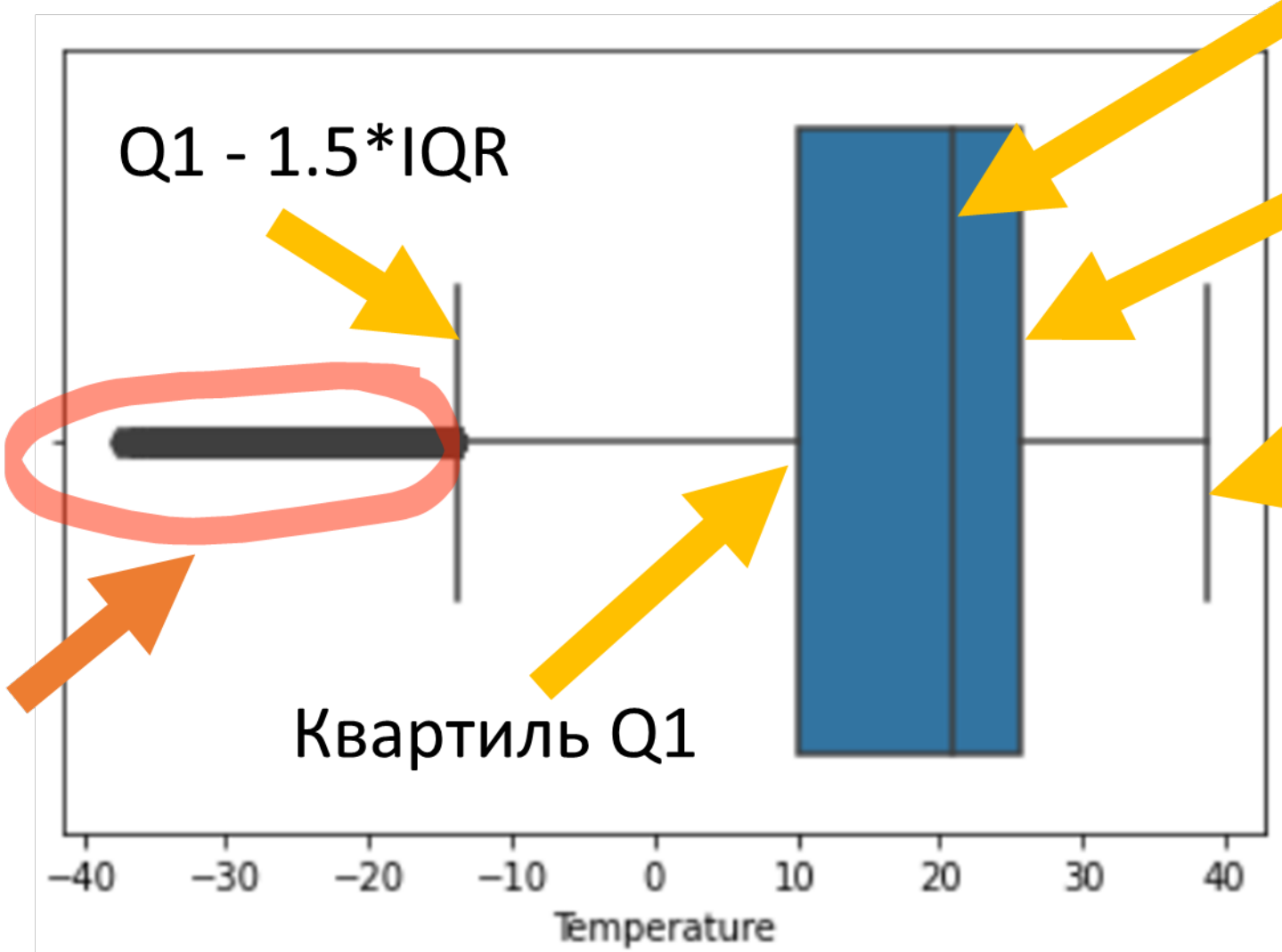
0.25 квантиль (Q1)

0.5 квантиль (медиана, Q2)

0.75 квантиль (Q4)

Максимальная температура

```
sbn.boxplot(df['Temperature'])
```



Медиана, Q2

Квартиль Q3

Q3 + 1.5*IQR

IQR = Q3 – Q1

Область выбросов

Расчет квартилей Q1 и Q3

```
Q1 = df["Temperature"].quantile(0.25)
Q3 = df["Temperature"].quantile(0.75)
print('Q1 = {:0.2f} градусов Цельсия'.format(Q1))
print('Q3 = {:0.2f} градусов Цельсия'.format(Q3))
```

```
Q1 = 10.03 градусов Цельсия
Q3 = 25.81 градусов Цельсия
```

Расчет межквантильного расстояния

```
IQR = Q3 - Q1
print('IQR = {:0.2f} градусов Цельсия'.format(IQR))
```

```
IQR = 15.79 градусов Цельсия
```

Расчет нижней и верхней границ отсечения выбросов

```
Low = Q1 - 1.5*IQR
Upp = Q3 + 1.5*IQR
print('Low = {:0.2f} градусов Цельсия'.format(Low))
print('Upp = {:0.2f} градусов Цельсия'.format(Upp))
```

```
Low = -13.66 градусов Цельсия
Upp = 49.50 градусов Цельсия
```

Условие отсечения выбросов: < Low или > Upp

```
df_out = df[(df["Temperature"] < Low) |(df["Temperature"] > Upp)]
```

```
df_out.shape
```

```
: (6438, 4)
```

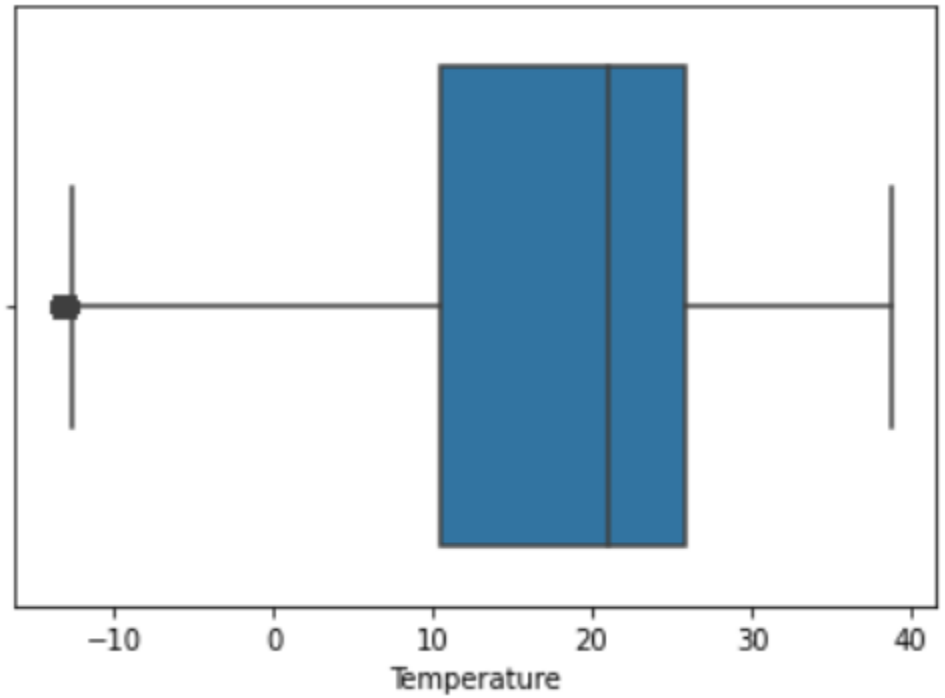
Сохранение данных без отсеченных выбросов

```
df_clean = df[~((df["Temperature"] < Low) |(df["Temperature"] > Upp))]
```

```
df_clean.shape
```

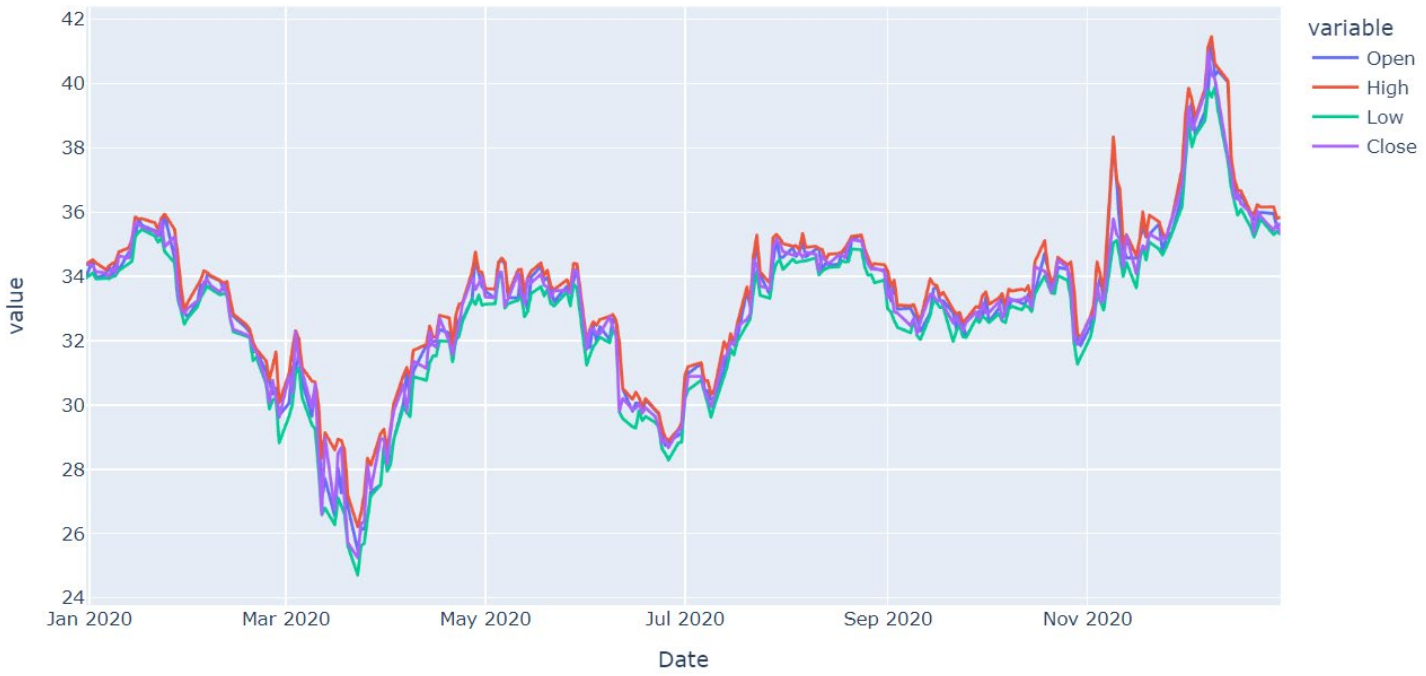
```
: (538373, 4)
```

```
sbn.boxplot(df_clean['Temperature'])
```



Лекция 6. Работа с временными рядами в библиотеках Python

Почему временные ряды являются особенными данными?



Временной ряд – это упорядоченная последовательность значений какого-либо параметра исследуемого процесса, зафиксированная за некоторый период времени

Наблюдение за изменением данных во времени позволяет:

анализировать ход изменения процесса и объяснять его закономерности

прогнозировать будущие значения и предсказывать поведение процесса

Виды временных рядов

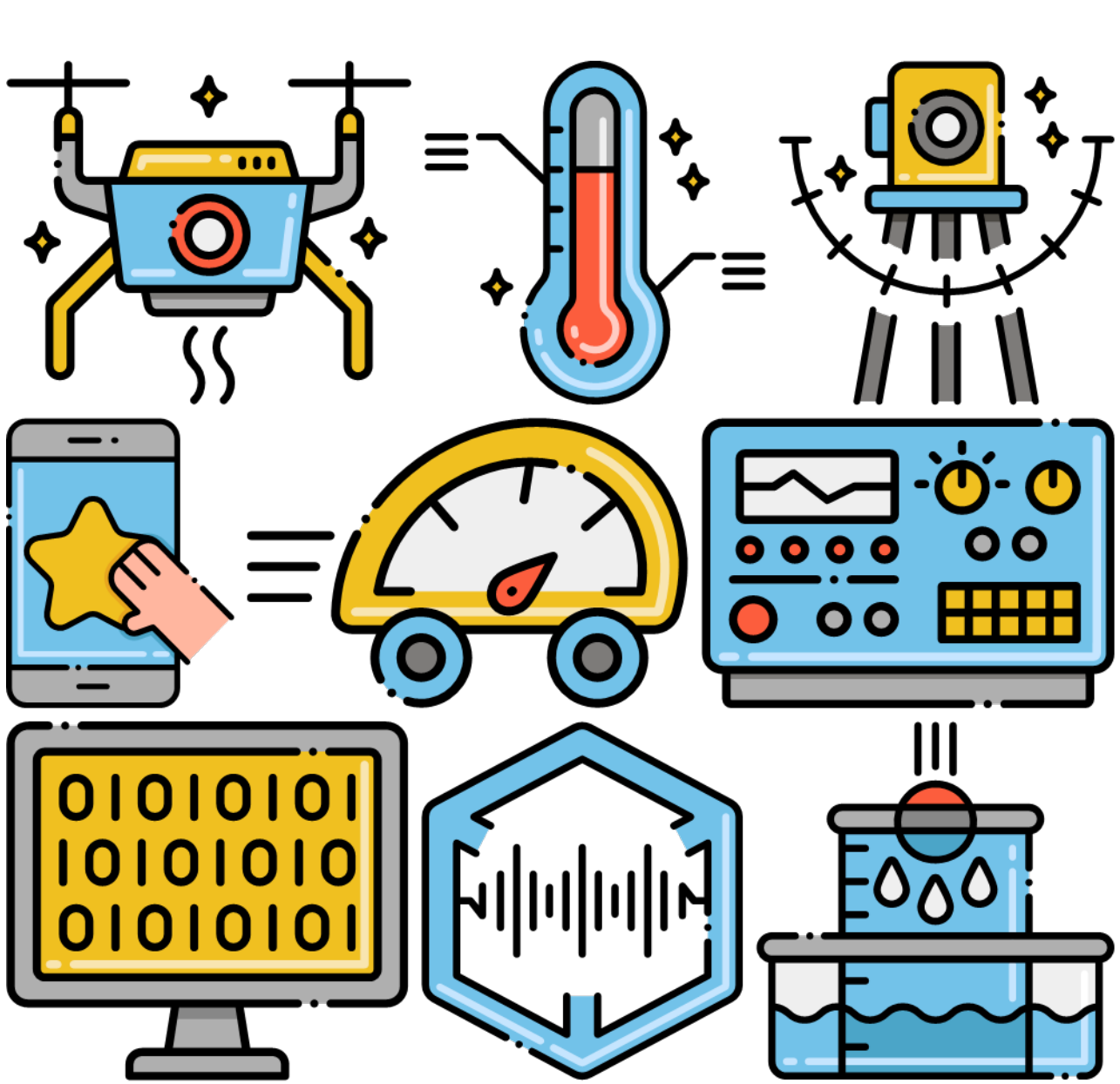
Одномерные
и
Многомерные

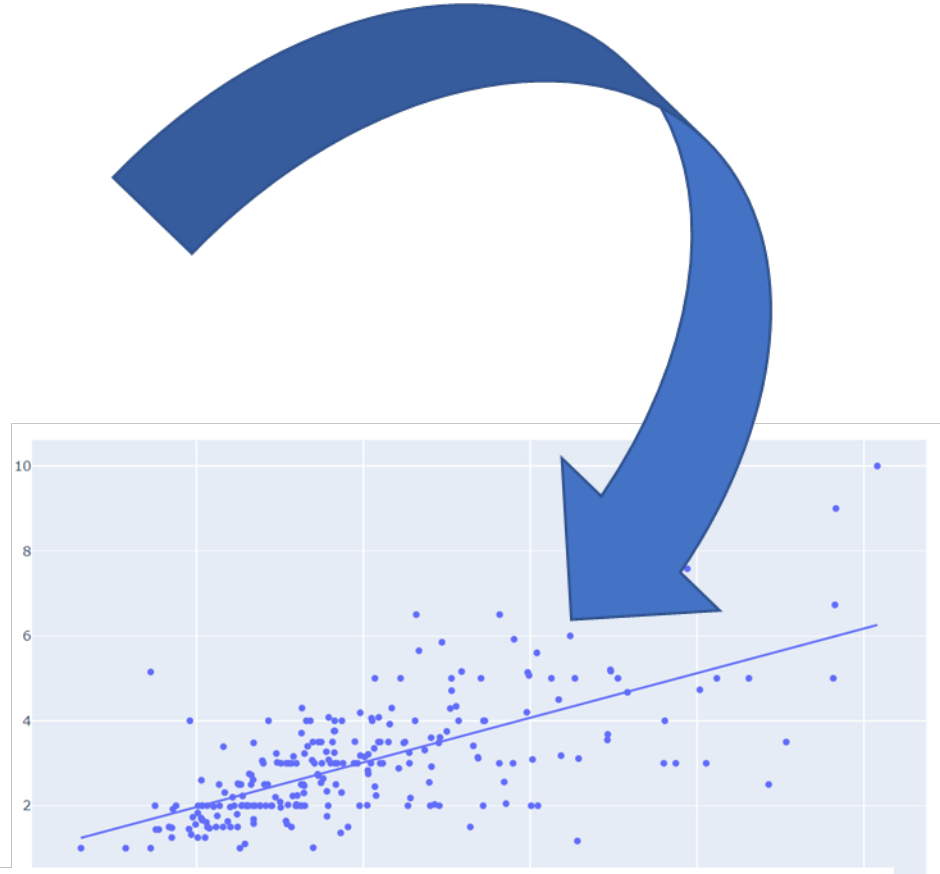
Полные
и
Неполные

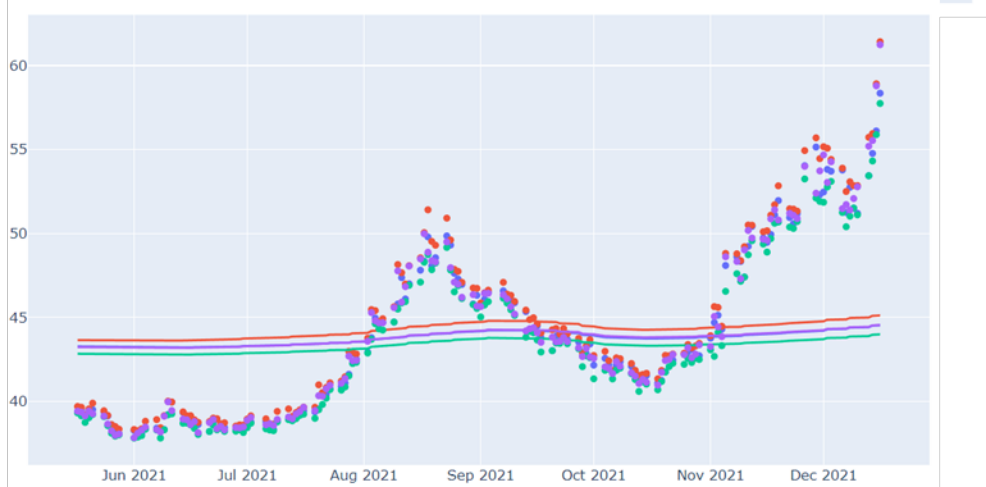
Моментные
и
Интервальные

Детерминированные
и
Случайные

Стационарные
и
Нестационарные


0101010101
1010101010
0101010101


10
8
6
4
2


60
55
50
45
40
Jun 2021
Jul 2021
Aug 2021
Sep 2021
Oct 2021
Nov 2021
Dec 2021

Моделирование временных рядов бывает:

- эвристическим
- физическим
- дискретно-событийным

В общем случае временной ряд может содержать следующие компоненты:

- тренд
- циклы
- периодические колебания
- шумы

Аддитивная модель –
это сумма перечисленных компонентов
Мультипликативная модель –
это произведение перечисленных компонентов

Компьютерная модель стационарного временного ряда с нулевым средним

Это простейшая модель временного ряда, целью которой является синтез последовательности независимо и одинаково распределенных случайных величин $\{X_t\} \sim \mathrm{IID}(0, \sigma^2)$, имеющих 0 среднее и дисперсию σ^2

```
import numpy as np
import plotly.express as px
zm = np.random.normal(loc=0.0, scale=1., size=1000)
zm[:20]
```

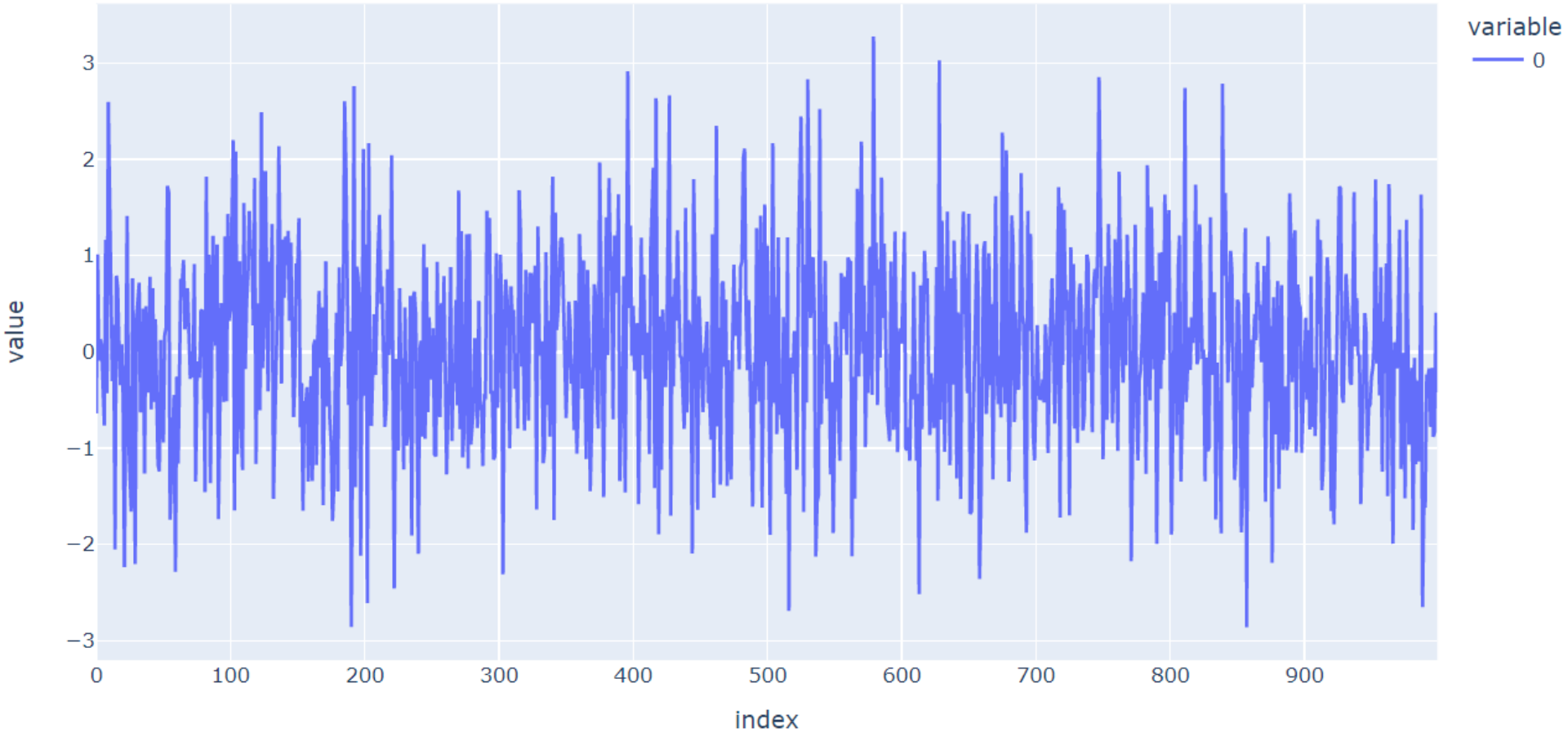
```
array([-1.15895047, -1.24428974,  1.47937251,  0.40382238, -0.2814799 ,
       -0.57725072, -1.88338459, -0.86949924, -0.90505808,  0.0099084 ,
        0.33639664, -0.46517187, -0.38612668, -0.39067157, -0.00452809,
        1.87990887,  0.66758648, -1.150283  ,  1.77284496, -0.55723942])
```

NumPy

plotly

Компьютерная модель стационарного временного ряда с нулевым средним

```
fig = px.line(zm)
fig.show()
```

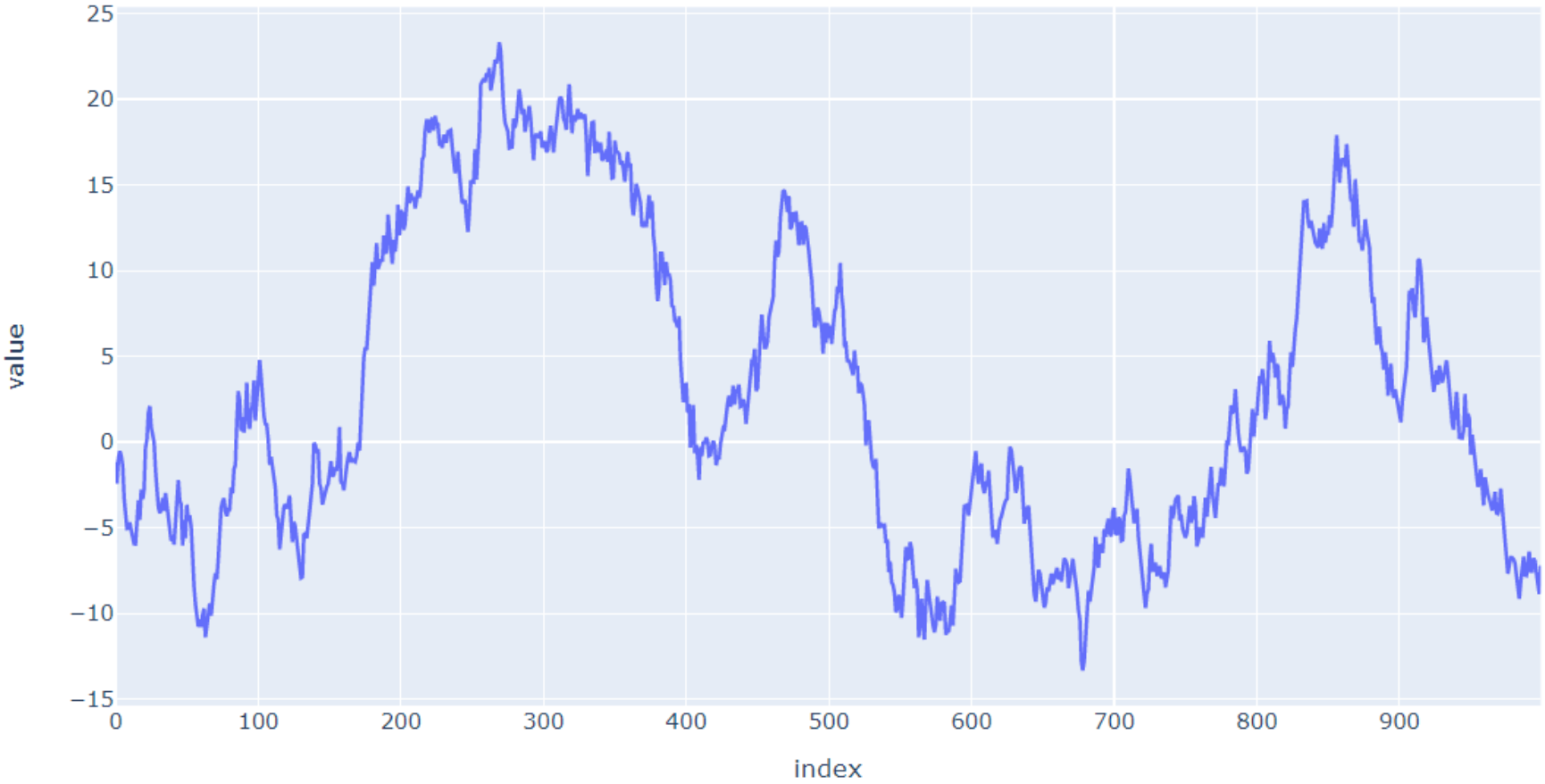


$$\{S_t, t = 0, 1, 2, \ldots\}$$

$$S_0 = 0 \quad S_t = X_1 + \ldots + X_t$$

$$\{X_t\} \sim \mathrm{IID}(0, \sigma^2)$$

Компьютерная модель временного ряда - случайное блуждание




```
rw = np.cumsum(zm)
fig = px.line(rw)
fig.show()
```

finance.yahoo.com/quote/AAPL?p=AAPL

yahoo! finance

Search for news, symbols or companies

S&P 500	Dow 30	Nasdaq	Russell 2000	Crude Oil
4,725.79	35,950.56	15,653.37	2,241.58	73.76
+29.23 (+0.62%)	+196.66 (+0.55%)	+131.47 (+0.85%)	+19.68 (+0.89%)	-0.03 (-0.04%)

Apple Inc. (AAPL)

NasdaqGS - NasdaqGS Real Time Price. Currency in USD

Add to watchlist

176.28 +0.64 (+0.36%)
At close: December 23 04:00PM EST

176.34 +0.06 (+0.03%)
After hours: Dec 23, 07:59PM EST

Summary | Chart | Conversations | Statistics | Historical Data | Profile | Financials | Analysis | Options | Holders | Sustainability

Previous Close	175.64	Market Cap	2.892T
Open	175.85	Beta (5Y Monthly)	1.20
Bid	176.27 x 800	PE Ratio (TTM)	31.42
Ask	176.40 x 1000	EPS (TTM)	5.61
Day's Range	175.28 - 176.85	Earnings Date	Jan 25, 2022 - Jan 31, 2022
52 Week Range	116.21 - 182.13	Forward Dividend & Yield	0.88 (0.50%)
Volume	68,356,867	Ex-Dividend Date	Nov 05, 2021
Avg. Volume	90,578,800	1y Target Est	174.93

Получение временных рядов финансовых данных


```
import yfinance as yf
import plotly.express as px
```

Импорт библиотек

```
apple = yf.Ticker('AAPL')
```

Считывание тикера

не связан с Yahoo.Finance и является открытой библиотекой

```
ts  =  apple.history(start='2021-01-01',
                     end ='2021-12-01')
```

Считывание исторических данных за период времени

Набор из нескольких временных рядов: Open, High, Low, Close

```
ts
```

```
:
```

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2020-12-31	133.258497	133.914457	130.912956	131.877014	99116600	0.0	0
2021-01-04	132.701914	132.791359	125.983331	128.617096	143301900	0.0	0
2021-01-05	128.100300	130.932844	127.643112	130.207306	97664900	0.0	0
2021-01-06	126.937458	130.247057	125.605665	125.824318	155088000	0.0	0
2021-01-07	127.573531	130.823500	127.076595	130.117844	109578200	0.0	0
...	...	...	...	...	...	...	...
2021-11-23	161.119995	161.800003	159.059998	161.410004	96041900	0.0	0
2021-11-24	160.750000	162.139999	159.639999	161.940002	69463600	0.0	0
2021-11-26	159.570007	160.449997	156.360001	156.809998	76959800	0.0	0
2021-11-29	159.369995	161.190002	158.789993	160.240005	88748200	0.0	0
2021-11-30	159.990005	165.520004	159.919998	165.300003	174048100	0.0	0

231 rows × 7 columns

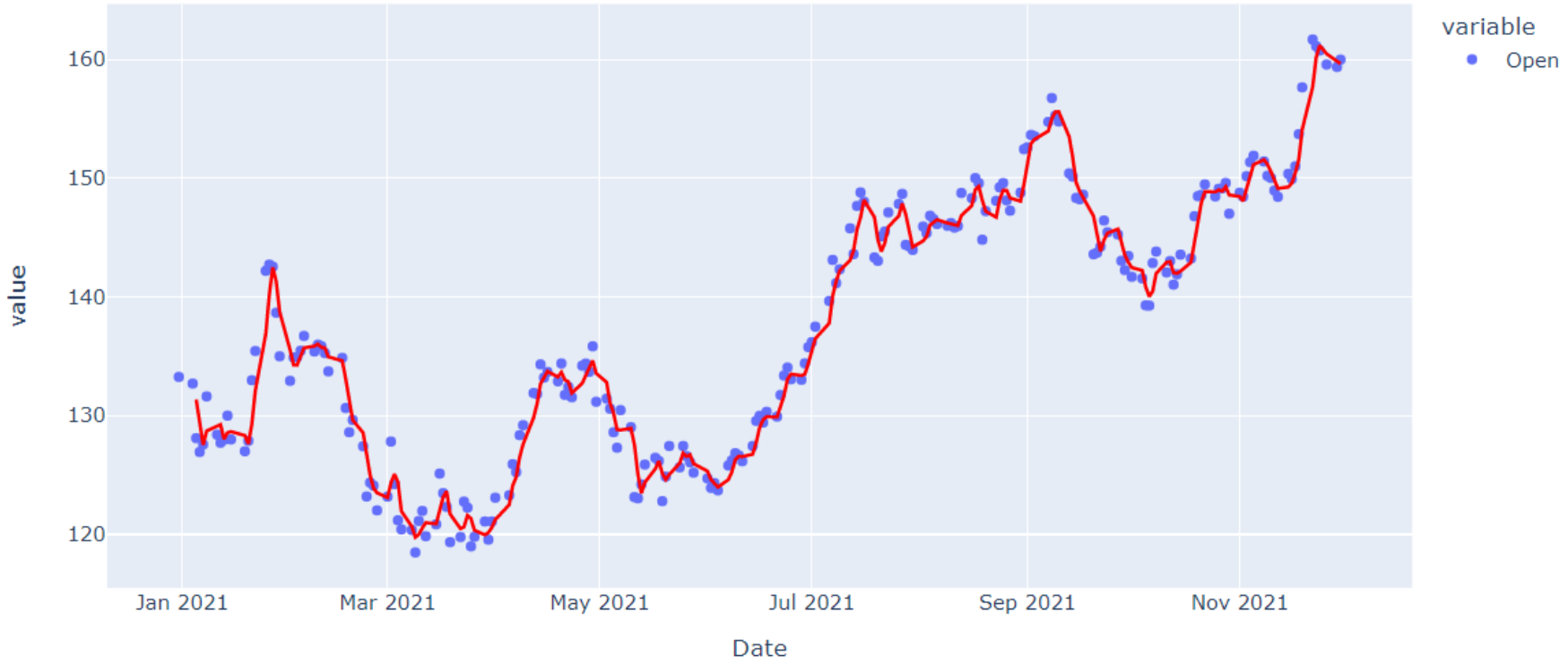
Визуализация временных рядов: Open, High, Low, Close

```
fig = px.line(ts, y=['Open','High','Low','Close'])
fig.show()
```

Отображение тренда для временного ряда Open

```
fig = px.scatter(ts, y=['Open'], trendline="rolling",
                 trendline_options=dict(window=3),
                 title="Отображение линии тренда",
                 trendline_color_override="red")
fig.show()
```

Отображение линии тренда



statsmodels

библиотека рассчитывает тренд разными способами

РАЗДЕЛ 2
ХРАНИЛИЩА ДАННЫХ И ТЕХНОЛОГИИ РАБОТЫ С НИМИ

Лекция 7. Системы управления базами данных

База данных (БД) представляет собой совокупность структурированных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области.

Система управления базами данных (СУБД) – это набор программ, которые управляют структурой БД и контролируют доступ к данным, хранящимся в БД.

Системы управления базами данных (СУБД) позволяют размещать в своих структурах данные и методы, с помощью которых происходит взаимодействие с потребителем или с другими программно-аппаратными комплексами.

- **Система баз данных** понимается фактически как синоним понятия информационная система и включает в себя данные, аппаратное обеспечение, программное обеспечение и пользователей.
- Цель: хранить данные и предоставлять информацию.
- В узком смысле система баз данных понимается как СУБД с управляемой ею базой данных, возможно, уже наполненной.

MySQL

MySQL – бесплатная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет компания Oracle. MySQL широкое распространение получила в интернете, как система хранения данных у сайтов, иными словами, подавляющее большинство сайтов хранят свои данные в базе MySQL.

Microsoft SQL Server

Microsoft SQL Server – это система управления реляционными базами данных, разработанная компанией Microsoft. Ее активно используют в корпоративном секторе, особенно в крупных компаниях. И это не просто СУБД – это целый комплекс приложений, позволяющий не только хранить и модифицировать данные, но еще и анализировать их, осуществлять безопасность этих данных и многое другое.

Oracle

Oracle Database – это система управления базами данных от компании Oracle. Эта СУБД также активно используется крупными компаниями и стоит немаленьких денег, но взамен она предоставляет огромный функционал и надёжность. Поэтому Oracle Database и Microsoft SQL Server являются серьезными конкурентами друг другу.

PostgreSQL

PostgreSQL – это бесплатная реляционная система управления базами данных. Ее активно используют на UNIX-подобных платформах, для реализации как малых и средних, так и крупных проектов.

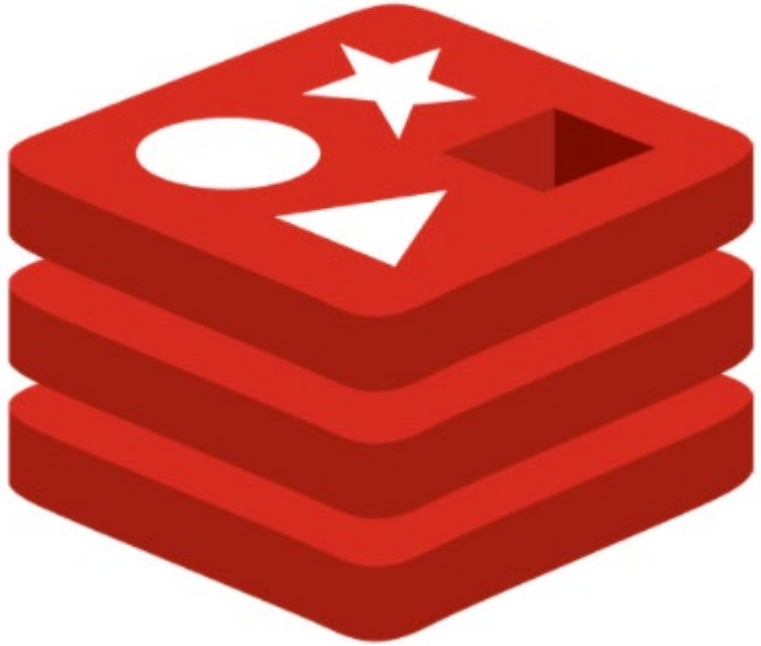
MongoDB – это документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. MongoDB – классифицируется как NoSQL и использует JSON-подобные документы и схему базы данных.

SQL (Structured Query Language) — язык структурированных запросов, с помощью него пишутся специальные запросы к базе данных с целью получения данных из базы данных или для манипулирования этими данными.

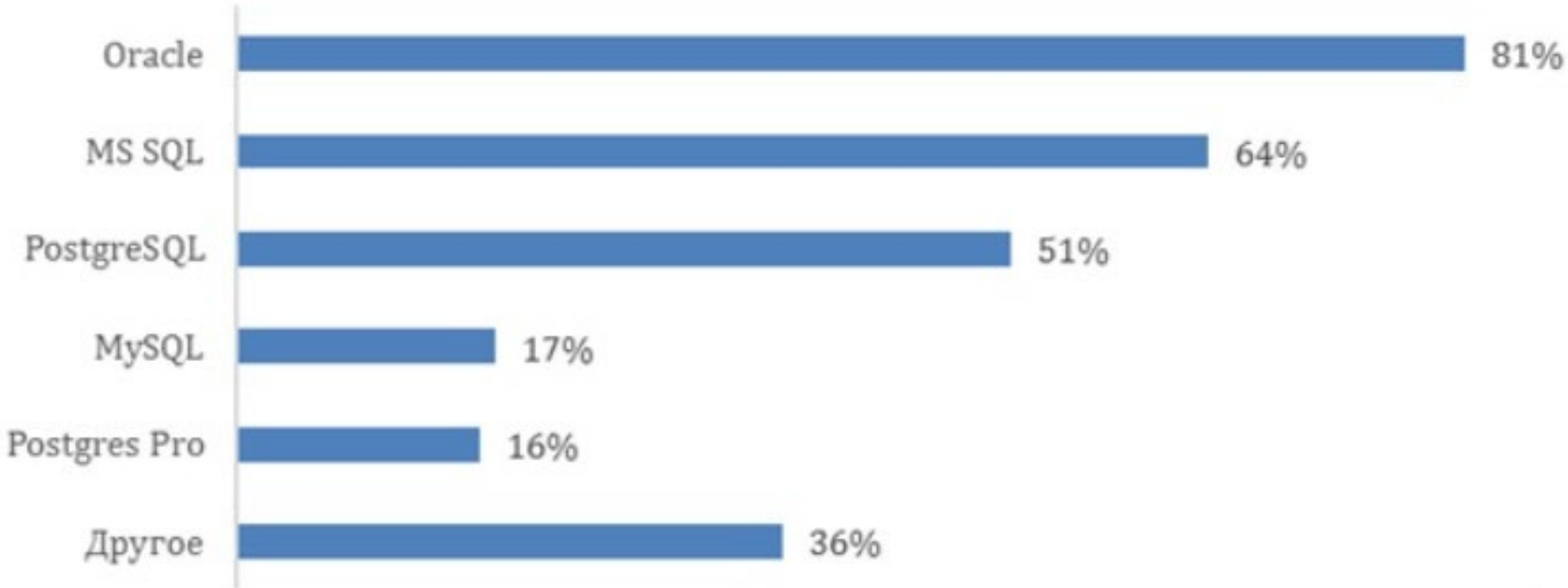
Каждая СУБД хранит файлы базы данных по-своему, т.е. в своем собственном формате, однако для того чтобы нам было легче управлять данными в базе данных был разработан специальный язык, который является стандартом и он позволяет нам, независимо от того в какой СУБД создана база данных, манипулировать данными в этой базе данных. Этот язык назвали SQL.

РАСПРЕДЕЛЁННАЯ ОТКАЗОУСТОЙЧИВАЯ КОЛОНОЧНАЯ СУБД **APACHE CASSANDRA**

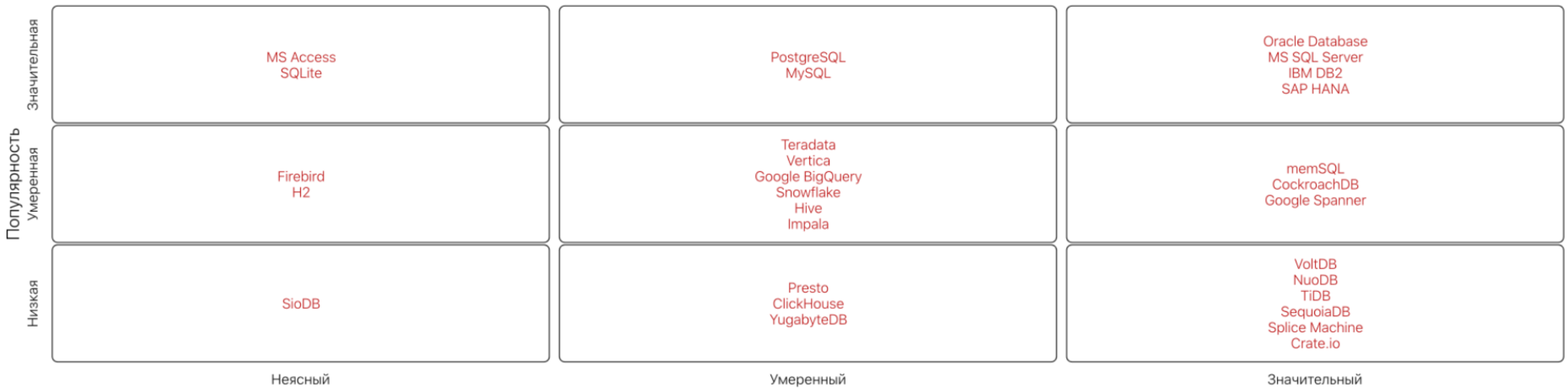
Apache Cassandra — распределенная колоночная система управления базами данных, относящаяся к классу NoSQL. Отлично подходит для надёжного хранения большого количества данных, поступающих с высокой скоростью.



Redis - популярная in-memory база данных, которая используется во многих проектах и может выполнять различные задачи, такие как кэширование, ограничение скорости, хранение сессий


Oracle
81%
MS SQL
64%
PostgreSQL
51%
MySQL
17%
Postgres Pro
16%
Другое
36%

Основные игроки и их продукты

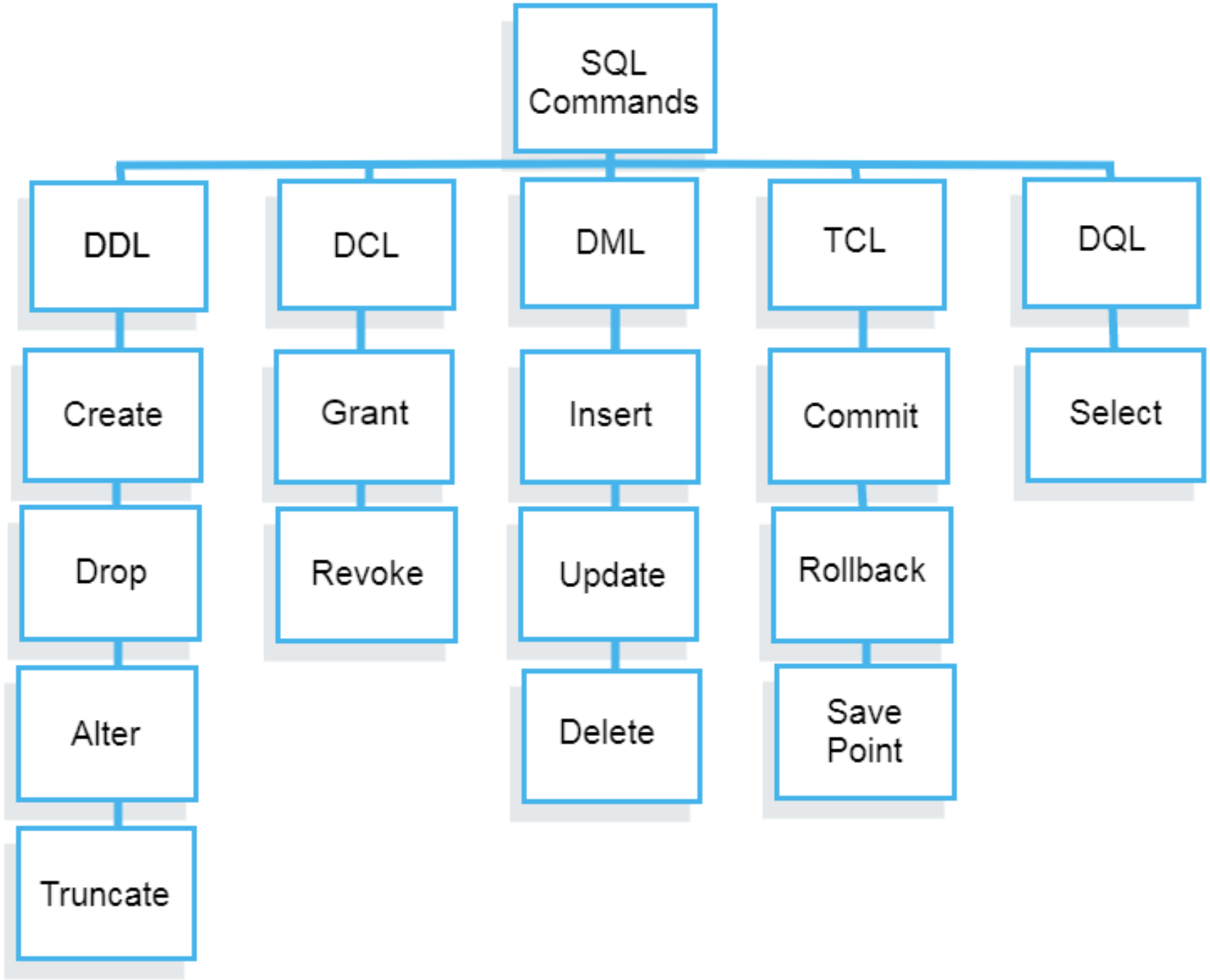


Лекция 8. Использование языка SQL (DDL) для создания структур данных

- 1970 г. – д-р Эдгар Ф. «Тед» Кодд описал реляционную модель для баз данных.
- 1974 — появился язык структурированных запросов.
- 1978 — IBM выпустила продукт под названием System/R.
- 1986 — IBM разработала прототип реляционной базы данных, стандартизированной ANSI.
- 1989 — Выпущена первая версия SQL.
- 1999 — запущен SQL 3 с такими функциями, как триггеры, объектная ориентация и т. д.
- SQL2003 — оконные функции, функции, связанные с XML, и т. д.
- SQL2006 — поддержка языка запросов XML.
- Улучшенная в SQL2011 поддержка временных баз данных.

Типы SQL

- Язык определения данных (DDL)
- Язык манипулирования данными (DML)
- Язык управления данными (DCL)
- Язык управления транзакциями (TCL)
- Язык запросов данных (DQL)


SQL Commands
DDL
DCL
DML
TCL
DQL
Create
Drop
Alter
Truncate
Grant
Revoke
Insert
Update
Delete
Commit
Rollback
Save Point
Select

Язык SQL

Команды DML
(Data Manipulation Language)

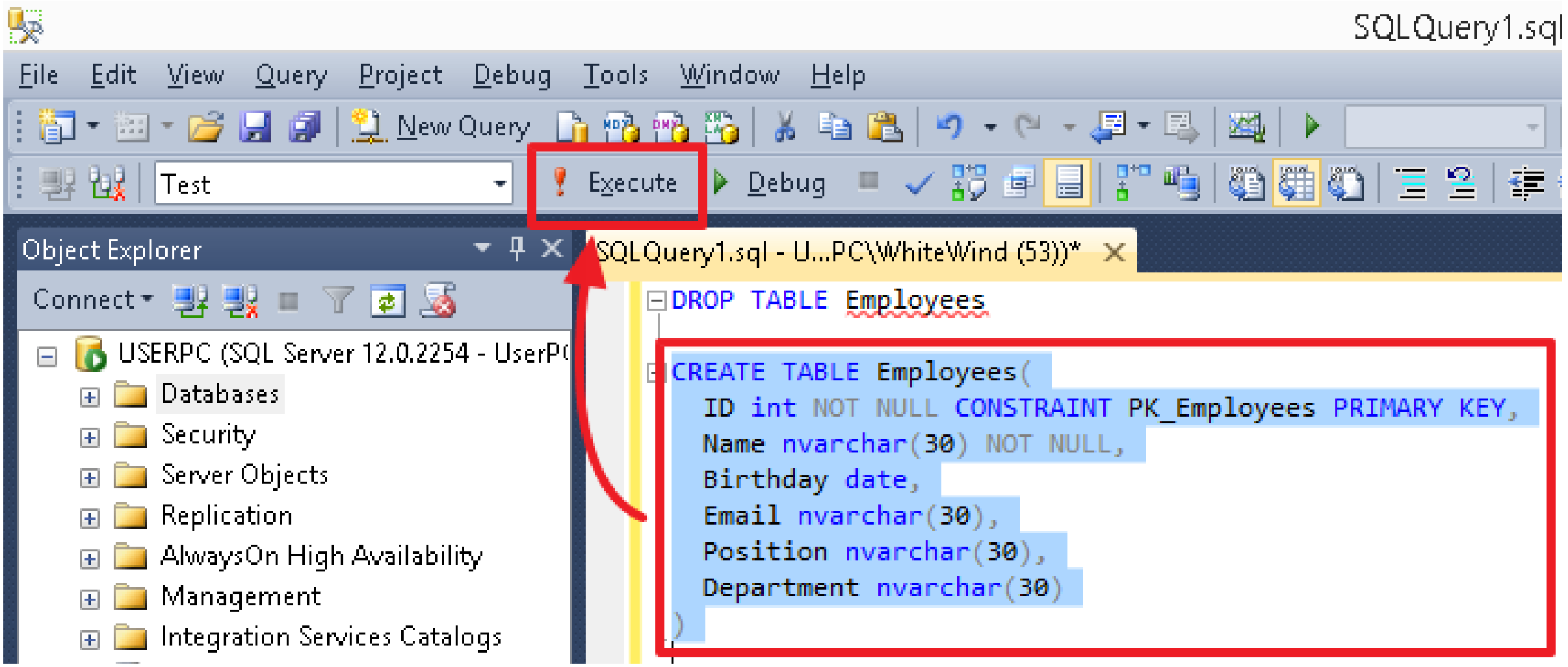
- ✓ SELECT – Выборка (получение) данных
- ✓ INSERT – Вставка данных
- ✓ UPDATE – Редактирование (изменение) данных
- ✓ DELETE – Удаление (строчек) данных

Команды DDL
(Data Definition Language)

- ✓ CREATE – Создание объекта базы данных (например, таблицы) или самой базы данных
- ✓ ALTER – Редактирование объекта
- ✓ DROP – Удаление объекта

Команды TCL
(Transaction Control Language)

Команды DCL
(Data Control Language)


SQLQuery1.sql
File Edit View Query Project Debug Tools Window Help
New Query
Test
Execute
Debug
Object Explorer
Connect
USERPC (SQL Server 12.0.2254 - UserPC
Databases
Security
Server Objects
Replication
AlwaysOn High Availability
Management
Integration Services Catalogs
SQLQuery1.sql - U...PC\WhiteWind (53))*
DROP TABLE Employees
CREATE TABLE Employees(
ID int NOT NULL CONSTRAINT PK_Employees PRIMARY KEY,
Name nvarchar(30) NOT NULL,
Birthday date,
Email nvarchar(30),
Position nvarchar(30),
Department nvarchar(30)
)

Простую базу данных (без указания дополнительных параметров) можно создать, выполнив следующую команду:

```
CREATE DATABASE Test
```

Удалить базу данных можно командой (стоит быть очень осторожным с данной командой):

```
DROP DATABASE Test
```

Для того, чтобы переключиться на нашу базу данных, можно выполнить команду:

```
USE Test
```

Для примера рассмотрим создание таблицы с данными о сотрудниках

```
CREATE TABLE [Сотрудники](
  [Табельный номер] int,
  [ФИО] nvarchar(30),
  [Дата рождения] date,
  [E-mail] nvarchar(30),
  [Должность] nvarchar(30),
  [Отдел] nvarchar(30)
)
```

Удалить таблицу о сотрудниках

```
DROP TABLE [Сотрудники]
```

Добавление информации в таблицу о сотрудниках

```
INSERT Employees(ID,Position,Department) VALUES
(1000,N'Директор',N'Администрация'),
(1001,N'Программист',N'ИТ'),
(1002,N'Бухгалтер',N'Бухгалтерия'),
(1003,N'Старший программист',N'ИТ')
```

Первичный ключ

```
ALTER TABLE Employees ADD CONSTRAINT PK_Employees PRIMARY KEY(ID)
```

```
ALTER TABLE имя_таблицы ADD CONSTRAINT имя_ограничения PRIMARY KEY(поле1,поле2,…)
```

Первичный ключ при создании таблицы

```
CREATE TABLE Employees(
  ID int NOT NULL,
  Name nvarchar(30) NOT NULL,
  Birthday date,
  Email nvarchar(30),
  Position nvarchar(30),
  Department nvarchar(30),
  CONSTRAINT PK_Employees PRIMARY KEY(ID)
```

```
INSERT Employees(ID,Position,Department,Name) VALUES
(1000,N'Директор',N'Администрация',N'Иванов И.И.'),
(1001,N'Программист',N'ИТ',N'Петров П.П.'),
(1002,N'Бухгалтер',N'Бухгалтерия',N'Сидоров С.С.'),
(1003,N'Старший программист',N'ИТ',N'Андреев А.А.')
```

```
DROP TABLE Employees

CREATE TABLE Employees(
  ID int NOT NULL,
  Name nvarchar(30),
  Birthday date,
  Email nvarchar(30),
  PositionID int,
  DepartmentID int,
  ManagerID int,
CONSTRAINT PK_Employees PRIMARY KEY (ID),
CONSTRAINT FK_Employees_DepartmentID FOREIGN KEY(DepartmentID) REFERENCES Departments(ID)
ON DELETE CASCADE,
CONSTRAINT FK_Employees_PositionID FOREIGN KEY(PositionID) REFERENCES Positions(ID),
CONSTRAINT FK_Employees_ManagerID FOREIGN KEY (ManagerID) REFERENCES Employees(ID)
)

INSERT Employees (ID,Name,Birthday,PositionID,DepartmentID,ManagerID)VALUES
(1000,N'Иванов И.И.','19550219',2,1,NULL),
(1001,N'Петров П.П.','19831203',3,3,1003),
(1002,N'Сидоров С.С.','19760607',1,2,1000),
(1003,N'Андреев А.А.','19820417',4,3,1000)
```

Прочие ограничения – UNIQUE, DEFAULT, CHECK

```
UPDATE Employees SET Email='i.ivanov@test.tt' WHERE ID=1000
UPDATE Employees SET Email='p.petrov@test.tt' WHERE ID=1001
UPDATE Employees SET Email='s.sidorov@test.tt' WHERE ID=1002
UPDATE Employees SET Email='a.andreev@test.tt' WHERE ID=1003
```

```
ALTER TABLE Employees ADD CONSTRAINT UQ_Employees_Email UNIQUE(Email)
```

```
ALTER TABLE имя_таблицы ADD CONSTRAINT имя_ограничения UNIQUE(поле1,поле2,…)
```

```
ALTER TABLE Employees ADD HireDate date NOT NULL DEFAULT SYSDATETIME()
```

Индексы, создаваемые при создании ограничений PRIMARY KEY и UNIQUE

```
ALTER TABLE имя_таблицы ADD CONSTRAINT имя_ограничения
PRIMARY KEY NONCLUSTERED(поле1,поле2,…)
```

```
ALTER TABLE Employees DROP CONSTRAINT PK_Employees
ALTER TABLE Employees DROP CONSTRAINT UQ_Employees_Email
```

```
ALTER TABLE Employees ADD CONSTRAINT PK_Employees PRIMARY KEY NONCLUSTERED (ID)
ALTER TABLE Employees ADD CONSTRAINT UQ_Employees_Email UNIQUE CLUSTERED (Email)
```

Ограничения при создании/изменении БД:

PRIMARY KEY – первичный ключ;
FOREIGN KEY – настройка связей и контроль ссылочной целостности данных;
UNIQUE – позволяет создать уникальность;
CHECK – позволяет осуществлять корректность введенных данных;
DEFAULT – позволяет задать значение по умолчанию;
Так же стоит отметить, что все ограничения можно удалить, используя команду «**ALTER TABLE** имя_таблицы **DROP CONSTRAINT** имя_ограничения».

Создание самостоятельных индексов

```
CREATE INDEX IDX_Employees_Name ON Employees(Name)
```

```
CREATE UNIQUE NONCLUSTERED INDEX UQ_Employees_EmailDesc ON Employees(Email DESC)
```

Удаление самостоятельных индексов

```
DROP INDEX IDX_Employees_Name ON Employees
```

Лекция 9. Запросы на языке SQL (DML)

Язык SQL. Оператор SELECT

SELECT *<Столбец1>, <Столбец2>, … <СтолбецN>*
FROM *<Имя_таблицы>*
JOIN *–ы (Соединения дополнительных таблиц)*
WHERE *<Условие или условия отбора данных из таблицы>*
GROUP BY *<Признак группировки>*
HAVING *<Условия отбора на основе данных группировки>*
ORDER BY *<Столбец1>, …*

Язык SQL. Оператор SELECT

```
SELECT *
  FROM Persons
WHERE FILIALID = 1 OR FILIALID = 2
ORDER BY BirthDate
```

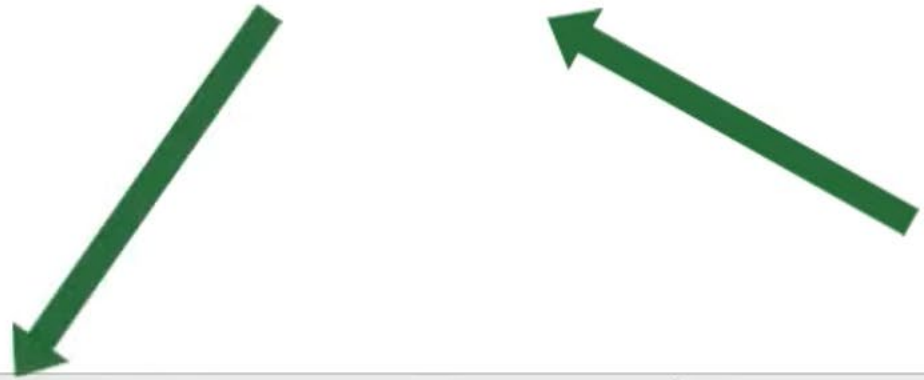
base_sql

Select Database | Content | Query | Structure | Relations | Triggers | Table Info | Table History | Users | Console

TABLES

- products
- users

```
1 SELECT last_name, first_name, birthday FROM users;
```



id	first_name	last_name	birthday	age
1	Дмитрий	Иванов	1996-12-11	20
2	Олег	Лебедев	2000-02-07	17
3	Тимур	Шевченко	1998-04-27	19
4	Светлана	Иванова	1993-08-06	23
5	Олег	Ковалев	2002-02-08	15
6	Алексей	Иванов	1993-08-05	23
7	Алена	Процук	1997-02-28	18

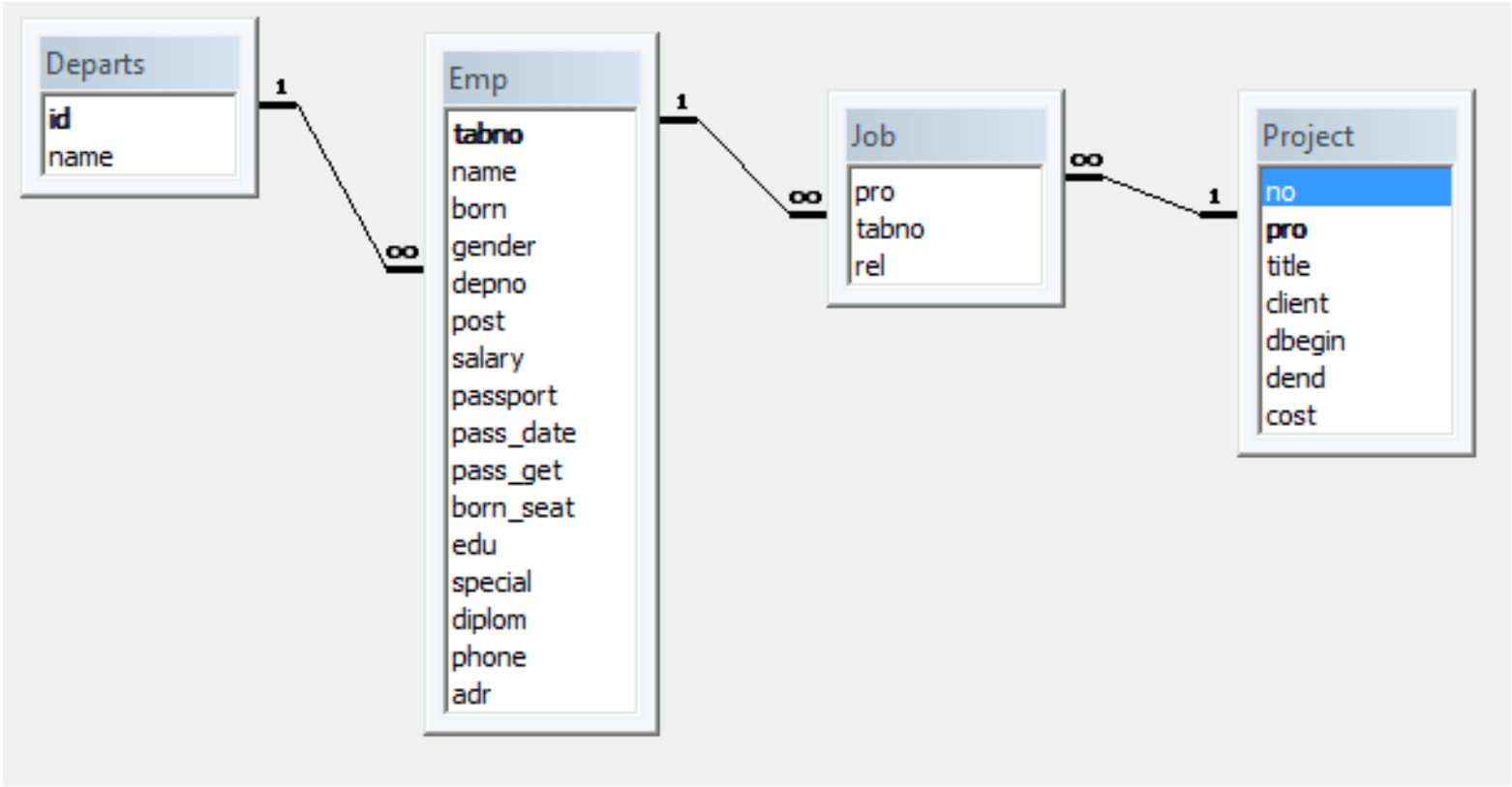
Query Favorites | Query History

last_name	first_name	birthday
Иванов	Дмитрий	1996-12-11
Лебедев	Олег	2000-02-07
Шевченко	Тимур	1998-04-27
Иванова	Светлана	1993-08-06
Ковалев	Олег	2002-02-08
Иванов	Алексей	1993-08-05
Процук	Алена	1997-02-28

TABLE INFORMATION

- created: 10/09/2017
- updated: 10/09/2017
- engine: InnoDB
- rows: 7
- size: 16,0 KiB
- encoding: utf8
- auto_increment: 8

Пример БД: проектная организация



Departs – отделы, Project – проекты,

Emp – сотрудники, Job – участие в проектах.

Пример БД: проектная организация

Departs – отделы:

did – номер отдела, первичный ключ;
name – название отдела, обязательное поле.

Project – проекты:

No – номер проекта, первичный ключ;
title – название проекта, обязательное поле;
pro – краткое название проекта, обязательное уникальное поле;
client – заказчик, обязательное поле;
dbegin – дата начала выполнения проекта, обязательное поле;
dend – дата завершения проекта, обязательное поле;
cost – стоимость проекта, обязательное поле.

Job – участие в проектах:

pro – краткое название проекта, внешний ключ;
tabNo – номер сотрудника, участвующего в проекте, внешний ключ;
rel – роль сотрудника в проекте; может принимать одно из трех значений: 'исполнитель', 'руководитель', 'консультант'.
Первичный ключ – комбинация полей **pro** и **tabNo**.

Команда SELECT – выборка данных

Общий синтаксис:

SELECT [{ ALL | DISTINCT }] { *список_вывода* | * }
  FROM *имя_таблицы1* [*алиас1*] [, *имя_таблицы2* [*алиас2*].,..]
  [WHERE *условие_отбора_записей*]
  [GROUP BY { *имя_поля* | *выражение* }.,..]
  [HAVING *условие_отбора_групп*]
  [UNION [ALL] SELECT …]
  [ORDER BY *имя_поля1* | *целое* [ASC | DESC]
    [, *имя_поля2* | *целое* [ASC | DESC].,..]];

Примеры:

```
select * from departs;
select   name, post   from emp;
```

Формирование списка вывода (проекция)

Общий синтаксис списка вывода:

[{all | distinct}] { * | *выражение1* [*алиас1*] [, *выражение2* [*алиас2*] .,..]}

Список ввода находится между ключевыми словами **SELECT** и **FROM**.

- Вывести все поля всех записей из таблицы Проекты (Project):

```
select * from project;
```

2. Вывести список сотрудников с указанием их должности и № отдела:

```
select depno, name, post
    from emp;
```

3. Вывести список сотрудников с указанием их должности и зарплаты:

```
select name 'ФИО', post 'Должность', salary*0.87 'Зарплата'
    from emp;
```

Упорядочение результата

1. select *
   from Project
   order by dbegin;
2. select depno, name, post
   from emp
   order by depno, name; -- order by 1,2;
3. select name 'ФИО', post 'Должность', salary 'Зарплата'
   from emp
   order by 3 DESC;
4. select depno 'Номер отдела', post 'Должность', salary 'Зарплата'
   from emp
   order by 1, 3 DESC, 2;

Выбор данных из таблицы (селекция)

WHERE – содержит условия выбора отдельных записей. Условие является логическим выражением и может принимать одно из 3-х значений:

- TRUE – истина,
- FALSE – ложь,
- NULL – неизвестное, неопределённое значение (интерпретируется как ложь).

Условие формируется путём применения различных операторов и предикатов.

Операторы сравнения:

=	равно,	<>, !=	не равно,	>	больше,
>=	больше или равно,	<=	меньше или равно,	<	меньше.

Вывести список сотрудников 2-го отдела:

```
select *  from emp
    where depno = 2;
```

Логические операторы

Для формирования условий используются следующие логические операторы:

AND – логическое произведение (И),
OR – логическая сумма (ИЛИ),
NOT – отрицание (НЕ).

Операция И:

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

Операция НЕ:

a	NOT a
0	1
1	0

Операция ИЛИ:

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

Выбор данных из таблицы по условию

1. ```
   select * from emp
       where depno = 2 AND salary > 3000 ;
   ```

2. ```
   select * from emp
       where born > '31/12/1979' AND sex = 'м';
   ```
3. ```
   select * from emp
       where depno=2 OR depno = 5;
   ```

4. ```
   select * from emp
       where (depno=2 OR depno = 5) AND salary >= 3000 ;
   ```

5. ```
   select * from emp
       where NOT (depno=2 OR depno = 5);
   ```
   ```

Предикаты формирования условия

Предикат вхождения в список значений:

имя_поля **IN (*значение1* [, *значение2,…*])**

выражение **IN (*значение1* [, *значение2,…*])**

Примеры:

```
select *
    from emp
    where depno  IN ( 5, 8, 9 ) ;
```

- ```
  select  *
      from  emp
      where  post  IN ( 'инженер', 'ведущий инженер' );
  ```
  ```

Предикаты формирования условия

Предикат вхождения в диапазон:

имя_поля **BETWEEN** ***минимальное_значение*** **AND** ***максимальное_значение***

выражение **BETWEEN** ***минимальное_значение*** **AND** ***максимальное_значение***

Минимальное значение должно быть меньше либо равно максимальному.

Примеры:

```
select *
    from emp
    where depno BETWEEN 2 AND 5 ;
```

- ```
  select *
      from emp
      where salary*0.87 BETWEEN 2000 AND 3000;
  ```
  ```

Предикаты формирования условия

Предикат поиска подстроки: ***имя_поля*** **LIKE '*шаблон*'**

Этот предикат применяется только к полям типа CHAR и VARCHAR. Возможно использование шаблонов:
'_' – один любой символ,
'%' – произвольное количество любых символов (в т.ч., ни одного).

Примеры:

```
1. select *  from emp
      where post  LIKE  '%экономист%' ;

2. select  *  from  emp
      where  post  LIKE  'инженер_%' ;
```

Предикаты формирования условия

Предикат поиска неопределенного значения:

значение **IS [NOT] NULL**

Если значения является неопределенным (NULL), то предикат IS NULL выдаст истину, а предикат IS NOT NULL – ложь.

Примеры:

```
select *
    from emp
    where phone IS NULL ;
```

```
select *
    from project
    where cost IS NOT NULL ;
```

Лекция 10. Объекты баз данных

Агрегирующие функции

COUNT – подсчёт количества строк (значений). Применяется к записям и полям любого типа. Имеет 3 формата вызова:

- **count (*)** – количество строк результата;
- **count (*имя_поля*)** – количество значений указанного поля, не являющихся *NULL*-значениями.
- **count (distinct *имя_поля*)** – количество разных не-NULL значений указанного поля.

MAX, MIN

SUM

AVG

Примеры использования агрегирующих функций

1. ```
   select  max(cost) "Максимальная цена",  min(cost)
   "Минимальная цена"
       from  project;
   ```

2. ```
   select  sum(salary)
       from  emp
       where  depno = 8;
   ```

3. ```
   select  avg(salary)
       from emp
       where  sex = 'Ж';
   ```

4. ```
   select   min(dbegin),  max(dend)
       from  project;
   ```

Группировка данных: предложение GROUP BY

Агрегирующие функции обычно используются совместно с предложением ***GROUP BY***.

Например, следующая команда считает количество сотрудников по отделам:

```
select  depno,  count(*)
        from emp
        group by depno;
```

depno	name	…
1	Белов С.В.	
1	Иванова К.Е.	
1	Седов О.Л.	
2	Волков Н.Е.	
2	Рогов И.Л.	
3	Санина В.П.	
3	Дымова С.Т.	
3	Павлов К.Д.	
3	Орлов Т.Ф.	

depno	count(*)
1	3
2	2
3	4

Примеры использования GROUP BY

1. ```
   select depno, MIN(salary) minsal, MAX(salary) maxsal
       from emp
       group by depno;
   ```
2. ```
   select depno, COUNT(distinct post) cnt
       from emp
       group by depno;
   ```

   ```
   select depno, SUM(salary) allsal
       from emp
       group by depno;
   ```
4. ```
   select post, AVG(salary) avgsal
       from emp
       group by post;
   ```
   ```

Операции реляционной алгебры

Унарные операции:

- **селекция**

  Например:

  ```
  select *
      from emp
      where depno = 5;
  ```

- **проекция**

  Например:

  ```
  select  distinct name,  post,  salary
      from  emp;
  ```

Общий алгоритм выполнения операции *SELECT*

1. Выбор записей из указанной таблицы (***from***).
2. Проверка для каждой записи условия отбора (***where***).
3. Группировка полученных в результате отбора записей (***group by***) и вычисление для этих групп значений агрегирующих функций.
4. Выбор тех групп, которые удовлетворяют условию отбора групп (***having***).
5. Сортировка полученных записей в указанном порядке (***order by***).
6. Извлечение из полученных записей тех полей, которые заданы в списке вывода, и формирование результирующего отношения.

   Если в части FROM указывается 2 и более таблицы, то приведенный алгоритм выполняется для декартова произведения этих таблиц.

Самосоединение

В команде SELECT можно обратиться к одной и той же таблице несколько раз. А для того чтобы исключить соединение записи таблицы с самой собой в запросе на самосоединение необходимо также указывать условие типа "не равно" (<>, >, <).

Пример использования самосоединения:

Вывести список детей сотрудников, у которых есть младшие братья или сёстры:

```
SELECT e.name, c1.name AS child1, c1.born AS born1,
   c2.name AS child2, c2.born AS born2
FROM children c1, children c2, emp e
   WHERE c1.tabno=e.tabno  -- первое условие соединения
   AND c1.tabno=c2.tabno   -- второе условие соединения
   AND c1.born<c2.born     -- условие исключения
ORDER BY 1, 3;
```

Подзапросы

Подзапрос – это запрос SELECT, расположенный внутри другой команды. Подзапросы можно разделить на следующие группы в зависимости от возвращаемых результатов:

✓**скалярные**

✓**векторные**

✓**табличные**

Подзапросы бывают:

✓ **некоррелированные** – не содержат ссылки на запрос верхнего уровня; вычисляются один раз для запроса верхнего уровня;

✓**коррелированные –** содержат условия, зависящие от значений полей в основном запросе; вычисляются для каждой строки запроса верхнего уровня.

Расположение подзапросов в командах DML

В команде **INSERT**:

- Вместо **VALUES**, например, добавление данных из одной таблицы в другую:

```
insert into emp select * from new_emp;
```

В команде **UPDATE**:

- в части **WHERE** для вычисления условий, например, повышение зарплаты на 10% всем участникам проектов:

```
update emp set salary = salary*1.1
    where tabNo IN (select tabNo from job);
```

- в части **SET** для вычисления значений полей, например, повышение зарплаты на 10% за каждое участие сотрудника в проекте:

```
update emp e set salary = salary*(1+(select count(*)/10 from job j
                                    where j.tabNo = e.tabNo) );
```

В команде **DELETE**:

- в части **WHERE** для вычисления условий, например, удаление сведений об участии в закончившихся проектах:

```
delete from job
    where pro IN (select pro from project where dend < sysdate);
```

Расположение подзапросов в команде SELECT

- Чаще всего подзапрос располагается в части **WHERE.**

Пример 1:

```
select * from emp
    where salary > (select avg(salary) from
emp);
```

Пример 2. :

```
select * from emp
 where salary > ALL (select avg(salary) from
emp  group by depno);
```

Примеры использования подзапросов в части WHERE

Выдать список сотрудников, имеющих детей:

а) с помощью операции соединения таблиц:

```
SELECT  e.*
    FROM  emp e, children c
    WHERE e.tabno=c.tabno;
```

б) с помощью некоррелированного векторного подзапроса:

```
SELECT  *
    FROM  emp
    WHERE  tabno  IN  (SELECT  tabno FROM children);
```

в) с помощью коррелированного табличного подзапроса:

```
SELECT  *
    FROM  emp  e
    WHERE  EXISTS (SELECT * FROM children c
                   WHERE e.tabno=c.tabno);
```

Расположение подзапросов в команде SELECT

➢ Подзапрос в части **FROM.**

Например,

```
select * from emp e
    where salary > (select avg(salary) from emp m
                    where m.depno = e.depno);
```

Это работает долго, т.к. коррелированный подзапрос вычисляется для каждой

строки основного запроса. Можно ускорить выполнение данного запроса:

```
select *
    from emp e,
    (select depno, avg(salary) sal
            from emp
            group by depno) m     -- подзапрос вычисляется 1 раз
    where m.depno = e.depno
            and salary > sal;
```

Расположение подзапросов в команде SELECT

- Подзапрос в части **HAVING.**
  Например,

```
select depno, avg(salary) sal
    from emp
    group by depno
    having avg(salary) < (select avg(salary) from emp);
```

- Подзапрос в части **SELECT.**
  Например,

```
 select depno, name,
    (select count(*) from job j where j.tabno = e.tabno) cnt
from emp e;
```

Этот запрос выведет даже тех сотрудников, которые не участвуют в проектах (для них **cnt** будет равен 0).

Представления

Представление (view, обзор) – это хранимый запрос, создаваемый на основе команды *SELECT.*

Назначение представлений:

- Хранение сложных запросов.
- Представление данных в виде, удобном пользователю.
- Сокрытие конфиденциальной информации.
- Предоставление дифференцированного доступа к данным.

Создание представления выполняется командой **CREATE VIEW**:

```
CREATE  [ OR  REPLACE ]  VIEW <имя представления>
                              [ (<список имён столбцов>) ]
AS  <запрос>  [ WITH  CHECK  OPTION ];
```

Запрос (команда SELECT), на основании которого создаётся представление, называется **определяющим запросом**.

Представления: примеры

1. CREATE VIEW employees
   AS SELECT tabno, depno, name, post, born, phone
   FROM emp;

2. CREATE VIEW pro_stat
   AS SELECT title, e.name,
      (select count(*) from job j where j.pro=p.pro and rel='исполнитель')
      jobs,
      (select count(*) from job j where j.pro=p.pro and rel='консультант')
      consult
   FROM emp e, project p, job j
      where e.tabno=j.tabno and j.pro=p.pro
      and j.rel='руководитель';

Оператор CASE

Оператор **CASE** может быть использован в одной из двух синтаксических форм записи:

1-я форма:

```
CASE <проверяемое выражение>
    WHEN <сравниваемое выражение 1> THEN <возвращаемое значение 1>
    …
    WHEN <сравниваемое выражение N> THEN <возвращаемое значение N>
    [ELSE <возвращаемое значение>]
END
```

2-я форма:

```
CASE
        WHEN <предикат 1> THEN <возвращаемое значение 1>
        …
        WHEN <предикат N> THEN <возвращаемое значение N>
        [ELSE <возвращаемое значение>]
END
```

Лекция 11. NoSQL хранилища данных

NoSQL

- NoSQL (от англ. **not only SQL** — не только SQL) — термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных

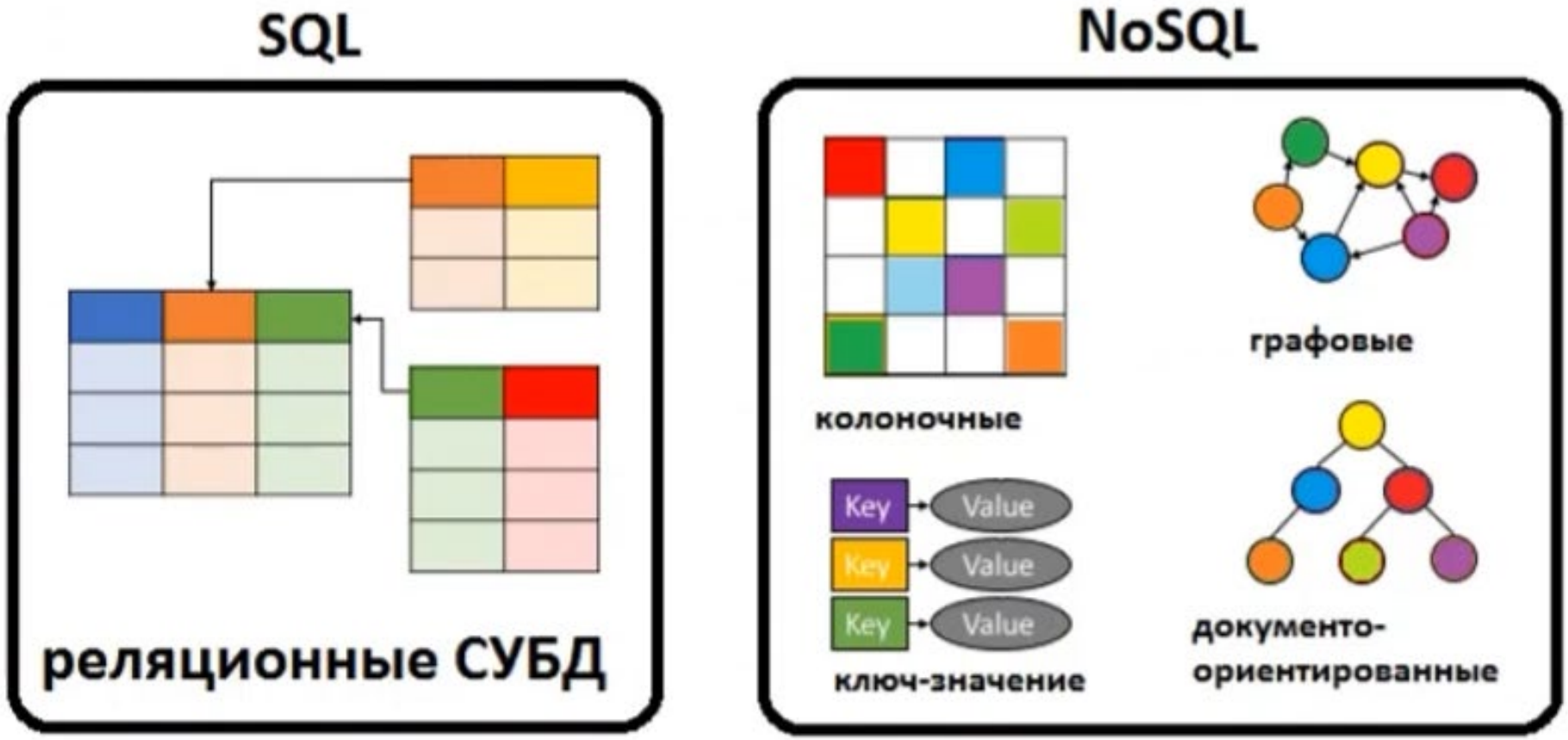
Причины возникновения NoSQL

- Потребность в распределенных СУБД
- Потребность в быстрой работе с данными
- Некоторые часто встречаемые задачи можно моделировать проще

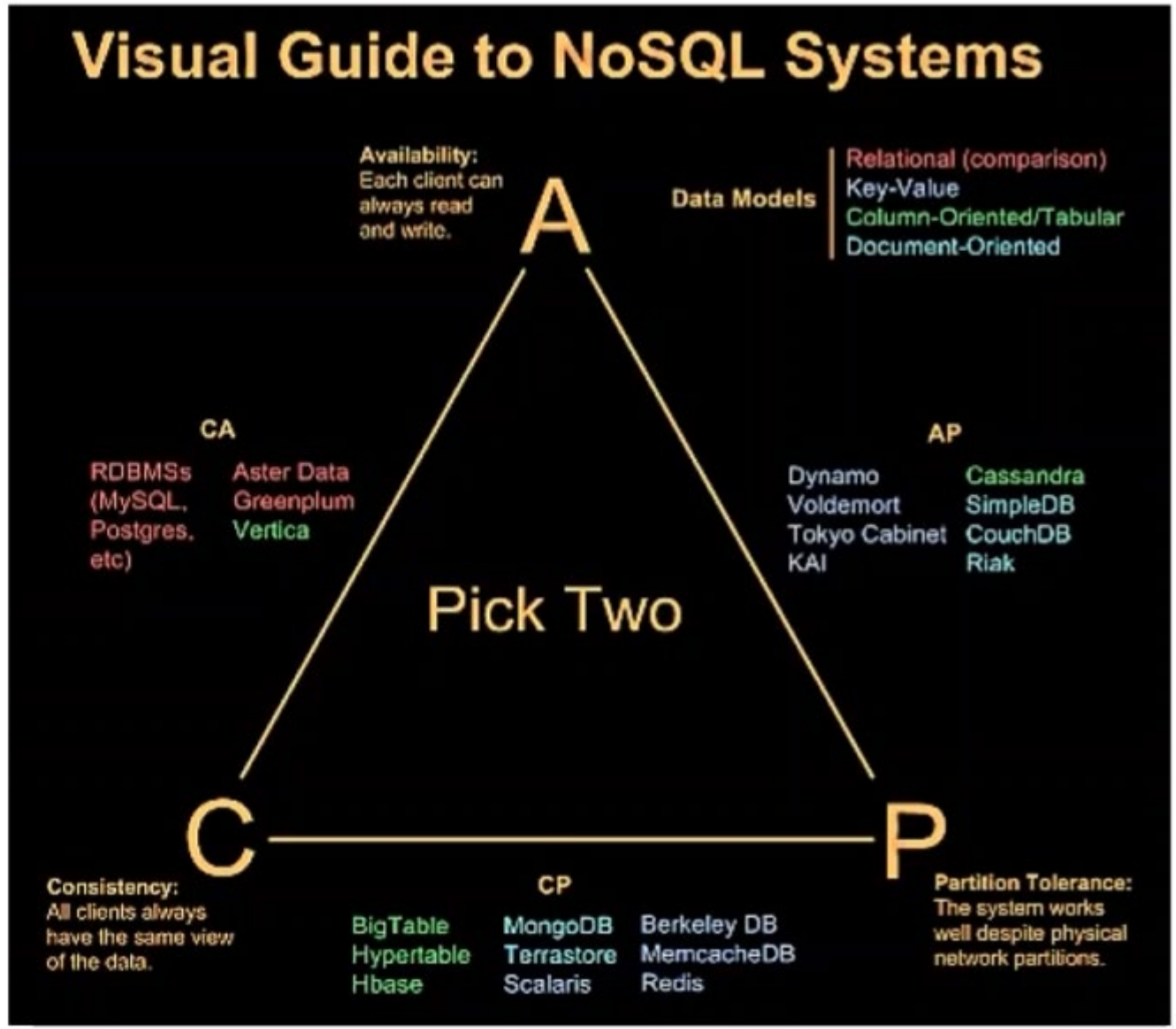
Принцип NoSQL СУБД - BASE

- **Базовая доступность** (basic availability) — каждый запрос гарантированно завершается (успешно или безуспешно).
- **Гибкое состояние** (soft state) — состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.
- **Согласованность в конечном счёте** (*eventual consistency*) — данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время.

Карта СУБД


SQL
реляционные СУБД
NoSQL
колоночные
графовые
Key
Value
Key
Value
Key
Value
ключ-значение
документо-
ориентированные

Теоретические основы NoSQL



Хранилище «ключ-значение»

- Модель данных: множество пар ключ-значение
- Для БД содержание значение непрозрачно (просто какой-то BLOB)

Применение:

- Хранение кеша
- Хранение пользовательских сессий

Примеры «ключ-значение»

- Tarantool

- Redis

- Dynamo DB

Графовые базы данных

- Навеяны теорией графов G=(V, E) от математиков 18го века
- Хорошо моделируют сложные данные
- Модель данных: узлы, ребра и их атрибуты

- Пример: Neo4j

Хранение по строкам и столбцам

Хранение по столбцам
Чтение 3х столбцов

Хранение по строкам
Чтение всех столбцов

- **Key-Value Store**
  - навеяны Dynamo DB от Amazon
  - модель данных: множество пар ключ-значение
  - для БД содержание значение непрозрачно (просто какой-то BLOB)

key	value
firstName	Bugs
lastName	Bunny
location	Earth

- Примеры:
  - Voldemort
  - Redis
  - Riak

- **Document-oriented store**
  - навеяны Lotus Notes от IBM
  - модель данных: множество множеств ключ-значение
  - для БД содержание значение прозрачно

```
{
"id":1,
"name":"Martin",
"billingAddress":[{"city":"Chicago"}]
}

// in orders
{
"id":99,
"customerId":1,
"orderItems":[
  {
  "productId":27,
  "price": 32.45,
  "productName": "NoSQL Distilled"
    }
  ],
"shippingAddress":[{"city":"Chicago"}]
"orderPayment":[
  {
    "ccinfo":"1000-1000-1000-1000",
    "txnId":"abelif879rft",
    "billingAddress": {"city": "Chicago"}
  }
  ],
}
```

- Примеры:
  - CouchDB
  - MongoDB

Лекция 12. Хранилища больших данных ключ-значение

Базы данных «ключ-значение»

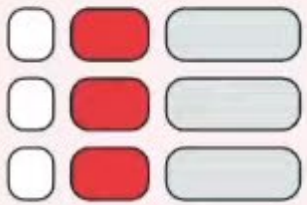
База данных на основе пар «ключ-значение» – это тип нереляционных баз данных, в котором для хранения данных используется простой метод «ключ-значение». База данных на основе пар «ключ-значение» хранит данные как совокупность пар «ключ-значение», в которых ключ служит уникальным идентификатором. Как ключи, так и значения могут представлять собой что угодно: от простых до сложных составных объектов. Базы данных с использованием пар «ключ-значение» поддерживают высокую разделяемость и обеспечивают беспрецедентное горизонтальное масштабирование, недостижимое при использовании других типов баз данных

DynamoDB — система управления базами данных класса NoSQL в формате «ключ — значение»

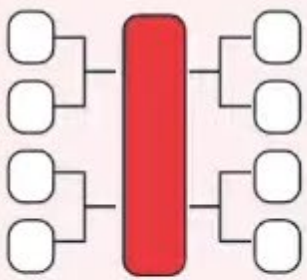
Что такое Redis?

SQL Database

Relational

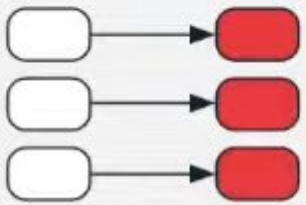


Analiticals (OLAP)

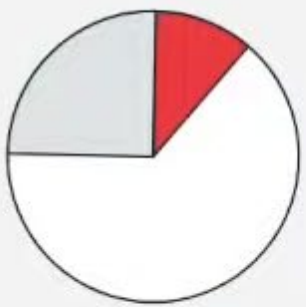


Non-SQL Database

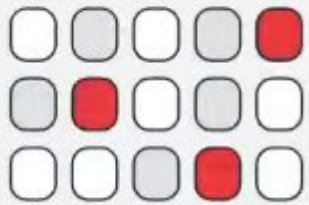
Key-Value



Graph



Column-Family



Document

Характеристики БД «ключ-значение»

Основные характеристики Redis

- Хранит все данные в оперативной памяти
- Периодически сохраняет данные на диске
- Представляет данные в виде структур 5 типов

Сравнение структур данных реляционных баз и Redis

Реляционная база	Redis
Таблицы	Строки
	Списки
	Хеши
	Множества
	Упорядоченные множества

Примеры использования Redis

- Ключ-значение
- Кеширование
- Поддержка типов
- Хранение сессий
- Очереди
- Синхронизация между сервисами
- Lua
- TTL

Типы данных Redis

- **Строки** (strings).
- **Списки** (lists).
- **Множества** (sets).
- **Хеш-таблицы** (hashes).
- **Упорядоченные множества** (sorted sets).

Базовые операции Redis

- set <key> <value>
- get <key>
- getset <key> <new_value>
- type <key>
- rename <key> <new_key>
- exists <key>
- keys <key_pattern>
- del <key>
- ttl <key>
- expire <key> <num_sec>
- expireat <key> <timestamp>

Подключение к базе данных Redis

```php
<?php
$redis = new Redis();
$redis->connect("127.0.0.1", 6379);

$response = $redis->set('key', 'value');
//bool(true)

$response = $redis->get('key');
//string(5) "value"

$response = $redis->getSet('key', 'new_value');
//string(5) "value"

$response = $redis->type('key');
//string

$reponse = $redis->expire('key', 10);
//bool(true)

$redis->ttl('key');
//int 10

$redis->close();
```

Операции над строками Redis

- append <key> <value>
- strlen <key>
- getrange <key> <from> <to>
- setrange <key> <from> <value>

Операции над числами Redis

- incr <key>
- incrby <key> <value>
- descr <key>
- descrby <key> <value>

Битовые маски Redis

- setbit <key> *<0 | 1>*
- getbit <key> *<offset>*

Основные операции над списками Redis

- rpush, lpush
- lrange
- llen
- lpop, rpop

Основные операции над множествами Redis

- sadd
- scard
- sdiff
- sdiffstore
- sinter
- sismember
- smembers
- smove
- srandmember
- sunion

Основные операции над упорядоч. множествами Redis

- zadd
- zrange <key> <start> <end>
- zrevrange
- zrangebyscore

Основные операции над хешами Redis

- hset <key> <field> <value>
- hkeys <key>
- hvals <key>
- hgetall <key>
- hincrby <key> <field> <incr_by_value>

Транзакции в Redis

- MULTI — начать запись команд для транзакции.
- EXEC — выполнить записанные команды.
- DISCARD — удалить все записанные команды.
- WATCH — команда, обеспечивающая поведение типа «check-and-set» (CAS) — транзакция выполняется только в случае, если другие клиенты не изменили значение переменной. Иначе EXEC не выполнит записанные команды.

Pipelining в Redis

```php
<?php
$redis = new Redis();

// Opens up the pipeline
$pipe = $redis->multi(Redis::PIPELINE);

// Loops through the data and performs actions
foreach ($users as $user_id => $username) {
    // Increment the number of times the user record has been accessed
    $pipe->incr('accessed:' . $user_id);

    // Pulls the user record
    $pipe->get('user:' . $user_id);
}

// Executes all of the commands in one shot
$users = $pipe->exec();

// Dumps our data
print_r($users);
```

Pub/Sub в Redis

```php
<?php
function myTest() {
        $arg_list = func_get_args();
        print_r($arg_list);
}

$redis = new Redis();
$redis->connect('127.0.0.1', 6379);
$redis->subscribe(["test", "test1"], 'myTest');
```

```php
<?php
$redis = new Redis();
$redis->connect('127.0.0.1', 6379);
$redis->publish("test", 'Hello Test!! '.rand(1, 100));
```

Лекция 13. Документо-ориентированные хранилища больших данных

MongoDB — документо-ориентированная система управления базами данных с открытым исходным кодом. Для хранения данных используется JSON-подобный формат. Эта СУБД отличается высокой доступностью, масштабируемостью и безопасностью

Что есть в MongoDB:

- ❑ гибкий язык для формирования запросов
- ❑ динамические запросы
- ❑ индексация коллекций
- ❑ профилирование запросов
- ❑ журналирование операций записи
- ❑ отказоустойчивость и масштабируемость
- ❑ асинхронная репликация и шардинг
- ❑ MapReduce
- ❑ полнотекстовый поиск с поддержкой морфологии

Главные особенности MongoDB:

- Это кроссплатформенная документоориентированная база данных NoSQL с открытым исходным кодом.
- Она не требует описания схемы таблиц, как в реляционных БД. Данные хранятся в виде коллекций и документов.
- Между коллекциями нет сложных соединений типа JOIN, как между таблицами реляционных БД. Обычно соединение производится при сохранении данных путем объединения документов.
- Данные хранятся в формате BSON (бинарные JSON-подобные документы).
- У коллекций не обязательно должна быть схожая структура. У одного документа может быть один набор полей, в то время как у другого документа — совершенно другой (как тип, так и количество полей).

Пример документа MongoDB

```
{
        _id : <ObjectId> ,

        CustomerName : Guru99 ,

        Order:
                {

                }
}
```

```
OrderID: 111
Product: ProductA
Quantity: 5
```

Example of how data can be embedded in a document

Структура хранилища MongoDB

СУБД MongoDB полагается на концепции базы данных, коллекций и документов. Рассмотрим основные термины, а для лучшего понимания сравним их с терминами из языка структурированных запросов (SQL):

- База данных — это физический контейнер для коллекций.
- Коллекция — группа документов MongoDB. В терминологии SQL это соответствует таблице.
- Документ — запись в коллекции MongoDB, набор пар ключ-значение. В терминологии SQL это похоже на строку в таблице базы данных.
- Поле — ключ в документе. В терминологии SQL похоже на столбец в таблице.
- Встроенный документ — в терминологии SQL похоже на создание связей между несколькими таблицами, по которым разбросаны данные, что делается операциями JOIN.

Лекция 14. Графовые хранилища больших данных

Neo4j — это

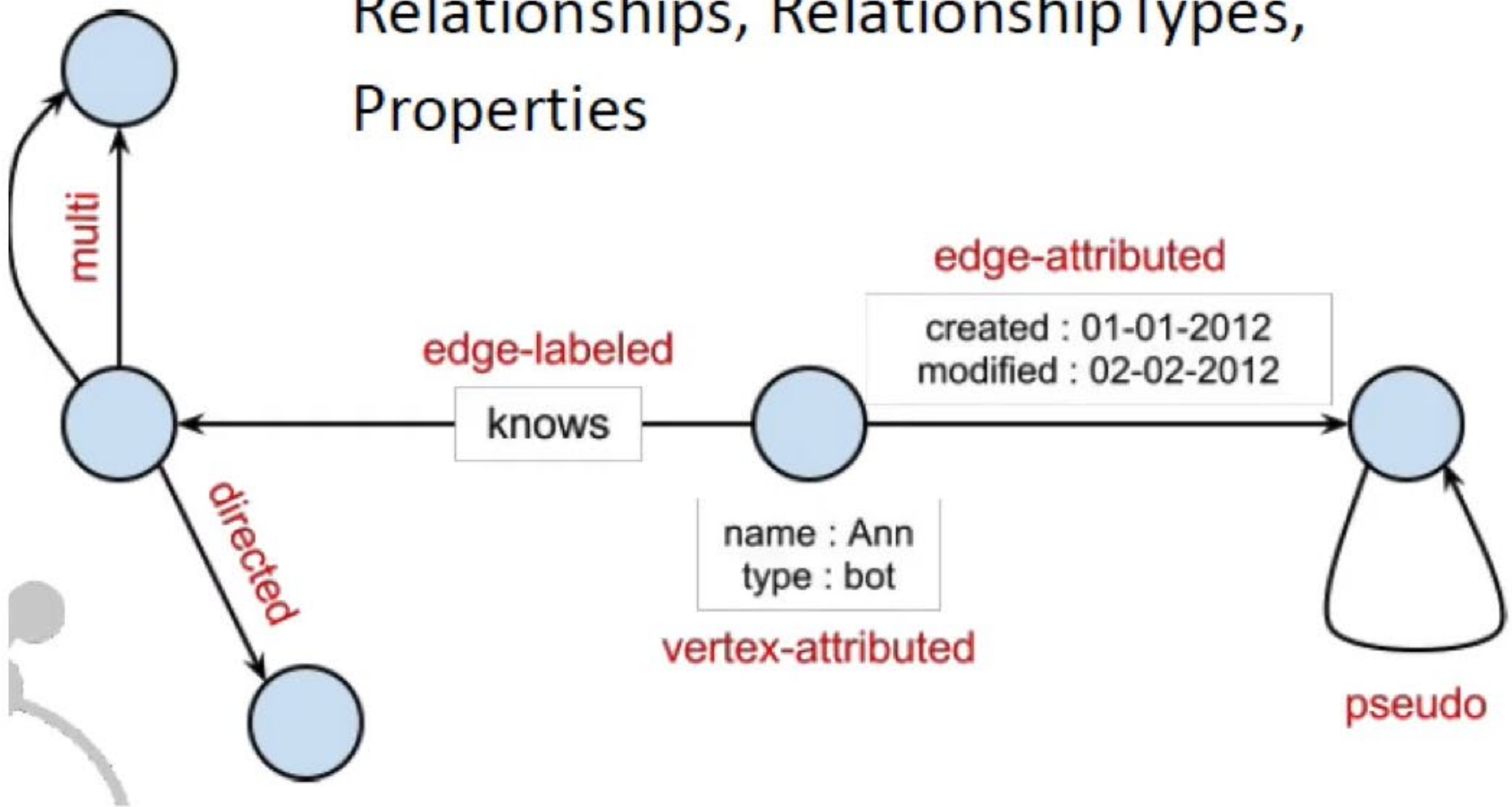
- Встраиваемая Java СуБД
- Полностью ACID
- Фреймворк построения индексов
- Поддержка 24/7 с 2003 года
- Высоко доступная кластеризация (CA)
- Большое комьюнити

Схема данных Neo4j

Nodes,

Relationships, RelationshipTypes,

Properties



Интерфейсы Neo4j

- Java, Ruby, Python API
- Embedded Java API
- Blueprints, Gremlin
- Cypher Query

Embedded Java API

- String STORAGE = "/home/foo/data“;
  EmbeddedGraphDatabase graphDb;
  graphDb = new EmbeddedGraphDatabase(STORAGE);
  Node root = graphDb.getReferenceNode();
- Node neo = graphDb.createNode();
  neo.setProperty("name", "Thomas Anderson");
  neo.setProperty("age", 29);
  Node trinity = graphDb.createNode();
  trinity.setProperty("name", "Trinity");
- root.createRelationshipTo(neo, ROOT);
  Relationship r;
  r = neo.createRelationshipTo(trinity, KNOWS);
  r.setProperty("age", "3 days")

Запросы в Neo4j

- Поиск узлов и связей по индексу/ам
- Траверс графа
  - Поиск путей по заданным условиям
  - TraversalDescription td_KNOWS =Traversal.description().breadthFirst().
    relationships(KNOWS, OUTGOING);
    Traverce t = td_KNOWS.traverce(neo);
    Path p; Iterator<Path> paths = t.iterator();
    for (Relationship n : relationships) {
        …
    }

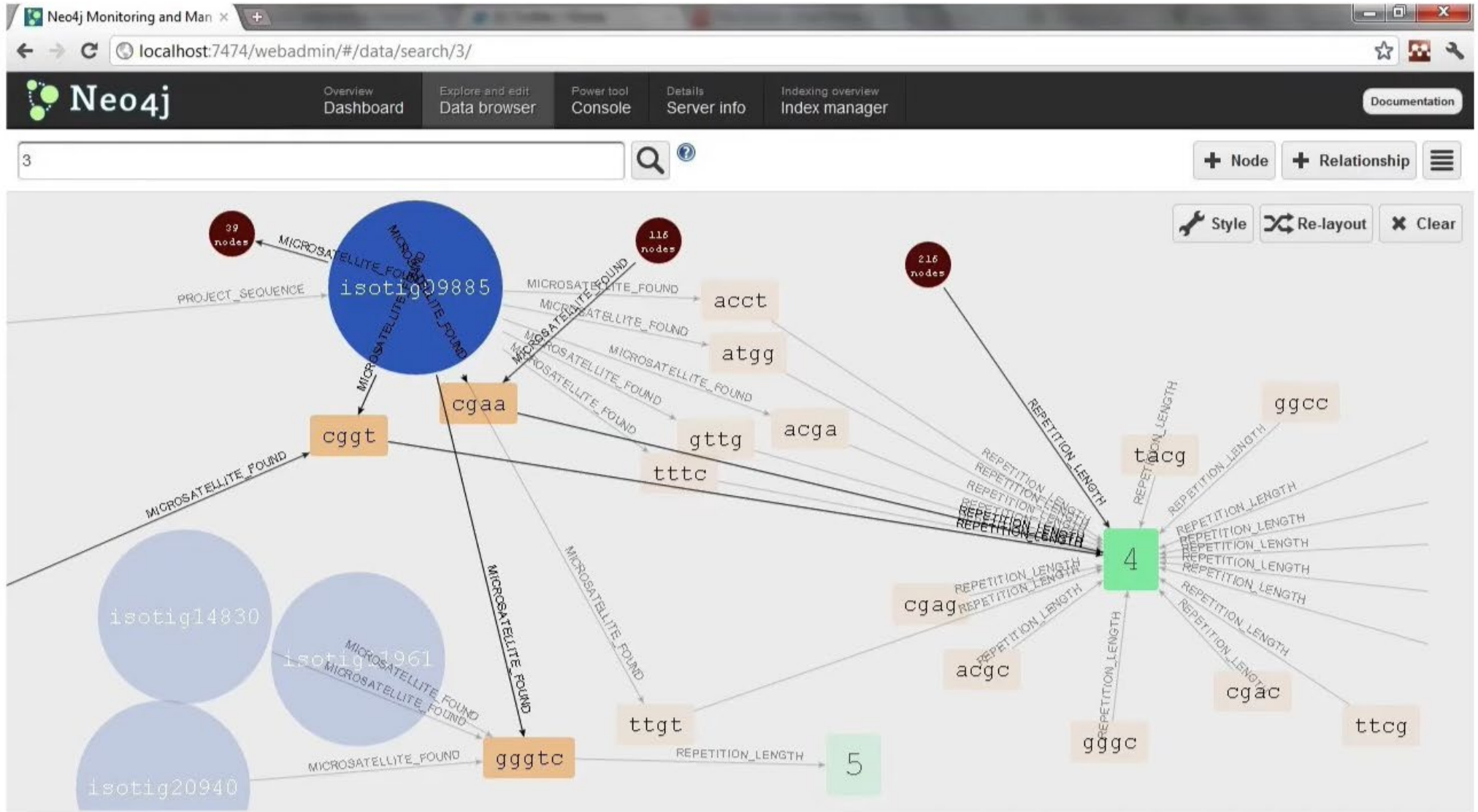
Пример на php (Neo4jPHP)

- ```
  $keanu->relateTo($matrix, 'IN')->save();
  $laurence->relateTo($matrix, 'IN')->save();
  ```
- ```
  $path = $keanu->findPathsTo($kevin)
          ->setMaxDepth(12)
          ->getSinglePath();
  ```

Свойства Neo4j

- **ACID**
- Многопоточность
- Транзакции только на запись
- Встроенные алгоритмы
  - Обход в ширину и в глубину
  - Поиск пути Дейкстры
  - Поиск всех путей

Neo4j WebAdmin



Графовые базы и распределенность

- Проблемы:
  - Операции обхода графа (в ширину и в глубину)
  - Поиск пути
- Возможный вариант
  - Наименьшее сечение

- Pregel (Google)
- GraphLab (CMU/UW)
- Giraph (Apache), used in Facebook
- PowerGraph (GraphLab Org)
- GrapChi (GraphLab Org)
- Cassovary (Twitter)
- Naiad (Microsoft Research)

- GraphBuilder (Intel Labs)
- Neo4j (Neo Technology)
- Grappa (UW)
- MLBase (Berkeley)
- GraphX (Berkeley)
- Stinger (Georgia Tech)
- bigML
- DEX (Sparsity Technologies)

РАЗДЕЛ 3
ИНТЕЛЛЕКТУАЛЬНЫЕ ПЛАТФОРМЫ АНАЛИЗА БОЛЬШИХ ДАННЫХ

Лекция 15. Облачные платформы интеллектуального анализа данных

Идеи облачных вычислений

Первые идеи об использовании облачных вычислений (Cloud Computing) как публичной услуги были предложены еще в 1960-х известным ученым в области информационных технологий, изобретателем языка Lisp, профессором MIT и Стэнфордского университета Джоном Маккарти (John McCarthy). Понятие облака (cloud) уже давно ассоциируется с метафорическим изображением Интернета, с помощью которого доступны некоторые сервисы. Облачные вычисления – это практическая реализация данной идеи. Облачные вычисления основаны на масштабированных и виртуализованных ресурсах (данных и программах), которые доступны пользователям через Интернет и реализуются на базе мощных центров обработки данных (data centers).


Облачные вычисления
SOA
Web 2.0
Системы разработки (development framework)
SaaS
Распределенные масштабируемые вычисления
ASP
grid-вычисления
Виртуализация
"utility computing"
ПО с открытым кодом

Услуги облачных вычислений

Storage-as-a-Service ("хранение как сервис")

Это, пожалуй, самый простой из СС-сервисов, представляющий собой дисковое пространство по требованию. Каждый из нас когда-нибудь сталкивался с ситуацией, когда на мониторе появлялось зловещее предупреждение: "Логический диск заполнен, чтобы освободить место, удалите ненужные программы или данные". Услуга Storage-as-a-Service дает возможность сохранять данные во внешнем хранилище, в "облаке". Для Вас, оно будет выглядеть, как дополнительный логический диск или папка. Сервис является базовым для остальных, поскольку входит в состав практически каждого из них. Примером может служить Google Drive и прочие схожие сервисы.

Database-as-a-Service ("база данных как сервис")
Здесь скорее больше для админов, ибо сия штука предоставляет возможность работать с базами данных, как если бы СУБД была установлена на локальном ресурсе. Причем, в этом случае гораздо легче "расшаривать" проекты между разными исполнителями, не говоря уже о том, сколько деньжат можно сэкономить на компьютерном железе и лицензиях, требуемых для грамотного использования СУБД в крупной или даже средней организации.

Information-as-a-Service ("информация как сервис")
Дает возможность удаленно использовать любые виды информации, которая может меняться ежеминутно или даже ежесекундно.

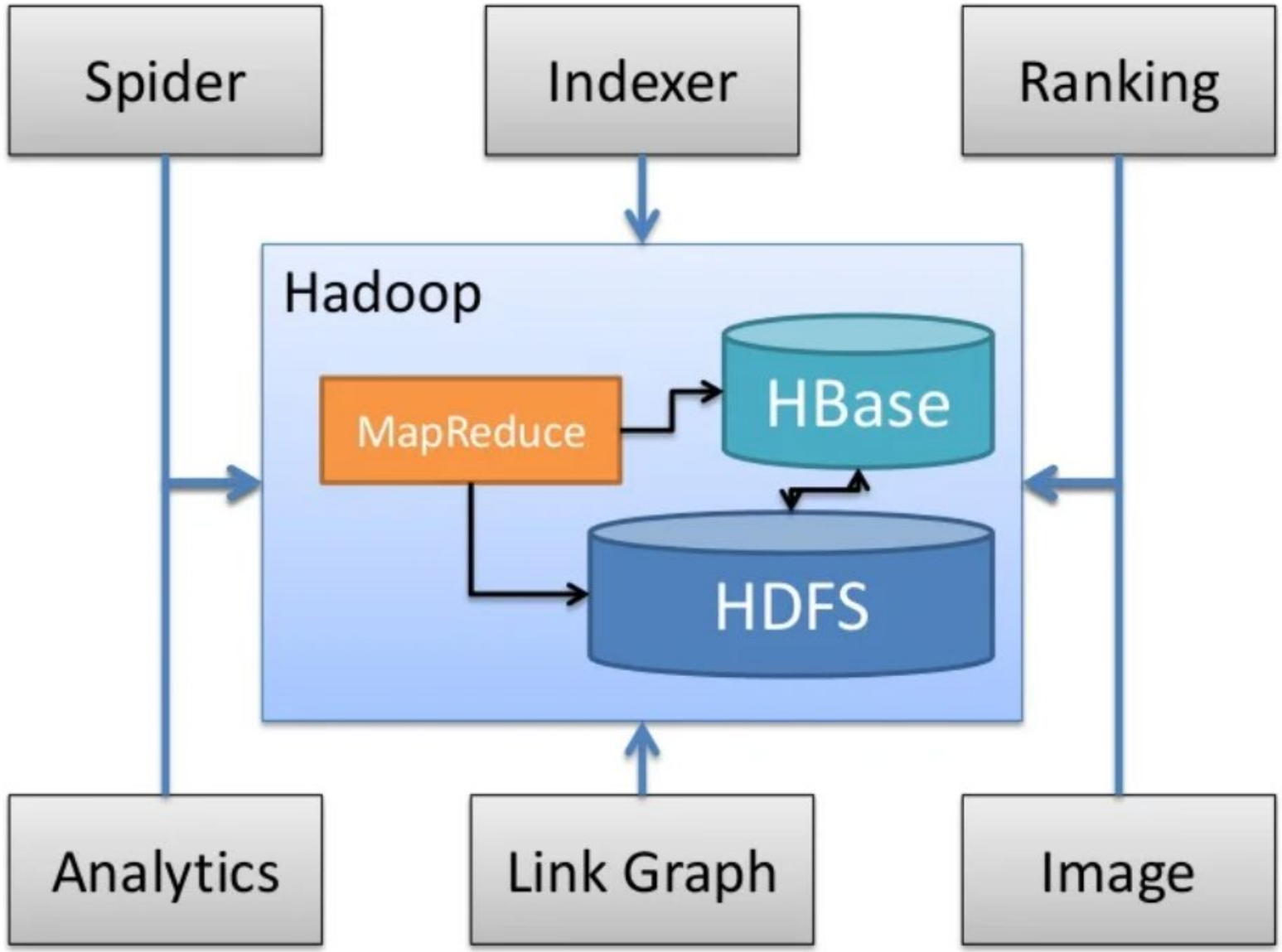
Process-as-a-Service ("управление процессом как сервис")
Представляет собой удаленный ресурс, который может связать воедино несколько ресурсов (таких как услуги или данные, содержащиеся в пределах одного "облака" или других доступных "облаков"), для создания единого бизнес-процесса.

Лекция 16. Разработка методов, алгоритмов и программных средств интеллектуального анализа больших данных: Hadoop, Spark и IoT

Hadoop

- Джентльменский набор
  - HDFS
  - MapReduce
  - HBase
  - Oozie
  - Pig
- Статистика
  - Ganglia
  - Hdpstat

Работа с Hadoop



Лекция 17. Эффективность и отказоустойчивость платформ интеллектуального анализа больших данных

ACID

- Atomicity
- Consistency
- Isolation
- Durability

ACID

- Atomicity
- Consistency
- Isolation
- Durability

ACID

- Atomicity
- Consistency
- Isolation
- Durability

ACID

- Atomicity
- Consistency
- Isolation
- Durability

ACID

- Atomicity
- Consistency
- Isolation
- Durability

Уровни изоляции

	Аномалии конкурентного доступа				
	Dirty read	Unrepeatable read	Lost updates	Phantoms	Write skew
Read uncommitted	✓	✓	✓	✓	✓
Read committed	✗	✓	✓	✓	✓
Repeatable read	✗	✗	✗	✓	✓
Snapshot isolation	✗	✗	✗	✗	✓
Serializable	✗	✗	✗	✗	✗

Strict Serializable

Serializable + Linearizable

Уровни изоляции

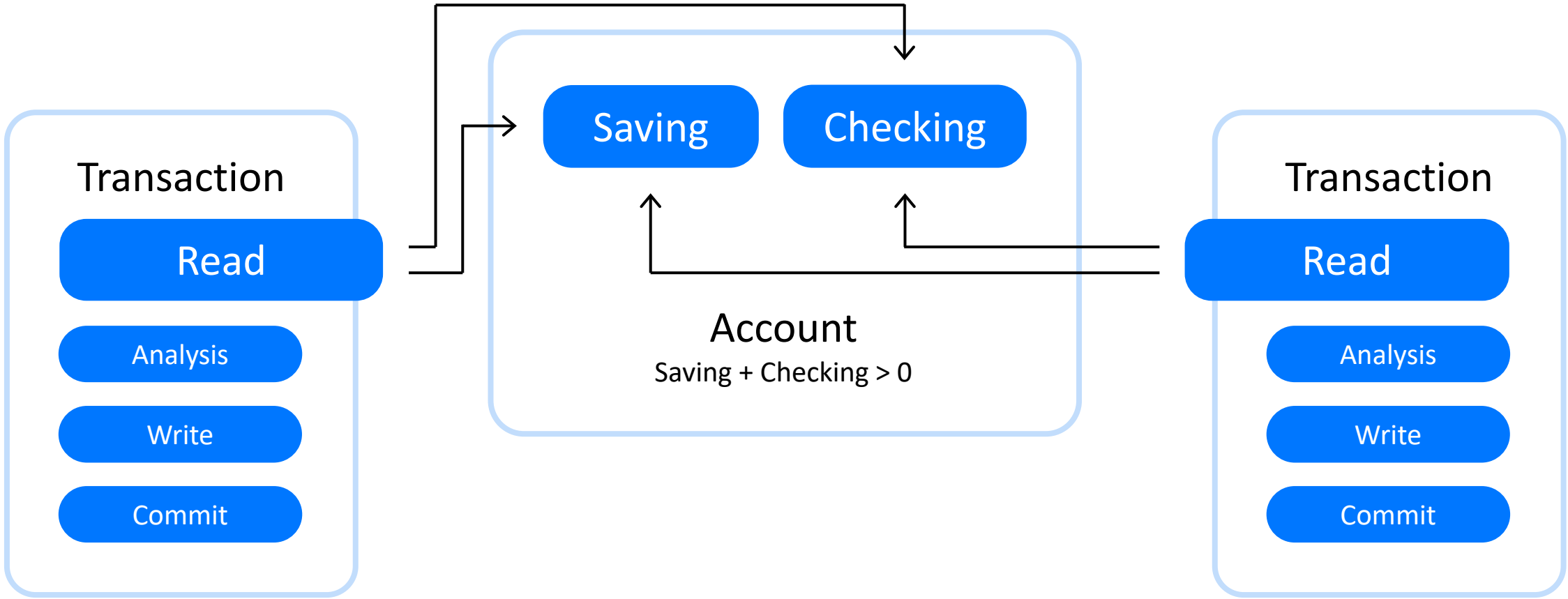
	Аномалии конкурентного доступа				
	Dirty read	Unrepeatable read	Lost updates	Phantoms	Write skew
Read uncommitted	✓	✓	✓	✓	✓
Read committed	✗	✓	✓	✓	✓
Repeatable read	✗	✗	✗	✓	✓
Snapshot isolation	✗	✗	✗	✗	✓
Serializable	✗	✗	✗	✗	✗
Strict Serializable	Serializable + Linearizable				

Уровни изоляции

	Аномалии конкурентного доступа				
	Dirty read	Unrepeatable read	Lost updates	Phantoms	Write skew
Read uncommitted	✓	✓	✓	✓	✓
Read committed	✗	✓	✓	✓	✓
Repeatable read	✗	✗	✗	✓	✓
Snapshot isolation	✗	✗	✗	✗	✓
Serializable	✗	✗	✗	✗	✗
Strict Serializable	Serializable + Linearizable				

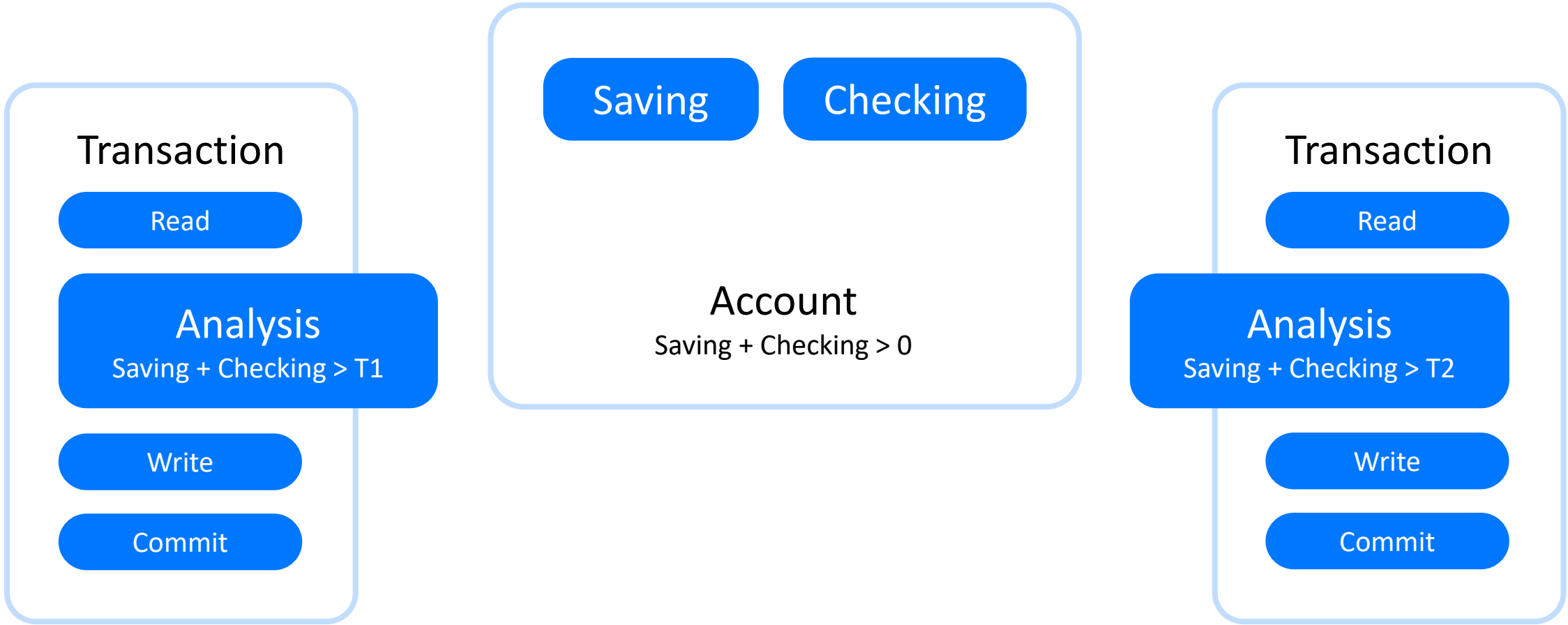
Пример аномального поведения

Write skews допустимая аномалия на уровне изоляции snapshot



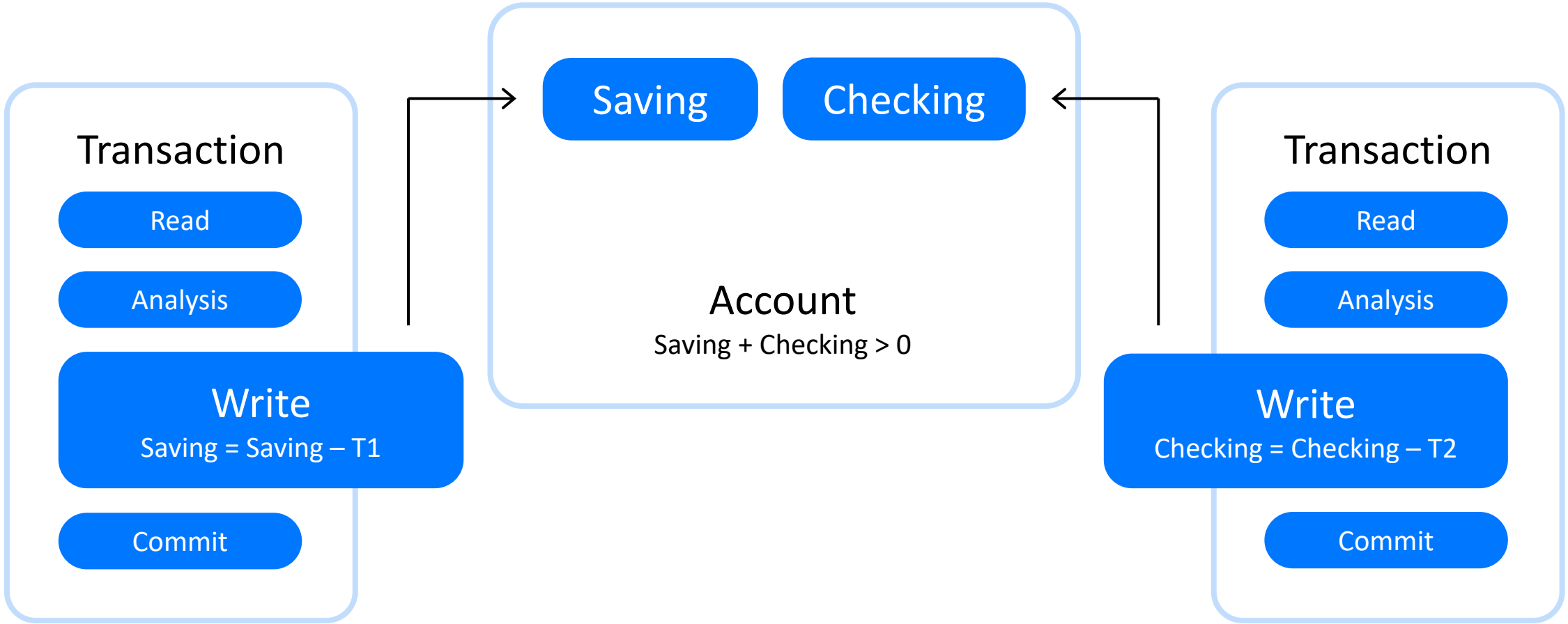
Пример аномального поведения

Write skews допустимая аномалия на уровне изоляции snapshot



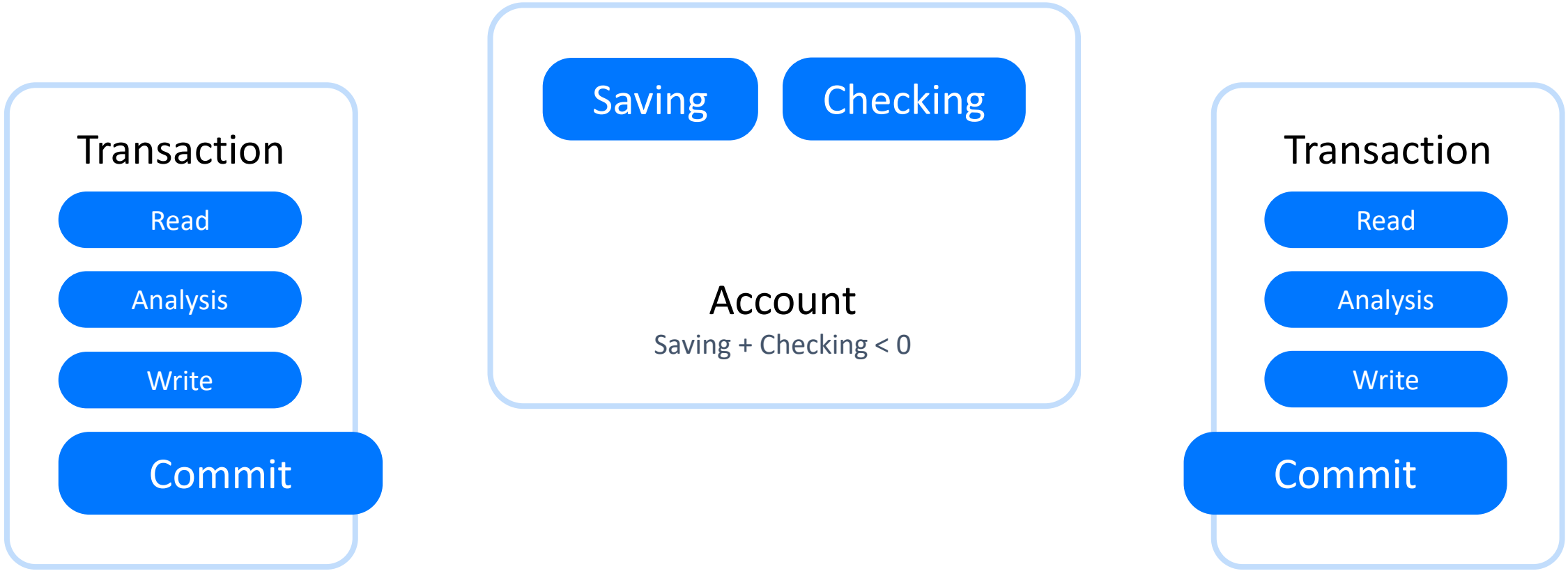
Пример аномального поведения

Write skews допустимая аномалия на уровне изоляции snapshot



Пример аномального поведения

Write skews допустимая аномалия на уровне изоляции snapshot



Пример аномального поведения

Snapshot isolation

- Обе транзакции выполняются
- Анализ данных в каждой из транзакций не покажет нарушение инварианта
- Инвариант нарушится после конкурентного выполнения транзакций

Serializable isolation

- Транзакции выполняются последовательно
- Анализ данных в каждой из транзакций будет корректным
- Инвариант не нарушится