

**MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
"IGOR SIKORSKY KYIV POLYTECHNIC INSTITUTE"
INSTITUTE OF SPECIAL COMMUNICATION
AND INFORMATION PROTECTION**

D.V. Lande, I.Yu. Subach, A.Ya. Gladun

**PROCESSING OF EXTREMELY LARGE DATA SETS
(BIG DATA)
Tutorial**

*Considered by the Academic Council of the ISZZI Igor Sikorsky KPI
for use in the training process
specialists of the second (master's) level of higher education
from specialty 122 "Computer science"*

Kyiv
ISCIP Igor Sikorsky KPI
2021

UDC 004.942 (075.8)

L 12

*Considered by the Academic Council
ISCIP KPI named after Igor Sikorsky
(Protocol No. 6 dated 12/30/2021)*

Reviewers: V.V. Vyshnivskiy, Sci.Dr., prof.
S.V. Tolyupa, Sci.Dr, prof.

L12 D.V. Lande, I.Yu. Subach, A.Ya. Gladun. Processing of extremely large data sets (Big Data): tutorial. Kyiv, 2021. – 168 p. (Translation into English using Google Translate)

ISBN 978-966-2344-83-7

The educational manual "Processing of extremely large data sets (Big Data)" includes the main provisions of the concept of Big Data, technologies for processing super-large data sets, methods of creating information systems that work with Big Data, examples of practical application of the results of Big Data processing, as well as a wide range of issues related to the practice of Big Data processing and analysis.

The guide discusses the application of intelligent data analysis methods to large volumes of data (Big Data). Relevant software tools were analyzed, including the powerful Hadoop big data processing system, the Elasticsearch search engine, the Neo4j graph database processing system, and the modern NoSQL database management system MongoDB . Examples of programs in the Python language are given in the manual to consolidate the educational material.

The publication is intended for cadets of the institute who are studying at the second (master's) level in the specialty 122 "Computer Science". It can also be useful for everyone who wants to learn the methods and technologies of *Big Data* processing on their own - cadets, students, graduate students, specialists in technical specialties and programmers.

UDC 004.942 (075.8)

ISBN 978-966-2344-83-7

©ISZZI Igor Sikorskyi KPI, 2021

CONTENT

Introduction	8
1. Formation of a data array for analytical processing	11
Data format	12
RSS feeds	15
Data loading procedures	15
Questions for practical work:	18
Tasks for independent work	19
Literature	19
2. Forming a JSON file for uploading to Elasticsearch	20
JSON format.....	20
Program to convert RSS to JSON	24
Questions for practical work:	26
Tasks for independent work	26
Literature	26
3. Installing the Elasticsearch system. Downloading information to the database	27
Basic properties of Elasticsearch	28
Installing Elasticsearch	31
Loading the Elasticsearch database	32
Checking the results of data loading	34
CRUD operations	36
Questions for practical work	36
Tasks for independent work	36
Literature	37
4. Queries to Elasticsearch and the web interface	38
Queries to Elasticsearch	38
Model of the search system based on the web interface	42

JSON file format program	44
Questions for practical work:	46
Tasks for independent work	46
Literature	46
5. Working with the Kibana system	47
Installing Kibana	47
Working with the Console tool	48
Working with the Discover tool	52
Working with the Visualizations tool	53
Questions for practical work	55
Tasks for independent work	55
Literature	55
6. Aggregation in Elasticsearch	56
Implementation of aggregation in requests	56
Dynamics of publications on request	57
Transferring data to CSV format	59
Data processing in the Excel environment	60
Questions for practical work	61
Tasks for independent work	61
Literature	62
7. Statistical processing of aggregated data	63
Development environment	63
Window smoothing	64
Exponential smoothing	66
Construction of a wavelet-scalogram	68
Questions for practical work	72
Tasks for independent work	73
Literature	73
8. Extraction of keywords from documents selected on request	74

Receiving an array of documents	74
Vocabulary formation	75
Selection of the most important words	77
Questions for practical work	78
Tasks for independent work	79
Literature	79
9. Formation of a network of concepts. Visualization in Gephi	80
Formation of the concept adjacency matrix	80
Basic information about the Gephi system	82
Installing the Gephi system	84
Network visualization in the Gephi environment	85
Questions for practical work	86
Tasks for independent work	87
Literature	87
10. Apache Hadoop	88
Core Components of Hadoop	89
Apache Hadoop and the Hadoop Ecosystem	90
Installing Hadoop on a cluster using Cloudera Manager	90
Deploying a distributed environment based on Apache Hadoop	91
Procedure for performing work	93
Installing Hadoop Offline	96
Questions for practical work.	97
Tasks for independent work	97
Literature	98
11. Programming MapReduce	99
Programming the WordCount problem	99
Map only job	104
Chains of MapReduce tasks	105
Distributed cache	106

Reduce Join	106
MapJoin	108
Questions for practical work:	110
Tasks for independent work	110
Literature	111
12. DBMS MongoDB	112
Installation of the server and line client	113
Some line client commands	113
Importing a database from a table in an external format	116
Database export to a table in an external format	117
Questions for practical work	117
Tasks for independent work	118
Literature	118
13. Compass MongoDB database client	119
Installing the Compass GUI	119
Running the MongoDB Compass GUI.....	120
Implementing CRUD commands	121
Questions for practical work	124
Tasks for independent work	124
Literature	124
14. Basics of working with Neo4j	125
Installing Neo4j	126
Starting the system	129
Creation of graph	131
Import data from .csv format into neo4j graph	134
Addition. A minimum of Python	137
Data types	137
Datetime module	152
Sys module	154

NumPy module	158
String module	159
Matplotlib: Scientific Graphics in Python	160
PyWavelets : Wavelet transforms in Python	165
Subject index	166

Introduction

Currently, the concept of Big Data plays an increasingly important role in the field of cyber security. Specialists engaged in processing, aggregating large volumes of data, solving problems caused by their growth, dynamics, and variability are currently called Data Scientists, respectively, science - Data Science.

Big data is a term that characterizes a set of data so voluminous and complex that it makes it impossible to use existing traditional database management tools and applications for their processing. The problem is the collection, cleaning, storage, search, access, transfer, analysis and visualization of such sets as a whole entity, and not as local fragments. As defining characteristics for big data, the "three V" are noted: volume (Volume, in the sense of the size of the physical volume), speed (Velocity, which means in this context the speed of growth and the need for high-speed processing and obtaining results), variety (Variety, in the sense the possibility of simultaneous processing of various types of structured and semi-structured data).

The term Big Data first appeared in an editorial by Clifford Lynch, editor of the journal Nature, on September 3, 2008, who devoted an entire special issue of that journal to the topic "what big data sets can mean for modern science."

The development of Big Data is related to the development of social networks, despite the fact that large volumes of data are also inherent in such industries as telecommunications, energy, transport, etc. And one of the first information technologies in the field of security and defense related to this problem is OSINT (Open Source Intelligence), which is a component of cyber security.

At this time, there are a number of publications related to the role of Data Science, the need to train specialists in this direction, and at the same time, a comprehensive training course related to a real practical base, which is built on modern free software, has not yet existed.

As part of the course "Processing of extremely large data arrays (Big Data)", the basic, most common today's technologies and tools in the field of cyber security are considered, the list of which allows you to get a complete picture of what specialists in the field of Data Science use today and about the tools that must be mastered. to conduct projects using Big Data.

Within the framework of the academic discipline, the following are considered:

- the possibilities of technologies for the analysis of extremely large data sets to solve cyber security problems, as well as the possibilities of applying scientific methods, including methods of intelligent data analysis, to Big Data;
- features of architectural solutions in the creation and deployment of Big Data processing systems, as well as the choice of big data storage and processing technology, the use of modern high-performance big data storage and processing systems;
- main technologies and tools for working with big data: Elastic Stack, Elasticsearch, Kibana, Hadoop, MapReduce, Neo4j, MongoDB;
- software components necessary to work in distributed information systems for processing Big Data.

As the main means of aggregation of large volumes of unstructured data, their search, visualization processing, the Elastic Stack component ecosystem is considered in this course. Elasticsearch is an information and search system, the core of the Elastic Stack, which allows processing of unstructured data, information search, data analysis, provides support for user-defined libraries and REST APIs; easy management and scaling. The Kibana utility is a window in the Elastic Stack, a tool for manipulation, analysis and visualization of information that implements such types of data display from Elasticsearch as histograms, maps, line graphs, time series.

The manual also contains information on technological platforms for Big Data processing, including the Apache Hadoop system designed for the organization of distributed processing of large volumes of data, MapReduce - a technology for distributed parallel processing of large data sets using a large number of computing clusters.

Means of analysis of large networks, graph DBMS are considered separately. The capabilities of two main systems are considered - Gephi graph analysis and visualization programs and Neo4j graph database management system. Among the features of the Gephi program, the user interface, graph composition capabilities, filtering, data exploration, visualization, and support for graphical data formats are studied. Graph DBMS Neo4j provides storage and processing of large volumes of network data, contains a declarative query language for Cypher graphs.

Thus, this manual substantiates and presents the main provisions of the training course "Processing of extremely large data sets" as an introduction to the specialty of Data Science in the field of cyber security, based on the study of the theoretical and technological foundations of this specialty, the practical application of relevant information technologies for the aggregation of large volumes of data.

1. Formation of a data array for analytical processing

The purpose of the practical work corresponding to the section is research and collection of Big Data in defined formats from social networks (RSS-feeds), saving and preparing them for further analytical processing.

The development of the direction of processing extremely large data sets (Big Data) is related to the development of the information component of the Internet, namely, web resources and social networks. This kind of content is considered as a kind of testing ground for conducting practical work within the framework of the course being studied. For further intellectual processing of such information resources, it is necessary at the first stage to create a system for its collection, but solving this interesting problem goes beyond the scope of the educational course.

That is why, for the initial collection of data for its further intelligent processing, we will limit ourselves to only one type of unstructured text information, which is freely available on the Internet and is, perhaps, one of the most standardized, namely, information provided in the RSS format.

At the end of the 20th century, several data description formats based on the XML standard were created to solve the problem of unifying the content of the Internet for the purpose of its further processing by software applications, generalization and further distribution (syndication). The most common format is called RSS, which stands for Really Simple Syndication, Rich Site Summary, although it was originally called RDF Site Summary. The content of all these abbreviations is a simple way of summarizing and distributing the information content of websites - content syndication.

The development of RSS began in 1997. This technology gained recognition when Netscape used it to populate the channels of its Netcenter portal. Soon, RSS technology was used to broadcast content on many news sites - including the BBC, CNET, CNN, Disney, Forbes, Wired, Red Herring, Slashdot, ZDNet and many

others. The first open official version of RSS was version 0.90. The format was based on RDF (Resource Description Framework).

Today, almost all leading websites, blogs, and some social networks operating on the Internet use RSS as a tool to quickly present their updates. Specifications for individual versions of the RSS format are listed on the following web pages:

RSS 0.90: <http://www.purplepages.ie/RSS/netscape/rss0.90.html>

RSS 0.91: <http://my.netscape.com/publish/formats/rss-spec-0.91.html>

RSS 1.0: <http://web.resource.org/rss/1.0/>

RSS 2.0: <http://backend.userland.com/rss/>

The term "RSS" (Really Simple Syndication) is often used as a general name for all web feeds, including those in a format other than RSS (for example, for feeds in the Atom format).

Data format

RSS is based on the XML standard. Below are the basics about RSS 2.0. The first tag in the RSS document necessarily indicates the XML format used. It is followed by the <rss> tag with the mandatory version attribute, which indicates the version of the document.

The RSS document must contain 2 tags: <channel> and <item>.

The <channel> tag contains basic information about this RSS feed. It occurs only 1 time.

It must contain the following three tags:

<title> – channel name. May match the site name.

<link> – Link to the site with which the channel is associated.

<description> – channel description.

Optional tags:

<language> – the language of the tape.

<copyright> – copyright.

<managingEditor> – email of the content editor of the channel

<webMaster> – e-mail of the webmaster of the channel

<pubDate> – channel publication date.

<lastBuildDate> – the date of the last change in the content of the channel.

<category> – channel content categories.

<generator> – the program with which the channel was generated.

<docs> – a link to the documentation of the used RSS format.

<ttl> – time of channel relevance in minutes.

<image> – the image that is displayed with the channel. In turn, the tag has the following parameters:

<Title> – title.

<Description> – description (an analogue of the ALT tag in HTML).

<Link> – a link to the site with which the channel is connected.

<URL> – image address.

<Width> – image width.

<Height> – image height.

<skipHours> – how many hours not to request updates from the channel

<skipDays> – how many days not to request updates from the channel

The <item> tag contains information about the publication.

Required nested tags:

<title> – title of the publication.

<link> – link to the page with the full text of the publication.

<description> – short text of the publication.

Optional nested tags:

<Author> – e-mail of the author

<Category> – message category

<Comments> – link to the page with comments to the message


<Enclosure> – attached multimedia object. Its parameters:

- <URL> – object address
- <Length> – object size in bytes
- <Type> – MIME file type
- <Guid> – message identifier
- <PubDate> – publication date.

Example:

```
<?xml version="1.0" encoding="windows-1251" ?>
<rss version="2.0"><channel><title>My
news</title><link>http://site.ua< /link><description>Description of my
news</description><image><url>http://site.ua
/moiLogotip.gif</url><link>http://site.ua </link><title >My
news</title></image><lastBuildDate>26 feb 2021 11:11:11
+0300</lastBuildDate><item><title>A new publication has
appeared!</title><link>http:/ /site.ua /?str=news</link><description>A
new article on "big data" appeared today</description><pubDate>26 feb
2021 11:11:11 +0300</pubDate><guid > http://site.ua
/?str=news</guid></item></channel></rss>
```

RSS feeds

Arrays of documents provided on web server resources in RSS format are called "RSS-feeds" (from the English Feed - feed, supply), information channels. The addresses of these feeds are specified explicitly on the pages of websites (the standard designation is ) , or they can be found in the source HTML code of the pages, for example, as in the fragment of the first page of the website of the UNIAN news agency:

```
<a class =" footer-menu__icon " target =" _blank " rel ="
noopener " aria-label =" rss " href
="https://rss.unian.net/site/news_rus.rss">
<i class =" union-rss "></i></a>
```

Procedures for downloading data

To download RSS feeds automatically, you can usually use standard Internet content download utilities, of which there are many, such as cURL or wget.

cURL is a website crawler that provides the ability to manipulate files on the web server side using parameters that can be passed in the URL string. cURL can be used to retrieve web pages without using a browser. In addition to HTTP requests, cURL supports SMTP, IMAP, Telnet, FTP and other network protocols. The basic use of cURL is to simply type in the shell the command cURL followed by the download URL. By default, cURL displays the output received in the standard output stream of the system, that is, calling cURL will display the program code of the page in the terminal window.

The cURL program can write the output to a file when using the "-o" option:

```
curl -o example.html www.example.com
```

Such a call ensures that the code of the title page of www.example.com is saved in the `example.html` file.

Wget is a console utility for downloading files using HTTP, HTTPS, and FTP protocols. wget lets you recursively download network files, copy both individual pages and entire websites, convert links, and more. This utility is ported and runs on many UNIX-like systems, Microsoft Windows, MacOS X, etc. wget is a non-interactive program, that is, after it is launched with certain parameters, it performs all the necessary actions and does not require additional intervention in its work. Wget can work as a crawler, that is, get resources that are referenced by HTML page elements, and recursively move through the web tree until all the necessary files are downloaded.

Procedures for downloading RSS feeds (feeds) are as follows:

```
curl -o habr https://habr.com/ru/rss/all/all/
```

or

```
wget -O habr https://habr.com/ru/rss/all/all/
```

Below is a list of RSS feeds that can be used to generate a dataset for further practical work.

```
http://economictimes.indiatimes.com/tech/software/rssfeeds/13357555.cms
http://nypost.com/tech/feed
http://www.smh.com.au/rss/headlines/technology-news/article/rss.xml
http://zeenews.india.com/rss/science-technology-news.xml
http://www.washingtontimes.com/rss/headlines/culture/technology/
https://www.thehindu.com/sci-tech/technology/?service=rss
https://www.economist.com/science-and-technology/rss.xml
https://feeds.skynews.com/feeds/rss/technology.xml
http://www.innertemplelibrary.com/category/secure-hospitals/feed/
http://e-news.com.ua/export/news.xml
http://fakty.ua/rss_feed/science
```

An example of a feed:

```
<rss version="2.0">
<channel>
<title>Science</title>
<link>https://fakty.ua/rss_feed/science</link>
<description>Science and technology news</description>
<language>en</language>
<image>
<link>https://fakty.ua/images/</link>
<url>https://fakty.ua/images/logo_uk.png</url>
<title>Science</title>
</image>
<item>
<title>
<![CDATA[
The "Action" application has a new function that will help Ukrainians
in the fight against fraudsters
]]>
</title>
<link>
<![CDATA[
https://science.fakty.ua/393252-v-prilozhenii-diya-poyavilas-novaya-
funkciya-kotoraya-pomozhet-ukraincam-v-borbe-s-moshennikami
]]>
</link>
<description>
<![CDATA[
```



```

It works by default, there is no need to update "Action".
]]>
</description>
<category>
<![CDATA[ Science and Technology ]]>
</category>
<guid>
https://science.fakty.ua/393252-v-prilozhenii-diya-poyavilas-novaya-
funktsiya-kotoraya-pomozhet-ukraincam-v-borbe-s-moshennikami
</guid>
<pubDate>Tue, 04 Jan 2022 22:30:00 +0200</pubDate>
</item>
<item>
<title>
<![CDATA[
Spoilers, reactions, QR codes: Telegram has released a New Year's
update
]]>
</title>
<link>
<![CDATA[
https://science.fakty.ua/392950-spojtery-reakcii-qr-kody-telegram-
vypustil-prednovogodnee-obnovlenie
]]>
</link>
<description>
<![CDATA[ New useful functions have appeared ]]>
</description>
<category>
<![CDATA[ Science and Technology ]]>
</category>
<guid>
https://science.fakty.ua/392950-spojtery-reakcii-qr-kody-telegram-
vypustil-prednovogodnee-obnovlenie
</guid>
<pubDate>Thu, 30 Dec 2021 16:58:00 +0200</pubDate>
</item>
>Tue, 28 Dec 2021 16:47:00 +0200</pubDate>
</item>
<item>
<title>
<![CDATA[
Messages and comments unavailable: Telegram has experienced a massive
outage
]]>
</title>
<link>
<![CDATA[
https://science.fakty.ua/392723-nedostupny-soobsheniya-i-komentarii-
v-telegram-proizoshel-masshtabnyj-sboj
]]>
</link>
<description>
<![CDATA[ Users noticed a crash on the evening of December 27th ]]>
</description>

```

```

<category>
<![CDATA[ Science and Technology ]]>
</category>
<guid>
https://science.fakty.ua/392723-nedostupny-soobcsHENiya-i-kommentarii-
v-telegram-proizoshel-masshtabnyj-sboj
</guid>
<pubDate>Mon, 27 Dec 2021 20:10:00 +0200</pubDate>
</item>
<item>
...
</channel>
</rss>

```

cURL or wget utilities can be called on a schedule, for example, using crontab tools. At the same time, in the subsequent stage of processing downloaded files, it is necessary to provide for the prohibition of taking into account documents that have the same addresses in the network (the <link> tag in the RSS feed).

Questions for practical work:

1. What is RSS and how is it used?
2. What is an RSS feed?
3. What does RSS consist of?
4. What is the difference between web feeds and RSS?

Tasks for independent work

1. Expand the list of RSS feeds with computer and telecommunications oriented channels (but not trading platforms).
2. Create a procedure (script) for periodically downloading information from the created list of RSS feeds.
3. Implement the procedure for creating a file that combines all downloaded RSS feeds and connect it to the download script.

literature

1. Dave Johnson. RSS and Atom in action: web 2.0 building blocks. - Manning Publications, 2006. - 398 pp. ISBN 9781932394498, 1932394494.

2. Michael Schrenk. Webbots, Spiders, and Screen Scrapers: A Guide to Developing Internet Agents with PHP/CURL. - No Starch Press, 2007. - 328 pp. ISBN 9781593271206, 1593271204.
3. <https://www.gnu.org/software/wget/>

2. Forming a JSON file for uploading to Elasticsearch

The purpose of the practical work corresponding to the chapter is to study and transform RSS-formats of Big Data into standard JSON text files for their further uploading to the Elasticsearch search engine for indexing and searching of any type of documents .

In the previous section, the information basis of the following practical classes was formed, files on the subject determined by the information sources, uploaded in the RSS format, which can be considered an exchange, communication format.

At the same time, for further work with modern search engines, the received data must be converted to the input formats of such systems. In particular, for further work with the Elasticsearch system, the data to be loaded must be provided in JSON format.

JSONformat

JSON (JavaScript Object Notation) is a text format for exchanging data between computers. JSON is text-based, human-readable. A format allows you to describe objects and other data structures. This format is mainly used to transmit structured information over the network.

JSON appeared due to the need to exchange data with the server in real time without using browser plugins, flash applications or Java applets. Due to its brevity compared to XML, the JSON format is more suitable for representing complex structures.

JSON is built on two structures:

- A set of name/value pairs. In various programming languages, this is implemented as an object, record, structure, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In many languages, this is implemented as an array, vector, list, or sequence.

The data structures used by JSON are supported by any modern programming language, which allows using JSON to exchange data between different programming languages and software systems.

As values in JSON can be used:

- a record is an unordered set of key-value pairs enclosed in curly braces { }. The key is described by a string, between it and the value there is a symbol ":". Key-value pairs are separated from each other by commas.
- an array (one-dimensional) is an ordered set of values. The array is written in square brackets "[]". Values are separated by commas. The array can be empty, i.e. not contain any value. The values of no more than one array can have different types.
- number (integer or real).
- literals true (logical value "true"), false (logical value "false") and null.
- string is an ordered set of zero or more Unicode characters enclosed in double quotes. Characters can be specified using escape sequences starting with the backslash "\"" (variants "\", \\, \/, \t, \n, \r, \f and \b are supported), or written in hexadecimal code in Unicode encoding in the form \uFFFF.

A "string" is very similar to a literal of the same data type in Javascript. "Number" is also very similar to a Javascript number, except that only decimal format is used (with a dot as a separator). Spaces can be inserted between any two syntax elements.

The following example shows a JSON representation of data about an object describing a person. The data includes string fields for first and last name, address information, and an array containing a list of telephone numbers. As you can see from the example, the value can be a nested structure.

```
{  
  "firstName": "Ivan",  
  "lastName": "Ivanov",  
  "address": {
```

```
"streetAddress": "Prov. P. Myrnogo, 10, apartment 101",
"city": "Amounts",
"postalCode": 10101
},
"phoneNumbers": [
"067 123-1234",
"050 123-4567"
]
}
```

Pay attention to the "postalCode" pair: 10101. Both numbers and strings can be used as JSON values. Therefore, the entry "postalCode": "10101" contains a string, and "postalCode": 10101 - a numeric value. Due to weak typing in Javascript and PHP, a string can be reduced to a number and not affect the logic of the program. However, it is recommended to be careful with the value type, since JSON is used for cross-system exchange.

The Elasticsearch system does not impose a clear data structure; you can store any JSON documents that are particularly suitable for Elasticsearch as opposed to rows and columns in relational databases. Such a document can be compared with a record in a database table. In traditional relational databases, tables are structured: they have a fixed number of columns, each with its own type and size. But the data can change dynamically, requiring support for new or dynamic columns. JSON documents initially support this data type.

below shows a JSON representation of an object describing a customer :

```
{
"name": "John Smith",
"address": "121 John Street, NY, 10010",
"age": 40
}
```

This is what a customer record might look like. It contains the name, address and age of the client. And another entry might look like this:

```
{
"name": "John Doe",
"age": 38,
"email": "john.doe@company.org"
}
```

Note that the second client does not have an address field, but instead has an email field. And other customers may have a completely different set of fields. So flexibility is implemented in terms of what can be stored in the table.

To upload to the Elasticsearch system, it is necessary to convert the file obtained in the previous practical work from the RSS format, namely to create the JSON format. The following simplified list of document fields, which should be uploaded to the Elasticsearch system in the future, is offered:

"title" – name of the message;

"textBody" – message text;

"source" – the source of the message;

"pubDate" – date and time in YYYYMMDD HH:MM format

"url" is the address of the message on the Internet.

Below is a fragment of the JSON file that should be retrieved:

```
{
  "title": " A new function has appeared in the "Diya" application
that will help Ukrainians in the fight against fraudsters",
  "textBody": " It works by default, you don't need to update
"Action",
  "source": " Facts",
  "PubDate": "2021-12-28T10:04:00Z",
  "URL": " https :// science . fakty . ua /393252- v - prilozhenii - diya
- poyavilas - novaya - funkiya - kotoraya - pomozhet - ukraincam - v -
borbe - s - moshennikami "
}
,
{
  "title": "Court in a smartphone: "Diya" is testing a new service",
  "textBody": "They want to send messages about court sessions
through the application",
  "source": "Facts",
  "PubDate": "2021-12-28T10:54:00Z",
  "URL": "https://science.fakty.ua/392788-sud-v-smartfone-diya-
testiruet-novyj-servis"
}
{
  "title": "James Webb May Rewrite Space History: Launch Video of
NASA's Newest Telescope",
  "textBody": "The device will begin operation in the summer of
2022",
}
```

```

"source": "Facts",
"PubDate": "2021-12-25T10:14:00Z",
"URL": " https://science.fakty.ua/392589-james-webb-mozhet-
perepisat-kosmicheskuyu-istoriyu-video-zapuska-novogo-teleskopa-
ot-nasa"
}

```

RSS to JSON conversion program

The practical task solved in this section is the development of a Python program that converts (converts) information provided in RSS format to JSON format without changing the content of this information. Below is an example of such a program (explanations are provided as comments in the program text, after the # sign):

```

#Program for data conversion from RSS format to JSON format
#Connection of modules for working with regular expressions and
time
import re
import datetime

#Opening the rss.xml file in "reading" mode
f = open("rss.xml", "r")

#Reading the contents of the rss.xml file into the variable t
t = f.read()

#Closing the rss.xml file
f.close()

# Splitting the file by lines and gluing lines using a skip
rss = t.split('\n')
t=""
for i in range(len(rss)):
t=t+" "+rss[i]

#Removing the first passes
t=re.sub('^s', '', t)

#Formation of header array
title = re.findall('<title>(.*?)</title>', t)

#The first header is the name of the feed, followed by its
specific processing

```



```

source=title[0]
source=re.sub('[\s\-*$]', '', source)
source=re.sub("'", '\'', source)

#Size of header array
x=range(len(title))

# Formation of an array of descriptions
text = re.findall('<description>(.*?)</description>', t)

# Formation of an array of hyperlinks
link = re.findall('<link>(.*?)</link>', t)

#Format of date and time in "YY-MM-MMTHH:MM:00Z" format
now = datetime.datetime.now()
tim=now.strftime("%Y-%m-%dT%H:%M:00Z")

#Output of results
for i in range(1, len(title)):
print "{\n"title\":"\""+title[i]+"\", "

#Specific text processing
text[i]=re.sub('[\s\-*$]', '', text[i])
text[i]=re.sub("'", '\'', text[i])
text[i]=re.sub('\'', '&', text[i])

#Further display of results
print "\"textBody\":"\""+text[i]+"\", "
print "\"source\":"\""+source+"\", "
print "\"PubDate\":"\""+tim+"\", "
print "\"URL\":"\", link[i], "\"\n\""
if i<len(title)-1:
print ", "

```

As a result of the execution of the given program, a JSON file should be generated, suitable for loading into the environment of the Elasticsearch information and search system.

Questions for practical work:

1. What is JSON format?
2. What are the main advantages of the JSON format?
3. What do you know about the rules for creating the structure of a JSON file in an object, an array, and when assigning a value?

4. What is Elastic search and its purpose?

Tasks for independent work

1. Write a program for creating a batch file in JSON format.
2. Install a library for working with regular expressions in the programming language environment (Python) on the computer.
3. Familiarize yourself with the basic capabilities of the Python programming language for working with temporal data.

literature

1. Fundamentals of Data Science and Big Data. Python and the science of data / Deva Silen, Arno Meisman. - Peter, 2017. - 336 p.
2. Pranav Shukla, Sharat Kumar. Elasticsearch, Kibana, Logstash and new generation search systems. - Peter, 2019. - 363 p.
3. Bontsanyny M. Analysis of social media on Python / trans. with Eng. A. V. Logunova. - M.: DMK Press, 2018. - 288 p.
4. S. Shaposhnikova. Fundamentals of Python programming. Introductory course. - Young Linuxoid Laboratory, 2011. - 44 c.

3. Installing the Elasticsearch system. Loading information into the database

The purpose of the practical work corresponding to the section is to learn how to install the Elastic search system and upload the information obtained in the previous work in JSON format to the database of the Elastic search search system for indexing and searching for documents of various types .

This section is devoted to the installation and study of the capabilities of the Elasticsearch system, an information and search system that provides the accumulation and search of large volumes of data. This system is part of the Elastic technology stack.

Elastic stack

Over the past few years, various systems for storing and processing large data sets have appeared. Among them, we can distinguish the components of the Elastic ecosystem, which are used for searching and processing data. The core components of the Elastic Stack are Kibana, Logstash, Beats, X-Pack, and Elasticsearch. The core of the Elastic Stack is the Elasticsearch search engine, which provides capabilities for storing, searching, and processing data. The Kibana utility, also known as the Elastic Stack window, is an excellent visualization and user interface for Elastic Stack. The Logstash and Beats components allow you to push data to the Elastic Stack. X-Pack provides powerful functionality: you can configure monitoring, add various messages, set security parameters to prepare your system for operation. Because Elasticsearch is the core of the Elastic Stack.

Elasticsearch is a highly scalable distributed search engine for full-text search and real-time data analysis. The utility allows you to store, search and analyze large volumes of data. Typically used as an underlying mechanism/technology to assist applications with complex search functions. Elasticsearch is a core component of the Elastic Stack.

Elasticsearch, as the heart of the Elastic Stack, plays a fundamental role in data search and analysis. It is built on a unique technology - Apache Lucene. This makes Elasticsearch fundamentally different from traditional relational database or NoSQL solutions. Below are the main advantages of using Elasticsearch as a data store:

- unstructured, document-oriented;
- the ability to search;
- possibility of data analysis;
- support for user libraries and REST API;
- easy management and scaling;
- work in pseudo-real time;
- high speed of work;
- resistance to errors and failures.

Basic properties of Elasticsearch

Elasticsearch provides data storage, search and analysis capabilities at scale.

The Elasticsearch system has a clear document orientation. JSON documents are best suited for it. They are organized using different types and indexes. Key concepts of Elasticsearch:

- index;
- type;
- document;
- cluster;
- node;
- shards and copies;
- markup and data types;
- reverse index.

An index is a container that stores and manages documents of the same type in Elasticsearch. The index can contain documents of the same type.

Indexes in Elasticsearch are roughly similar in structure to databases in relational databases. An Elasticsearch type corresponds to a table, and a document corresponds to a record in it.

Types help to logically group or organize documents of the same type by indexes.

Usually, documents with the most common set of fields grouped under one type. Elasticsearch does not require a structure, allowing any JSON document with any set of fields to be stored under a single type.

JSON documents are best suited for use in Elasticsearch. A JSON document consists of several fields and is the basic unit of information stored in Elasticsearch.

Elasticsearch is an unstructured system, thanks to which you can store documents with any number and types of fields.

Elasticsearch supports a wide set of data types for different storage scenarios of text data, numbers, booleans, binary objects, arrays, geopoints, geoforms, and many other specialized data types, such as IPv4 and IPv6 addresses.

JSON documents usually contain multiple fields. In JSON documents, each field has a specific type. Each field and its value can be seen as a key-value pair in a document, where the key is the name of the field and the value is the value of the field.

Elasticsearch is a distributed system. It covers multiple processes running on different devices in the network and interacting with other processes.

An Elasticsearch node is a single system server that can be part of a large cluster of nodes. It is involved in indexing, searching, and performing other

operations supported by Elasticsearch. Each Elasticsearch node is given a unique ID and name at startup.

Each Elasticsearch node has a main configuration file. YML file format (full name - YAML Ain't Markup Language). This file can be used to change values such as node name, ports, cluster. At a basic level, a node corresponds to a single running Elasticsearch process. It is responsible for managing the relevant part of the data. A cluster contains one or more indexes and is responsible for performing operations such as searching, indexing, and aggregation. A cluster is formed by one or several nodes, each of which is responsible for storing and managing its part of data. One cluster can store one or more indexes. The index logically groups different types of documents.

Any Elasticsearch node is always part of a cluster, even if it is a single node cluster. By default, each node tries to join a cluster named Elasticsearch. If multiple nodes are started within the same network without changing the cluster.name parameter in the config/elasticsearch.yml file, they are automatically clustered.

Shards help distribute the index across the cluster. They distribute documents from the same index from different nodes. The amount of information that can be stored in one node is limited by the disk space, RAM and computing capabilities of that node. Shards help distribute the data of one index throughout the cluster and thus optimize cluster resources. The process of dividing data into shards is called sharding. This is an integral part of Elasticsearch, necessary to optimize disk space from different cluster nodes, as well as computing power from different cluster nodes.

Each index shard can be configured to have some number of additional copies of the original or primary shard to provide high data availability.

Installing Elasticsearch

To install Elastic Stack components - Elasticsearch and later the Kibana system, you need to go to the start web page on the Internet (<https://www.elastic.co/start>, Fig. 3.1) and download the corresponding archive. Starting with Elastic Stack version 5.x, all components are updated together and have a single version number. This applies to the Elastic Stack version 7.x components.

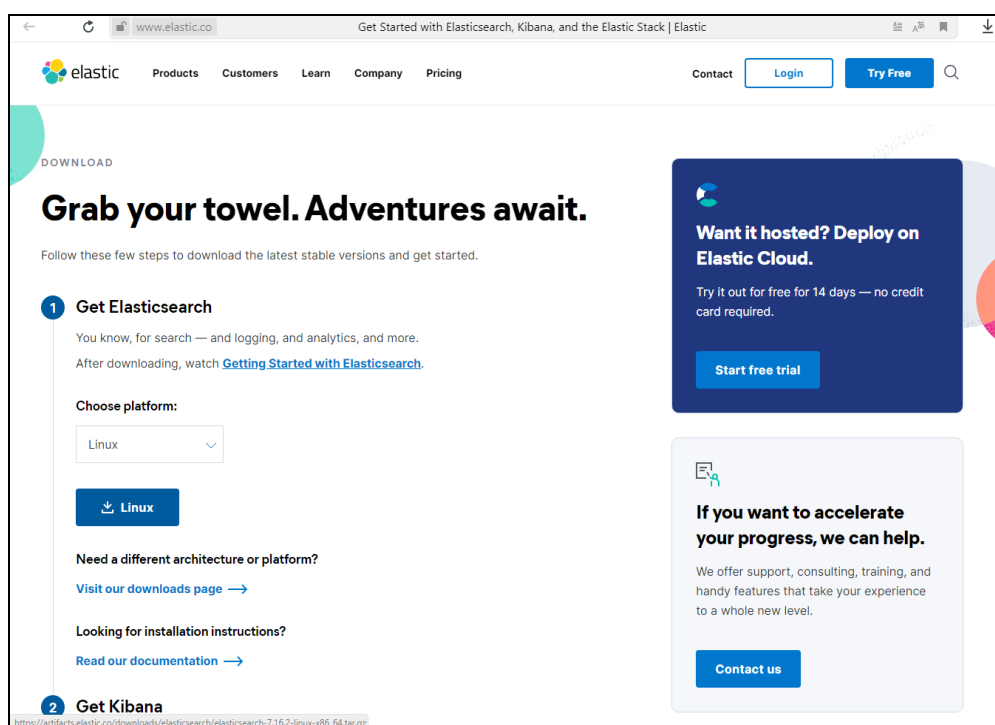


Fig. 3.1 – Start page of the system in Elasticsearch Internet

After downloading the distribution in ZIP format (Linux button), you can view its contents (Fig. 3.2).

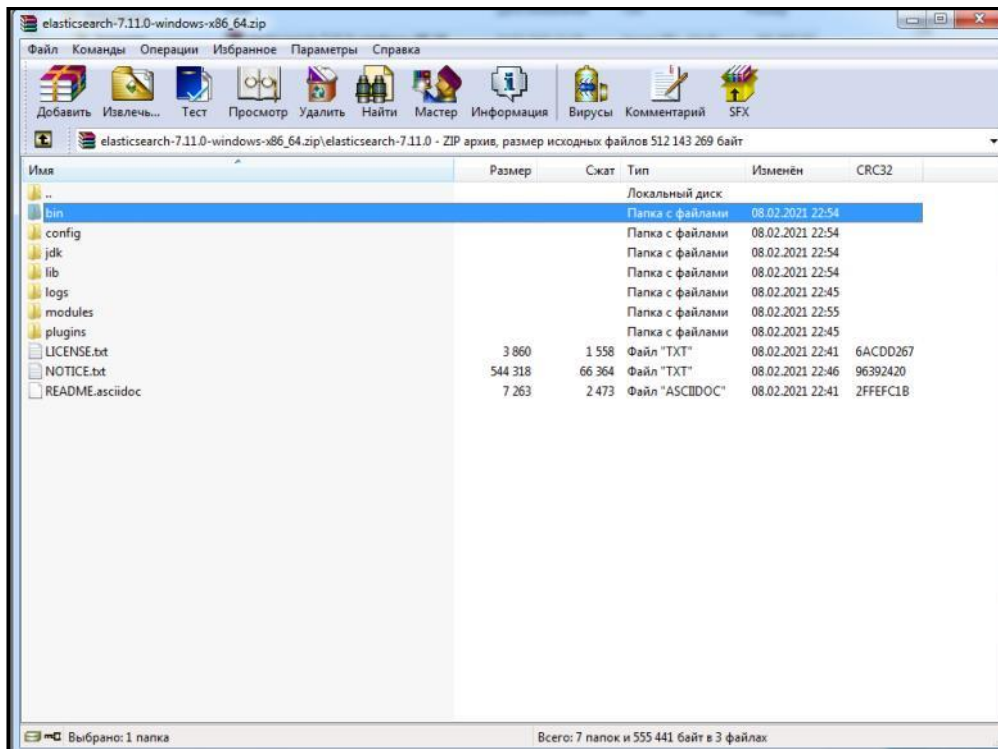


Fig. 3.2 - Contents of the system distribution in Elasticsearch

To install the Elasticsearch system on the computer (server), it is enough to unzip the files and go to the created folder. After that, you can run (depending on the operating system) files `bin/elasticsearch` or `bin/elasticsearch.bat` and the system will be installed.

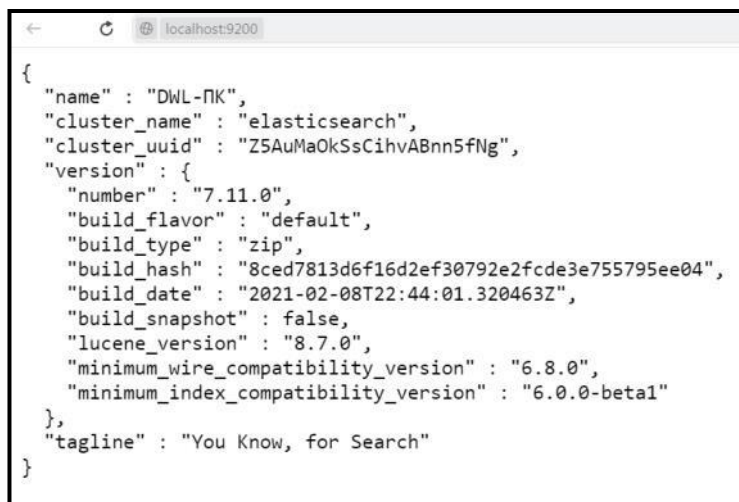
After accessing through a browser or cURL type utility at the address `curl http://localhost:9200`, you can see the response of the Elasticsearch system (Fig. 3.3).

The first task that arises after installing the Elasticsearch system is the task of creating and loading the database with JSON documents, the receipt of which is described in sections 1 and 2.

Loading the Elasticsearch database

To download a JSON file, you can use the cURL utility in XPOST data recording mode, by specifying the address of the Elasticsearch system (`http://localhost:9200`), the name of the index (`test`, set arbitrarily) and the name of

the type according to which the document should be indexed (`_doc`, is also set arbitrarily).



```
{
  "name" : "DWL-ПК",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "Z5AuMaOkSsCihvABnn5fNg",
  "version" : {
    "number" : "7.11.0",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "8ced7813d6f16d2ef30792e2fcde3e755795ee04",
    "build_date" : "2021-02-08T22:44:01.320463Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Fig. 3.3 – Access to the system at the local address `http://localhost:9200`

The file for downloading documents (differs from the JSON file by the presence of the curl command with the XPOST parameter, specifying the system address) looks like this:

```
curl -XPOST 'http:// localhost:9200/test/_doc/' -H 'Content-Type:
application/json' -d '
{
  "title":"Telegram messenger crashed in Ukraine, Instagram users also
have problems",
  "textBody":"Problems with access to social networks are experienced in
many countries",
  "source":"Facts",
  "PubDate":"2021-12-04T18:51:00Z",
  "URL":"https://science.fakty.ua/390973-v-ukraine-zasboil-messendzher-
telegram-problemy-takzhe-u-polzovatelej-instagram"
}'
curl -XPOST 'http:// localhost:9200/test/_doc/' -H 'Content-Type:
application/json' -d '
{
  "title":"Change of registration through "Action": how to participate
in testing and what is important to know",
  "textBody":"3.5 thousand volunteers take part in the project",
  "source":"Facts",
  "PubDate":"2021-12-02T10:02:00Z",
  "URL":"https://science.fakty.ua/390775-smena-propiski-cherez-diyu-kak-
prinyat-uchastie-v-nachavshem-sya-testirovanii-i-chto-vazhno-znat"
}'
```

```
curl -XPOST 'http://localhost:9200/test/_doc/' -H 'Content-Type:
application/json' -d '
{
  "title":" A mobile application that "saves" women from violence has become
available in Ukraine ",
  "textBody":" The application was developed in Israel ",
  "source":"Facts",
  "PubDate":"2021-11-11T14:59:03Z",
  "URL":" https :// science . fakty . ua /389234- v - ukraine - stalo -
available - mobilnoe - prilozhenie - spasayucshee - zhencshin - ot - nasiliya
"
}'
```

When downloading JSON documents, the Elasticsearch system issues a message (download log) of the following form:

```
{"_index":"test","_type":"_doc","_id":"JF37oXcButXogwwaYkt","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":0,"_primary_term":1}
{"_index":"test","_type":"_doc","_id":"JV37oXcButXogwwaYkvi","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1,"_primary_term":1}
. . .
{"_index":"test","_type":"_doc","_id":"Jl37oXcButXogwwaYkv6","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":9,"_primary_term":1}
```

Checking the data download results

To check the number of uploaded documents, it is enough to contact the Elasticsearch system as follows:

```
curl -XGET 'http:// localhost:9200/test/_doc/_count'
```

The result of executing the command to check the number of uploaded documents looks something like this:

```
{"count":10,"_shards":{"total":1,"successful":1,"skipped":0,"failed":0}}
```

To check the correctness of the creation of the index, a query (written in JSON format) is entered, according to which it is necessary to find a document in the body of which ("textBody" field) contains the word "service":

```
curl -XGET 'http:// localhost:9200/test/_doc/_search?pretty=true' -H
'Content-Type: application/json' -d '
{
"query" : {
"match" : { "textBody": " service " }
}
}'
```

If the database is correctly created and filled, the result will be obtained:

```
{
"took" : 0,
"timed_out" : false,
"_shards" : {
"total" : 1,
"successful" : 1,
"skipped" : 0,
"failed" : 0
},
"hits" : {
"total" : {
"value" : 1,
"relation" : "eq"
},
"max_score" : 2.3004205,
"hits" : [
{
"_index" : "test",
"_type" : "_doc",
"_id" : "LF37oXcButXogwwaY0uW",
"_score" : 2.3004205,
"_source" : {
"title" : " U new functions have appeared in the "Kyiv Digital"
application: what it is important for users to know ",
"textBody" : "Some existing services have also been improved .",
"source" : "Facts",
"PubDate" : "2021-11-08T15:40:00Z",
"URL" : " https://science.fakty.ua/388979-v-prilozhenii-kiev-cifrovoj-
poyavilis-novye-funksii-hto-vazhno-znat-polzovatelyam "
}
}
]
}
}
```

CRUD operations

CRUD - English Create, Read, Update, Delete – 4 main functions of data management "creation, reading, updating and deletion".

In Elasticsearch, CRUD operations target documents. The following APIs are implemented for CRUD operations:

- Index API;
- Get API;
- Update API;
- Delete API.

Questions for practical work

1. What are the features of the Elasticsearch search engine (as opposed to web search engines on the Internet)?
2. What is full-text search and how does it work?
3. How does full-text search differ from document search based on metadata?
4. What are CRUD operations in Elasticsearch?
5. What CRUD methods in the Elasticsearch system do you know?

Tasks for independent work

1. Write a program for creating a batch file for uploading to Elasticsearch based on the file created in the previous lesson in JSON format.
2. Install the curl utility (<http://curl.se>) on your computer and study the composition and values of its main parameters.
3. Familiarize yourself with the composition and content of CRUD operations in Elasticsearch.

literature

1. Pranav Shukla, Sharat Kumar. Elasticsearch, Kibana, Logstash and new generation search systems. - Peter, 2019. - 363 p.
2. Elasticsearch: The Definitive Guide / Clinton Gormley and Zachary Tong. - O'Reilly Media, Inc., 2015. - 719 p.
3. Elasticsearch Blueprints. A practical project-based guide to generating compelling search solutions using the dynamic and powerful features of Elasticsearch / Vineeth Mohan. - Packt Publishing, 2015. - 192 p.

4. queries and the web interface

of *the practical work* corresponding to the chapter is to learn how to form search queries in the Elastic search system and create a web interface (CGI procedures in Python) to the Elastic search database .

Queries to Elasticsearch

In the previous section, we already considered a query to the database by a term included in an unstructured text field.

In this case, the match query option was used in the JSON request. A match query is a query that is used by default in full-text search mode. This is one of the high-level queries that involves using a parser for the source field.

When we perform a match request, the following actions are expected:

1. Search for terms in all documents within a certain field (for example, "textBody").
2. Finding the best matches, sorted by score in descending order of birth frequency.

When an end user searches for any terms, sometimes it is necessary to query the two fields "textBody" (body of the document) and "title". This is possible using a search request in several fields.

The following query will find all documents that have the word "cybersecurity" in the title or description fields:

```
curl -XGET
'http://172.16.33.11:9200/_all/_search?pretty=true&size=100' -H
'Content-Type: application/json' -d '
{
  "query": {
    "bool": {
      "must": [
        {
          "multi_match": { "query": " cybersecurity",
            "fields": ["textBody", "title"]
```

```
}  
}  
]  
}  
}  
}'
```

Below is another query that hits multiple fields for the phrase “ cyber security ”. In addition, the search request is filtered by source ("CTVNews.ca"):

```
curl -XGET  
'http://172.16.33.11:9200/_all/_search?pretty=true&size=100' -H  
'Content-Type: application/json' -d '  
{  
  "query": {  
    "bool": {  
      "must": [  
        {  
          "multi_match": { "query": " cyber security ",  
            "fields": ["textBody", "title"]  
        }  
      ],  
      "filter": [  
        {  
          "match": {  
            "source": "CTVNews.ca"  
        }  
      ]  
    }  
  }  
}'
```

As a result of processing the given request, we get:

```
aggregator@cyber_aggregator_search:~/elastic$ ./query45_test_sm  
{  
  "took" : 13,  
  "timed_out" : false,  
  "_shards" : {  
    "total" : 113,  
    "successful" : 113,  
    "skipped" : 0,  
    "failed" : 0  
  },  
  "hits" : {  
    "total" : {
```

```

"value" : 2413,
"relation" : "eq"
},
"max_score" : 21.18586,
"hits" : [
{
  "_index" : "hb",
  "_type" : "_doc",
  "_id" : "6piHNnUBmqEEOXSaGYZ0",
  "_score" : 21.18586,
  "_source" : {
    "id" : "20191111102600.62_hb",

    "title" : " Nearly half of Canadians lack confidence
in cybersecurity of CRA, Elections Canada: survey ",
    "textBody" : "Thursday, December 9, 2021 10:20PM EST (Soumil
Kumar / Pexels.com) Share: Reddit Share Text: According to a new
survey, nearly half of Canadians lack confidence in the
cybersecurity of Elections Canada and the federal government
services such as the Canada Revenue Agency. More than 100
ransomware attacks targeted notable Canadian sites in 2021,
including hospitals and Rideau Hall, and in the wake of this,
Canadians are feeling a lack of confidence in institutions'
ability to protect from cyber threats, according to a report
released by the Angus Reid Institute on Thursday. The report
detailed the results of a survey that found that more than half
of Canadians said they were "not confident" that their municipal
government had good cybersecurity, and half were not confident
that their local health authority had good cybersecurity.

. . .

This survey comes after federal ministers urged Canadians to
bolster their cybersecurity earlier this week, stating in an open
letter that Canadians should build a response plan. METHODOLOGY
The Angus Reid Institute conducted an online survey from Nov. 3-
7, 2021, among a representative randomized sample of 1,611
Canadian adults who are members of the Angus Reid Forum. For
comparison purposes only, a probability sample of this size would
carry a margin of error of plus or minus 2.5 percentage points,
19 times out of 20.."
    "source" : "CTVNews.ca",
    "user" : "Cloud4Y",
    "pubDate" : "20 21 - 12 - 10 T 07 : 09 :00Z",
    "URL" : " https://www.ctvnews.ca/canada/nearly-half-of-canadians-
lack-confidence-in-cybersecurity-of-cra-elections-canada-survey-
1.5701735"
  }
},
{

```



```

"_index" : "hb",
"_type" : "_doc",
"_id" : "oJmYNnUBmqEEOXSaKzGy",
"_score" : 17.96392,
"_source" : {
  "id" : "20200507122600.104_hb",
  "title" : "NL health officials confirm cyberattack causing health
system interruptions",
  "textBody" : "Newfoundland and Labrador's minister of health
confirmed Wednesday the IT problems causing disruptions to the
province's health-care system are a result of a cyberattack. "At
this stage of our investigation, we can confirm we've been the
victim of a cyberattack that has impacted our health-care
systems," John Haggie said during a news conference. "We have
engaged cyber security experts to help us investigate and resolve
and we've informed the appropriate authorities." Haggie said the
cyberattack was detected on Saturday, and teams are working
around the clock to deal with the situation, but he said
information was limited at this time. "Those involved in the
attack may actually be monitoring what we are saying in the media
and on the floor of the house. It is very important that we don't
do or say anything that compromised the efforts to investigate
and resolve this matter," said Haggie. The province has activated
its provincial emergency operations center, and each of the
Regional Health Authorities are in a 'Code Grey' situation.
Haggie said the worst system impacts are in the Western Health
zone, and that system has been taken offline as of Wednesday
morning. Eastern Health and Central Health also say they expect
to continue canceling thousands of non -emergency procedures
Wednesday and Thursday. With files from the Canadian Press.
Atlantic Top Stories Nova Scotia issues eight-week grace period
for its vaccine mandate in public sector Thousands attend rally
outside NB legislature in support of striking CUPE workers NL
health officials confirm cyberattack causing health system
interruptions Nova Scotia reports 11 new COVID-19 cases Tuesday,
active cases drop to 161 NB reports one death, 40 new cases, and
75 recoveries Tuesday as active cases drop to 470 NS",
  "source" : "CTVNews.ca",
  "user" : "PhoenixContact",
  "pubDate" : "2020-05-07T12:26:00Z",
  "URL" :
  "https://habr.com/ru/post/500864/?utm_source=habrahabr&utm_
m_medium=rss&utm_campaign=500864"
}
},
. . .
}

```

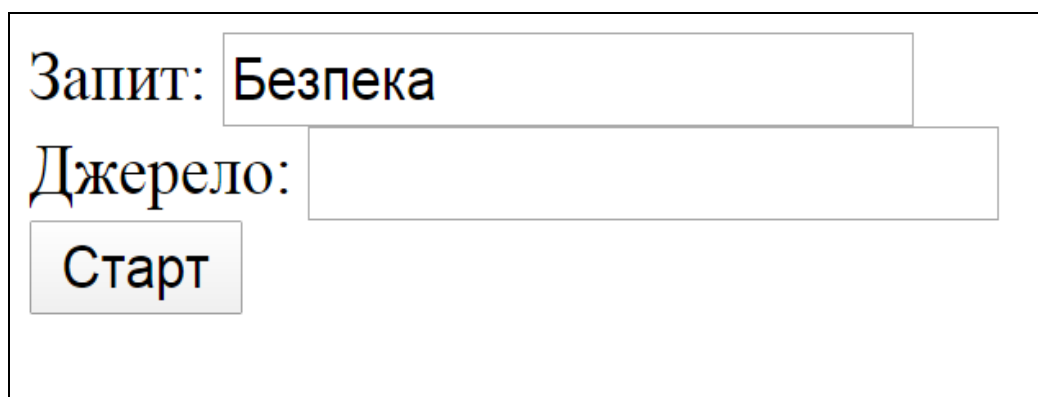
A model of a search engine based on a web interface

Elasticsearch can be used as a search engine for search engines installed on CGI-enabled web servers.

CGI (from the English Common Gateway Interface) is an interface standard used for communication of an external program with a web server. A program that works with such an interface together with a web server is called a gateway or a "CGI script".

The interface itself is designed so that any programming language that can work with standard I/O devices can be used. Even scripts for built-in command interpreters of operating systems have such capabilities.

Suppose that using the CGI mechanism, it is necessary to implement the web interface shown in Fig. 4.1.



Запит:

Джерело:

Fig. 4.1 - Web interface for entering a request

Below is an HTML form (CGI form) of the search engine model:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>RSS Aggregator</title>
</head>

<body bgcolor="white">
```

```

<table border=0 width=100%><tr>
<td width=170>

<form action="/cgi-bin/test_json.py" target="content">

Query: <input type="text" name="query" value="">
<br>Source: <input type="text" name="src" value="">
<br><input type="submit" name="act" value="Start">
</form>
</body>
</html>

```

The given CGI form contains a call to a Python CGI procedure (test_json.py) that accepts parameters and calls a query to the Elasticsearch system:

```

#!/usr/local/bin/python2.7
import os
import re
import cgi
import subprocess

form = cgi.FieldStorage()
query = form.getfirst("query", "Query not defined")
src = form.getfirst("src", "Source not defined")
os.system('/usr/local/www/apache24/cgi-bin/test_json.sh' %s %s'
% (query, src))

```

A shell script that provides the transmission of a request to the Elasticsearch system (test_json.sh) and the subsequent call of the response formatting program (prog2_html):

```

#!/bin/sh
query=$1
src=$2
/usr/local/bin/curl -XGET
'http://172.16.33.11:9200/_all/_search?pretty=true&size=20' -H
'Content-Type: application/json' -d '
{
"query": {
"bool": {
"must": [
{
"multi_match": { "query": "'$query'",

```

```

    "fields": ["textBody", "title"]
  }
}
],
"filter": [
{
  "match": {
    "source": "'$src'"
  }
}
]
}
}
}' | /usr/local/www/apache24/cgi-bin/prog2_html

```

Program for formatting a JSON file

As a result of processing the user's request, the Elasticsearch system provides results in the form of a JSON file. Below is the code of the program module in the Python language (prog2_html), which, based on the received JSON file, forms an HTML file that provides the output of results in the form presented in Fig. 4.2.

The screenshot shows a yellow background with three search results. Each result has a title and a description. The first result is about Ukrainian sensations and security. The second is about protecting oneself from hacker attacks. The third is about the start of an exhibition in Kyiv.

3. : **Українські сенсації. Увага! Вас слухають**
 Українські сенсації. Увага! Вас слухають українські сенсації, прослуховування, інформаційна безпека, інформаційна гігієна, україна політика, соцмережі безпека, особиста інформація соцмережі безпека,...
https://www.youtube.com/watch?v=7r_oqoirV38

4. : **Як убезпечити себе від атаки хакерів – поради експерта з інформаційної безпеки**
 Як убезпечити себе від атаки хакерів – поради експерта з інформаційної безпеки українські сенсації, прослуховування, інформаційна безпека, інформаційна гігієна, україна політика, соцмережі безпека, особиста інформація соцмережі безпека,...
<https://www.youtube.com/watch?v=DWwOZq4aJ9U>

5. : **У Києві стартували «Зброя та Безпека – 2021» і «Авіасвіт – XXI»**
 15 червня 2021 року у киеві розпочалися xVii міжнародна спеціалізована виставка «зброя та безпека – 2021» та xii міжнародний авіакосмічний салон «авіасвіт – xxi», які традиційно проводяться в міжнародному виставковому центрі на броварському проспекті, 15. нові розробки у сфері оборони та авіакосмічної галузі представляють 332 підприємства, з яких – 35 закордонних компаній із 13 країн. Українську експозицію представляє: дк «укроборонпром» – 36 підприємств; гс «ліга оборонних підприємств України» – 23 підприємства; державне космічне агентство України – 2 підприємства, а також понад 130 інших державних і приватних підприємств

Fig. 4.2 – Fragment of the search results output form

The prog2_html output formatting program module looks like this:

```

#!/usr/bin/python2.7
import sys

```

```

import re
import datetime

t = sys.stdin.read()
json =t.split('\n')
t=""
for i in range(len(json)):
t=t+" "+json[i]
t=re.sub('^\\s','',t)
total = re.findall('"total" : (\d+)', t)
print "Content-type: text/html; charset=utf-8\n\n<html><body
bgcolor=yellow>"
print "<b>Found: "+total[0]+"</b><hr><ol>"
title = re.findall('"title" : "(.*?)",', t)
text = re.findall('"textBody" : "(.*?)",', t)
url = re.findall('"URL" : "(.*?)"', t)
for i in range(len(title)):
doc_ind=i+1
print "<li><b>"+": "+title[i]+"</b>"
print "<br>"+text[i]
print "<br><i>"+url[i]+"</i><br /><hr>"
print "</ol></body></html>"

```

Questions for practical work:

1. What types of queries are generated in Elasticsearch?
2. How does the DSL (Domain Specific Language) query language work in Elasticsearch?
3. How are search results sorted in Elasticsearch according to the relevance of each document to the query?
4. What is the role of the JSON REST API in Elasticsearch?

Tasks for independent work

1. Ensure the functioning of the means of interactive interaction with the user (for example, CGI) on the computer (server).
2. Familiarize yourself with basic search and filtering capabilities in Elasticsearch.
3. Familiarize yourself with the basics of HTML markup.

literature

1. Pranav Shukla, Sharat Kumar. Elasticsearch, Kibana, Logstash and new generation search systems. - St. Petersburg: Peter, 2019. - 363 p.
2. Goncharov A. HTML tutorial. - St. Petersburg: Peter, 2002. - 242 c.
3. M. Moka. Python for web applications, 2016. - 263 c.

5. Working with the Kibana system

The goal of the hands-on work corresponding to the chapter is to learn how to install and run the Kibana visualization system and to gain hands-on skills with the Console tools when working with the REST API for any possible Elasticsearch operations and the Discover and Visualizations tools for interactive data analysis.

Kibana is a visualization tool for included in the Elastic Stack, which, in particular, helps to visualize data from the Elasticsearch system. This tool is often called a window in the Elastic Stack. The Kibana system offers several visualization options, such as a histogram, map, line graphs, time series, etc.

The system not only allows you to visualize data, but also contains tools for developers, such as Console (Console).

Kibana also provides development tools. It is possible to manage the X-Pack settings for security in the Elastic Stack, and using the developer tools to create and test REST API requests.

Installing the Kibana system

To download the Kibana system to a computer (server), it is enough to go to the download page on the Internet at the address <https://www.elastic.co/downloads/kibana>, select the required operating system and activate the corresponding button (Fig. 5.1).

The Kibana system is distributed in the form of a ZIP file, which after deployment contains the set of files presented in Fig. 5.2.

To start the Kibana system, it is enough to run `kibana` or `kibana.bin` from the `bin` directory, depending on the operating system.

After that, it is enough to enter the address `http://localhost:5601` in the browser on the local computer to receive the response shown in Fig. 5.3.

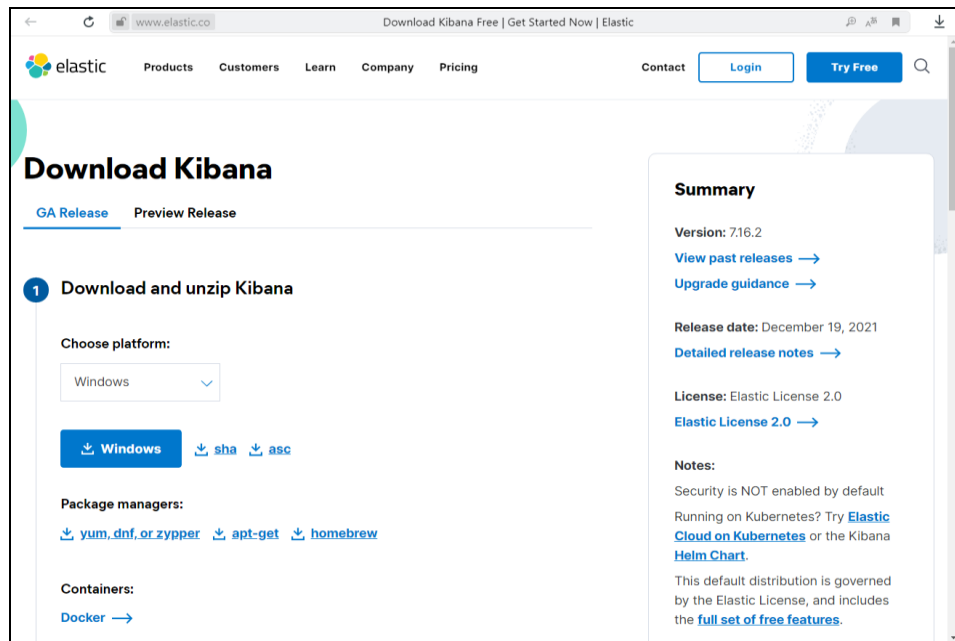


Fig. 5.1 - Online download page: <https://www.elastic.co/downloads/kibana>

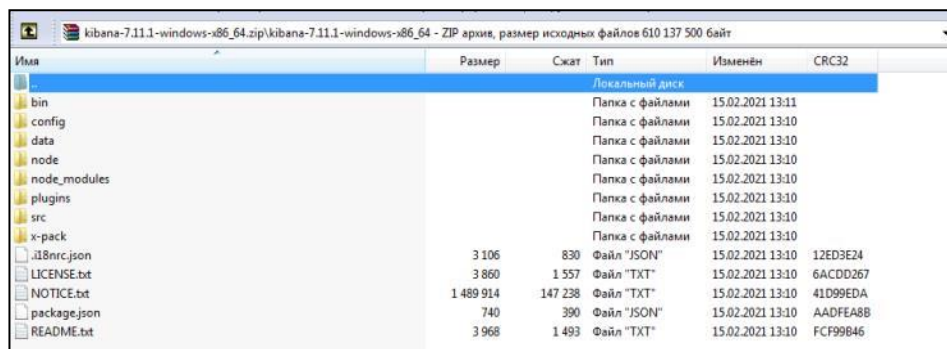


Fig. 5.2 – Contents of the Kibana system archive

Working with the Console tool

When following the hyperlink, we get a panel for selecting work modes (<http://localhost:5601/app/home#/3>). From this panel, first select work with the Console tool (http://localhost:5601/app/dev_tools#/console).

The Console tool will come in handy when working with the REST API for any possible Elasticsearch operation. That is, the Kibana system is needed not only for visualizations, but also as a user interface for interacting with the Elasticsearch system.

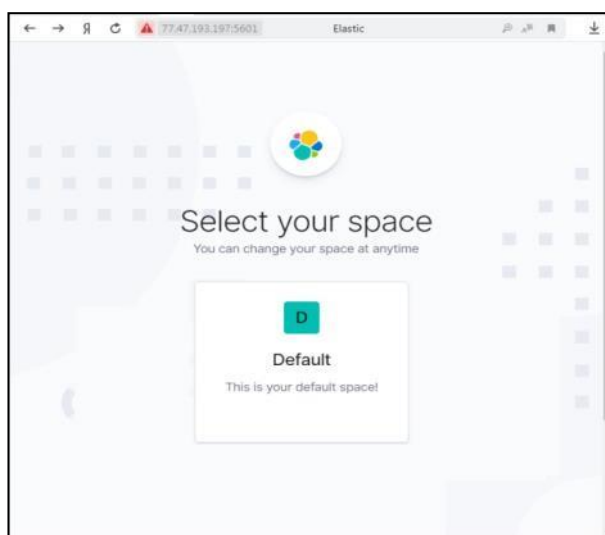


Fig. 5.3 - The first web page to access the Kibana system

Kibana Console is a convenient editor that supports the function of auto-completion and formatting of queries as they are written.

REST stands for Representational State Transfer. This is an architectural style for the interaction of systems with each other. REST evolved alongside the HTTP protocol, and almost all REST-based systems use HTTP as their protocol. HTTP supports various methods: GET, POST, PUT, DELETE, HEAD, etc. For example, GET is used to get or find something, POST is used to create a new resource, PUT can be used to create or restore an existing resource, and DELETE is used to permanently delete.

After starting Kibana, select the Dev Tools link in the left navigation bar. The console is divided into two parts: the editor field and the result field. REST API commands can be entered, and after clicking the green triangle, the request will be sent to Elasticsearch (or the cluster).

After selecting the Console tool, the corresponding Console interface is displayed, which is presented in Fig. 5.4

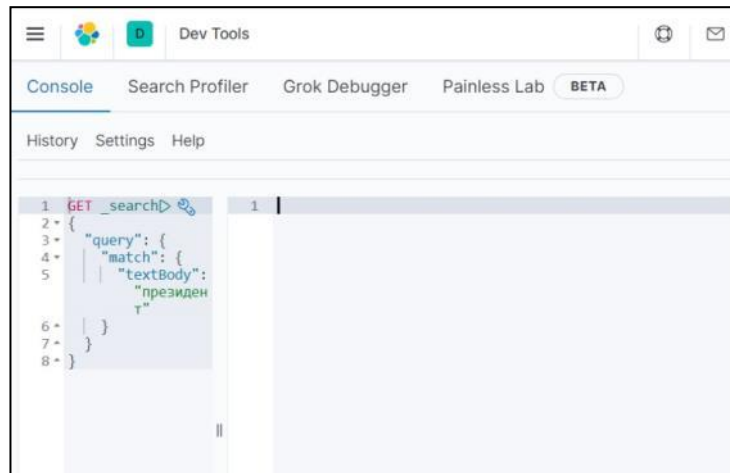


Fig. 5.4 – Console tool interface

On the example shown in Fig. 5.5, sent a GET request /. This is an analogue of the curl command, only much shorter. In this case, you don't need to type http, the host and port of the Elasticsearch node, namely http://localhost:9200.

As soon as we start typing in the console editor, we get a list of possible commands.

The following example shows how to get information about the number of entries in the hb index (GET /hb/_count command).

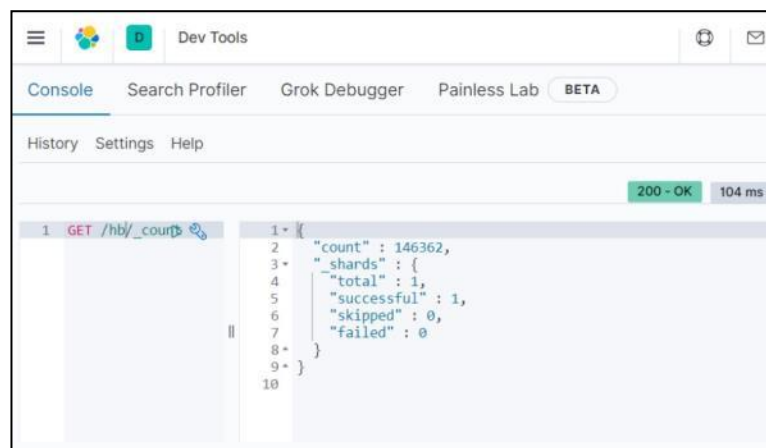


Fig. 5.5 – Information on the number of entries in the index

And Fig. 5.6 gives an example of processing requests of an already known (see section 4) structure:

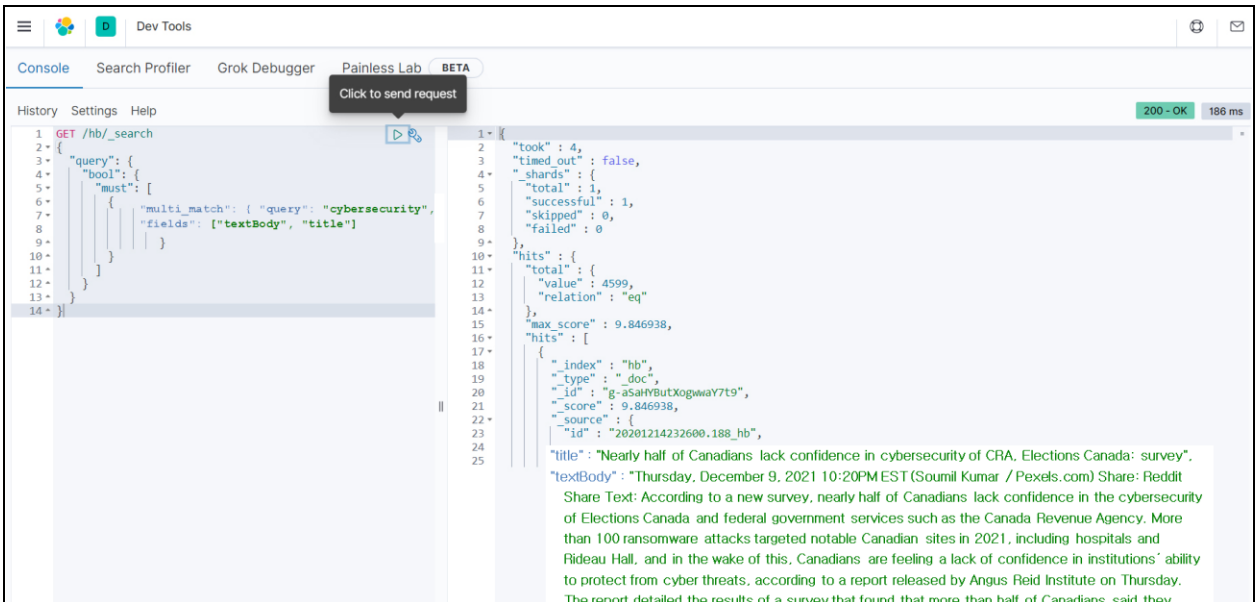


Fig. 5.6 – The result of processing a request for a word in two fields

The results of processing a search request in several fields with filtering by source are shown in Fig. 5.7.

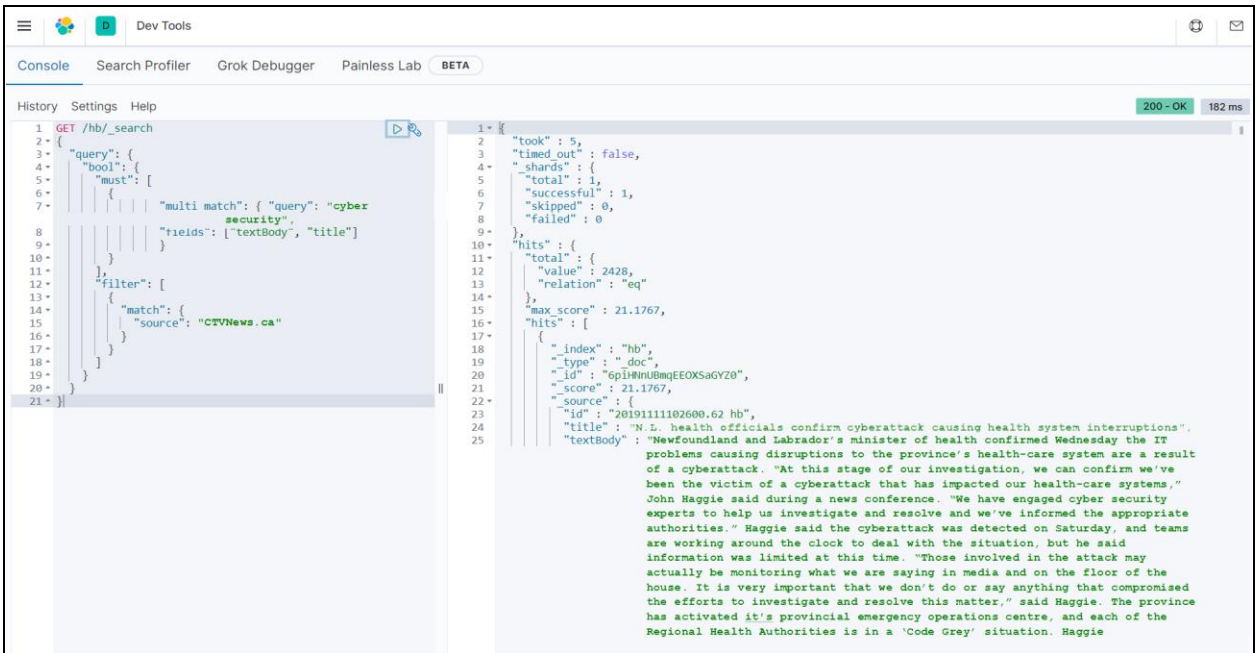


Fig. 5.7 – Result of query processing with filtering by the “source” field

Working with the Discover tool

The Kibana Discover tool is designed for interactive data analysis. With the help of this tool, the user can perform search and filter requests, view document

data. You can also save search queries or filter criteria for reuse or create visualizations based on filtered results.

On Fig. 5.8. the search results for the word Huawei in the hb index for a period of 200 days are given.

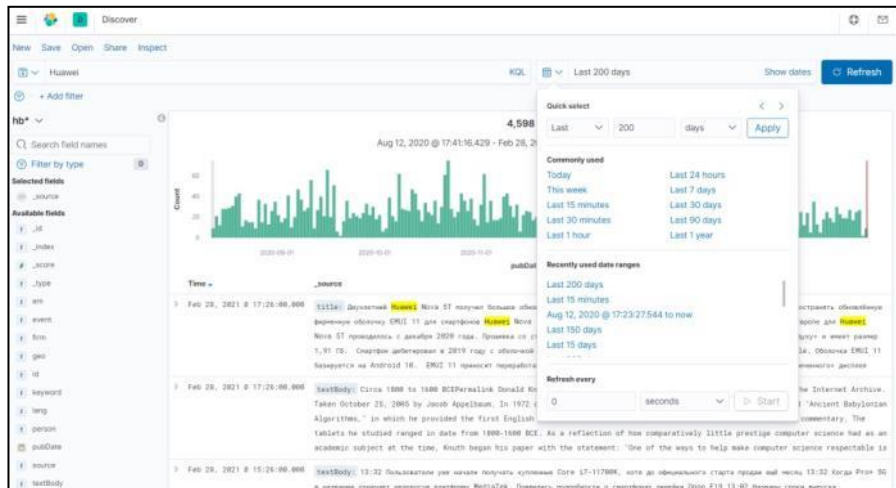


Fig. 5.8 – Results of search results analysis

The Discover tool allows you to display selected fields for research, add additional filtering based on the content of individual fields, etc. On Fig. 5.9 shows the results of a search for the word Huawei in the hb index (dates, source and titles) for a time range of 200 days, filtered by the word Huawei. As you can see, the diagram of the number of documents per day has also changed.

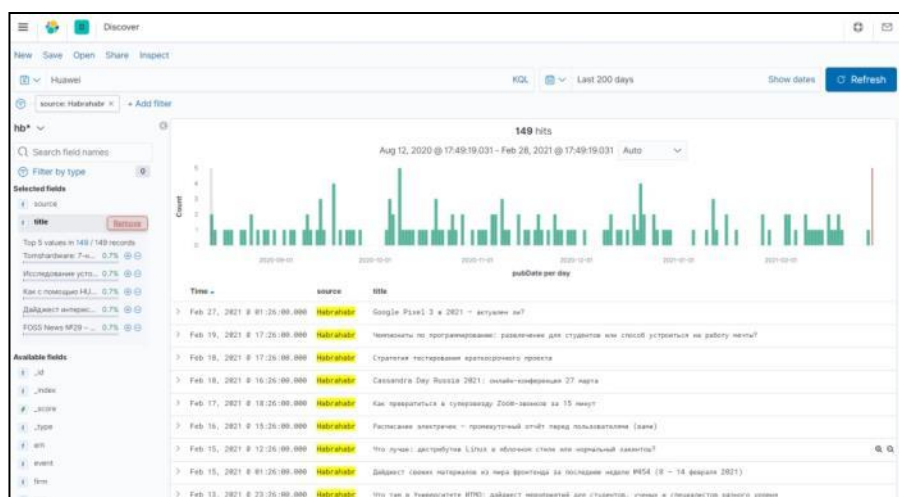


Fig. 5.9 – Results of the analysis of search results with filtering

Working with the Visualizations tool

The Kibana Visualizations tool is designed to create visual images based on data (visualizations). There are opportunities to create various design options: histograms, line charts, maps, tag clouds, etc. The user can choose the necessary visualizations to facilitate data analysis.

To create a new visualization, follow these steps:

1. Go to the Visualize page and click the Create a new Visualization button or the “+” button.
2. Select the visualization type.
3. Select the data source.

In the first example (Fig. 5.10), a visualization of the Area type was created, in which the abbreviation HDD was used as a search criterion, the affiliation of documents to the source Habrahabr was selected as a filter condition, the abscissa axis corresponded to a time period of 200 days. The hb index was considered as a source of information. The visualization interface looks like this.

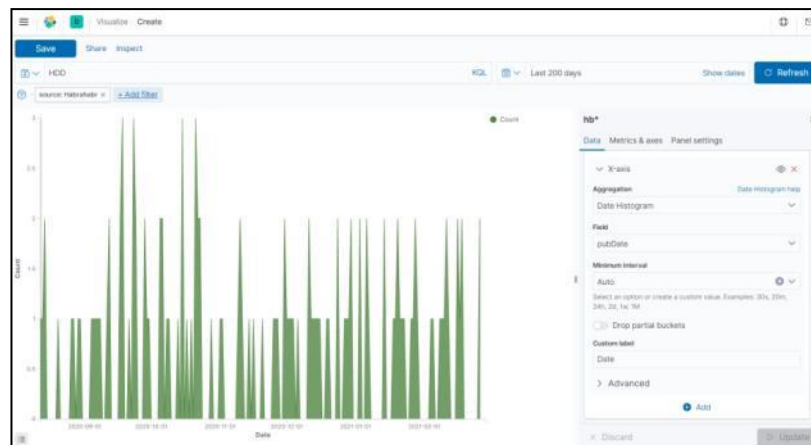


Fig. 5.10 – Area type visualization for search results

In the second example, a visualization of the Goal type was created, in which the word "safety" was used as a search criterion, and the names of sources (source.keywords) were used as the investigated parameter. The hb index was considered as a source of information. The visualization interface looks as follows (Fig. 5.11).

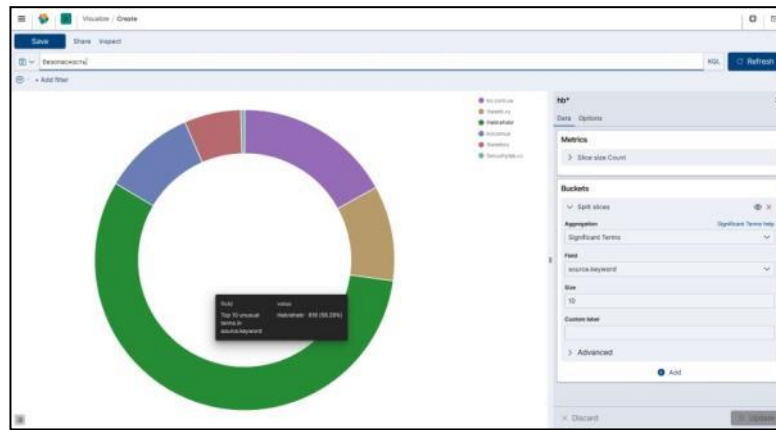


Fig. 5.11 - Goal type visualization for search results

In this chapter, we introduced Kibana as a data visualization and exploration tool for tasks such as log and time series analysis, application monitoring, and ongoing process monitoring. It offers powerful and easy-to-use features: bar charts, line graphs, pie charts, heat maps, and built-in geospatial support. In addition, Kibana provides close interaction with Elasticsearch, where it acts as a default visualization tool for data stored in Elasticsearch.

Questions for practical work

1. What is the Kibana system and what is its role in data processing by the Elasticsearch system?
2. What is the Console tool and its use when working with Elasticsearch?
3. What is the Discover tool and its capabilities when visualizing the results of Elasticsearch?
4. What is the Visualizations tool and its capabilities when visualizing the results of Elasticsearch?

Tasks for independent work

1. Ensure the functioning of the Kibana system on the computer.
2. Learn more about visualization capabilities in the Kibana system.
3. Build time charts of the number of posts from different RSS feeds.

literature

1. Pranav Shukla, Sharat Kumar. Elasticsearch, Kibana, Logstash and new generation search systems. - St. Petersburg: Peter, 2019. - 363 p.
2. Learning Kibana 5.0. Exploit the visualization capabilities of Kibana and build powerful interactive dashboards / Bahaaldine Azarmi. - Packt Publishing, 2017. - 275 p.

6. And aggregation in Elasticsearch

The purpose of the practical work corresponding to the section is to learn how to program and correctly implement generalized (aggregated) data obtained after analytical processing of results in Elasticsearch.

Analytics in Elasticsearch is designed to get a complete picture of the data. When searching, several records are examined in detail, while analytics allows you to look at the data more broadly and group it in different ways.

Implementation of aggregation in requests

Aggregation in Elasticsearch, which is defined by the aggregations element, allows you to receive data summarized by some characteristic. All aggregation requests look like this:

```
GET /<index_name>/<type_name>/_search
  {
    "query": { ... query type ... },
    "aggregations": {
      ... aggregation type ...
    }
  },
  "size": 0
}
```

The aggregations element must contain the actual aggregation request. The body of the request depends on the type of aggregation desired. The optional query element specifies the aggregation context, that is, if you need to limit the aggregation context, you must specify the query element. Aggregation will take into account all documents of a given index and type, unless the query element is specified (we can consider it equal to the match_all query, if there is no other query). For example, this parameter is indicated if it is necessary that the aggregation does not work on all data, but only on certain documents that meet specific conditions.

The query filters the documents that will be processed by the aggregations element. The size element specifies how many matching search documents should be returned in the response. The default value is 10. If size is not specified, the response will contain no more than ten relevant documents. Usually, if you want to get only the first test results of the aggregation, you should set the size element to 0 so that you don't get other results.

Dynamics of publications on request

To obtain the dynamics of the number of publications by query in Elasticsearch, it is necessary to aggregate data that correspond to the topic determined by the primary query by the field corresponding to the date and time value. Specifically, in order to aggregate by date all documents from the hb index that correspond to the Samsung query, by the time field, we enter the query in JSON format:

```
curl -XGET 'http://localhost:9200/hb/_search?pretty=true' -H 'Content-Type: application/json' -d '{
  "queries":
  {"multi_match":
  {"query" : "Samsung",
  "fields":
  ["title","textBody"]
  }
  },
  "aggregations":
  { "dates_with_holes":
  { "date_histogram":
  { "field": "pubDate",
  "interval": "day",
  "min_doc_count": 0
  }
  }
  },
  "size": 0
}'
```

As a result of the execution of this request, we receive the initial data for constructing a diagram - the answer looks like this:

```
{
  "took" : 2,
```

```

"timed_out" : false,
"_shards" : {
"total" : 1,
"successful" : 1,
"skipped" : 0,
"failed" : 0
},
"hits" : {
"total" : {
"value" : 10000,
"relation" : "gte"
},
"max_score" : null,
"hits" : [ ]
},
"aggregations" : {
"dates_with_holes" : {
"buckets" : [
{
"key_as_string" : "2019-07-27T00:00:00.000Z",
"key" : 1564185600000,
"doc_count" : 1
},
{
"key_as_string" : "2019-07-28T00:00:00.000Z",
"key" : 1564272000000,
"doc_count" : 2
},
. . .
{
"key_as_string" : "2021-03-04T00:00:00.000Z",
"key" : 1614816000000,
"doc_count" : 35
},
{
"key_as_string" : "2021-03-05T00:00:00.000Z",
"key" : 1614902400000,
"doc_count" : 40
}
]
}
}
}
}

```

This data can be obtained from the Console tool of the Kibana system (Fig. 6.1).

For further processing, it is enough to use the data corresponding to the `key_as_string` and `doc_count` fields.

```

1 GET /hb/_search
2 {
3   "query": {
4     "multi_match": {
5       "query": "Samsung",
6       "fields": ["title", "textBody"]
7     }
8   },
9   "aggregations": {
10    "dates_with_holes": {
11      "date_histogram": {
12        "field": "pubDate",
13        "interval": "day",
14        "min_doc_count": 0
15      }
16    }
17  },
18  "size": 0
19 }
20 }

1 #! Deprecation: [interval] on [date_histogram] is deprecated, use
2 [{"fixed_interval"} or [{"calendar_interval"}] in the future.
3 {
4   "took": 1,
5   "timed_out": false,
6   "_shards": {
7     "total": 1,
8     "successful": 1,
9     "skipped": 0,
10    "failed": 0
11  },
12  "hits": {
13    "total": {
14      "value": 10000,
15      "relation": "gte"
16    },
17    "max_score": null,
18    "hits": [ ]
19  },
20  "aggregations": {
21    "dates_with_holes": {
22      "buckets": [
23        {
24          "key_as_string": "2019-07-27T00:00:00.000Z",
25          "key": 1564185600000,
26          "doc_count": 1
27        },
28        {
29          "key_as_string": "2019-07-28T00:00:00.000Z",
30          "key": 1564272000000,
31          "doc_count": 2
32        }
33      ]
34    }
35  }
36 }

```

Fig. 6.1 - Aggregation results in the Kibana system console

Data transfer to CSV format

To use the obtained results in the environment of specialized digital data processing systems, you can convert them to CSV format using a Python program, the code of which is given below:

```

#!/usr/bin/python2.7
import sys
import re

t = sys.stdin.read()
json = t.split('\n')
t=""
for i in range(len(json)):
t=t+" "+json[i]

t=re.sub('\s', '', t)
days = re.findall('"key_as_string" : "(.+?)T', t)
count = re.findall('"doc_count" : (\d+)', t)

for i in range(len(days)):
print days[i]+" "+count[i]

```

As a result of running the program, you can get data in CSV format as follows:

```
2019-07-27; 1
2019-07-28; 2
2019-07-29;0
2019-07-30; 1
2019-07-31; 2
2019-08-01;0
2019-08-02;0
. . .
2021-03-02; 12
2021-03-03;32
2021-03-04;35
2021-03-05; 40
```

Data processing in the Excel environment

For further processing of the data in the format, we will load the received CSV file into the Excel environment and plot the message dynamics graph (Fig. 6.2).

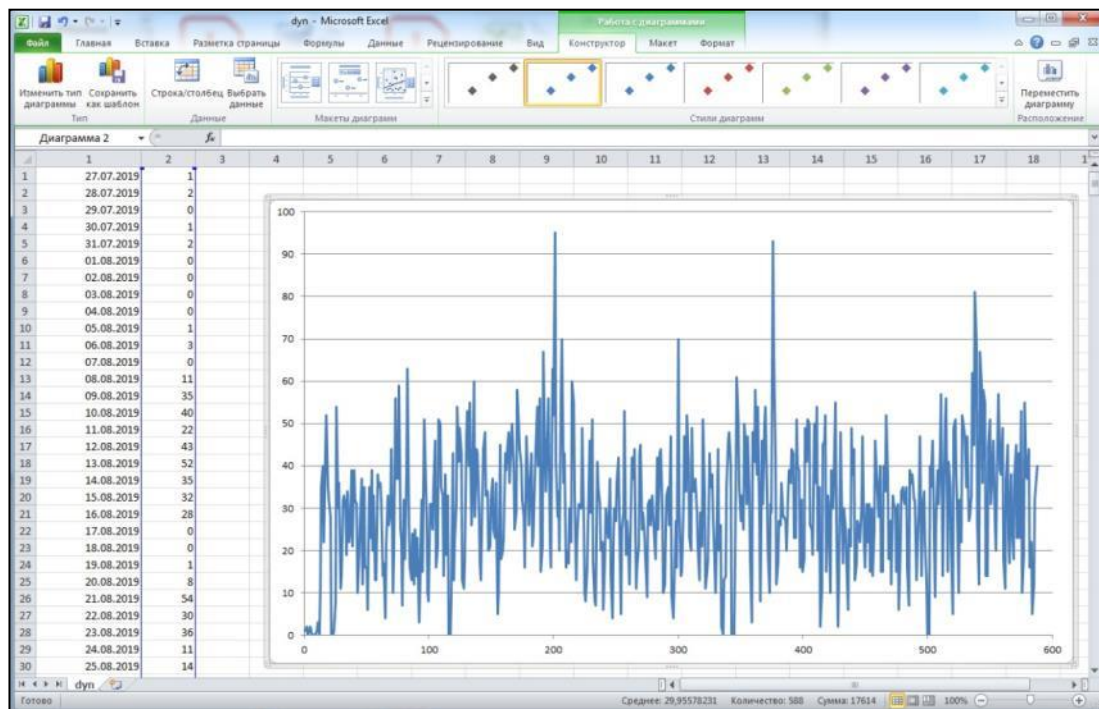


Fig. 6.2 – Data loaded into the Excel system and a graph constructed

After loading the CSV file into the digital data processing system, simple possibilities for its statistical processing arise. On Fig. 6.3. an example of finding the polynomial trend of this series is given.

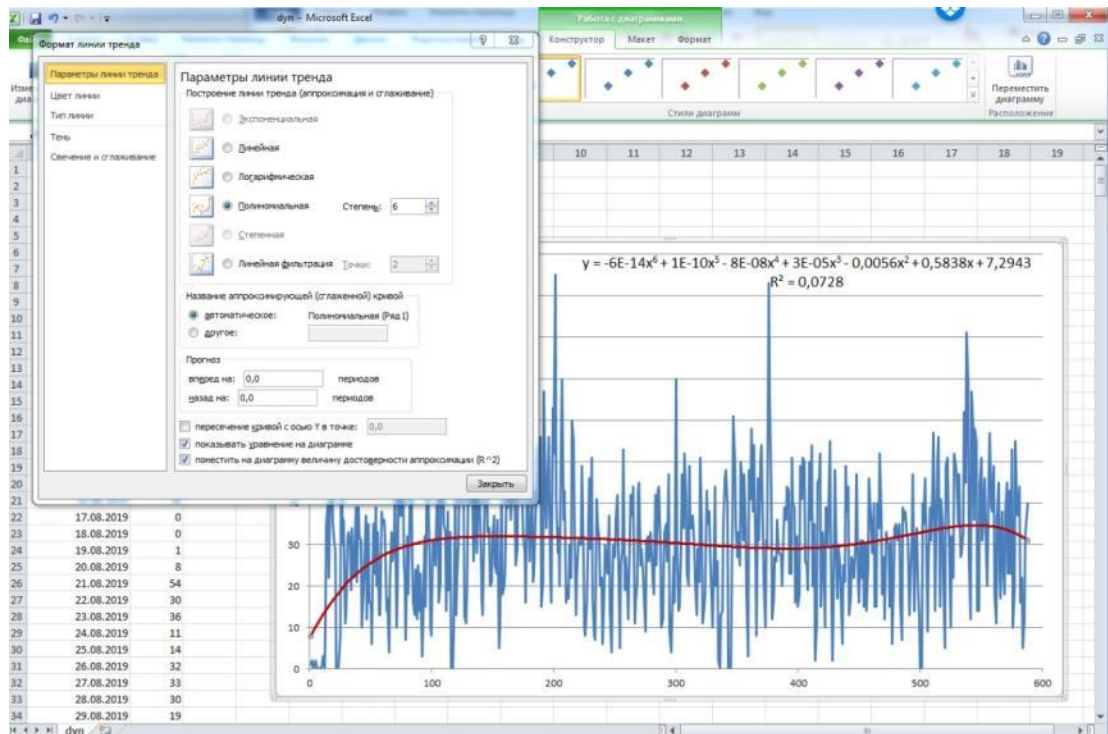


Fig. 6.3 – Polynomial trend of the investigated time series

Questions for practical work

1. What is data aggregation?
2. What types of aggregation are there in Elasticsearch?
3. What is CSV file format?
4. How to plot the downloaded data and determine the trend of the time series

Tasks for independent work

1. Upload a test database with accumulated data from RSS feeds from the Internet to the database of the Elasticsearch system.
2. Aggregate this data independently by the date-time field.

3. Upload the obtained results to an available DSP (Digital Signal Processing) type system.

literature

1. Pranav Shukla, Sharat Kumar. Elasticsearch, Kibana, Logstash and new generation search systems. - St. Petersburg: Peter, 2019. - 363 p.
2. Kuzmychev A.I. Econometric modeling and forecasting in Excel. Education village / Byshovets N.G., Kuzmychev A.I., Medvedev M.G., Ometsynska N.V. - K.: VOC of AMU, 2010. - 324 p.

7. Statistical processing of aggregated data

The purpose of the practical work corresponding to the section is to acquire practical skills in processing text documents obtained from the Internet, the ability to apply general methods and means of preparation, statistical processing, visualization and analysis of data aggregated according to some feature, using the main libraries of the Python language for working with "big data".

Development environment

WinPython tools (<https://winpython.github.io/>) are expected to be used as a programming environment for statistical data processing tasks. To do this, we install the software, which, in particular, is located on the website [sourceforge.net](https://sourceforge.net/projects/winpython/files/WinPython_3.8/3.8.8.0/) (https://sourceforge.net/projects/winpython/files/WinPython_3.8/3.8.8.0/). On Fig. 7.1 shows a fragment of a WinPython web page.

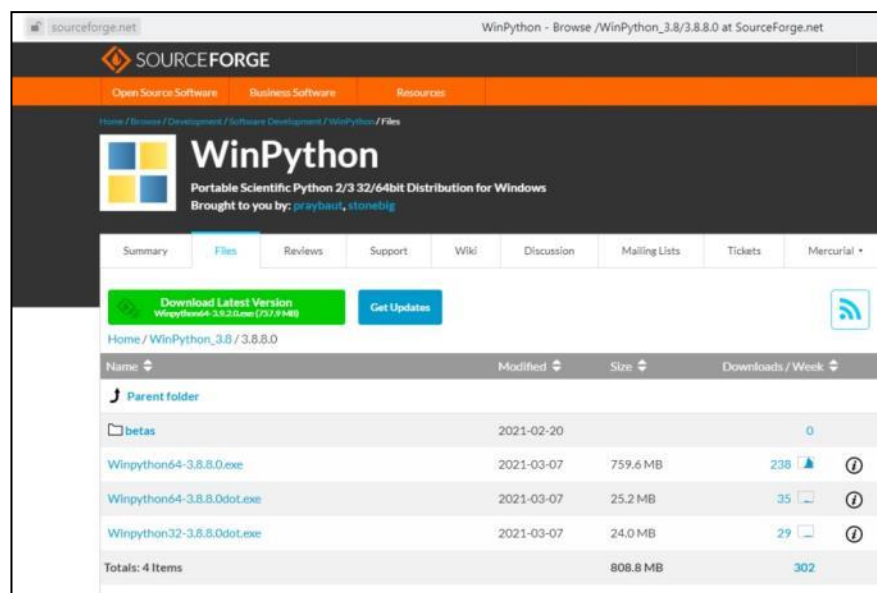


Fig. 7.1 – A fragment of the web page of the WinPython programming system

After deploying the WinPython distribution, you can activate the Jupyter Lab web development environment, which becomes available at <http://localhost:8888/lab> (Figure 7.2). It is in this programming environment that Python 3.8-based software is developed, which displays, among other things, graphical images.

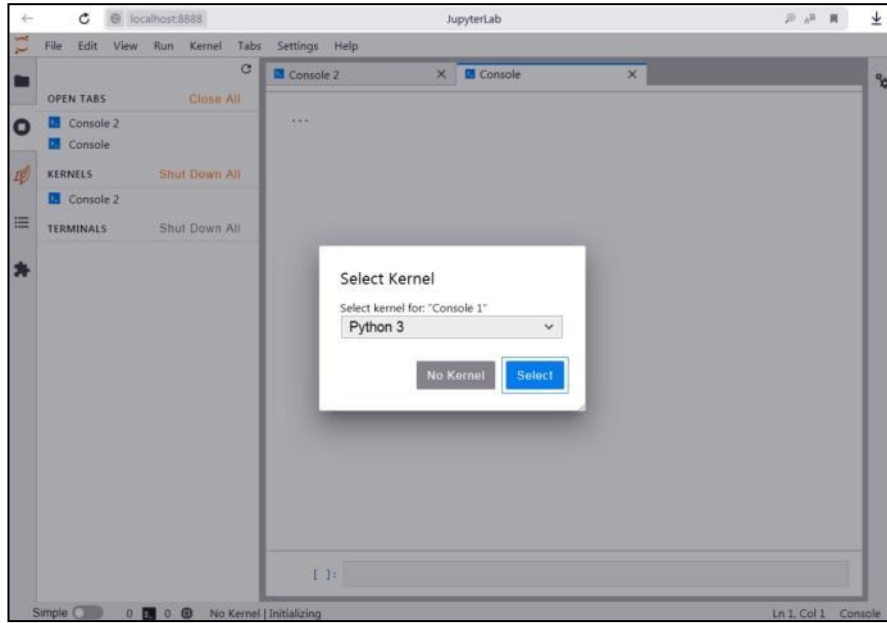


Fig. 7.2 - Downloading the Jupyter Lab development environment

Window smoothing

In the previous section, it was shown how as a result of aggregation a time series is obtained, the behavior of which can be determined by statistical processing. In some cases, it is useful to consider a smoother version of the original time series $D = \{d_t\}$, $t = 1, \dots, N$. Smoothing helps reveal significant trends in the dynamics of the series, while hiding noise and various features that appear at small scales. There are various methods of smoothing. The simplest way of smoothing is to calculate a windowed moving average. A simple moving average is equal to the average arithmetic value of the elements of a series from an interval of a given length, namely:

$$S = \{s_t\}, \quad s_t = \frac{1}{w} \sum_{k=-[w/2]}^{[w/2]} d_{t+k}$$

where w is the width of the smoothing interval (the number of elements for which the average is calculated), s_t is the value of the window moving average at point $(-[w/2] \leq t \leq [w/2])$.

When using windowed smoothing, the larger the width of the smoothing interval, the smoother the resulting function will be. On Fig. 7.3 shows how the smoothed series looks D when the value is increased w to 8.

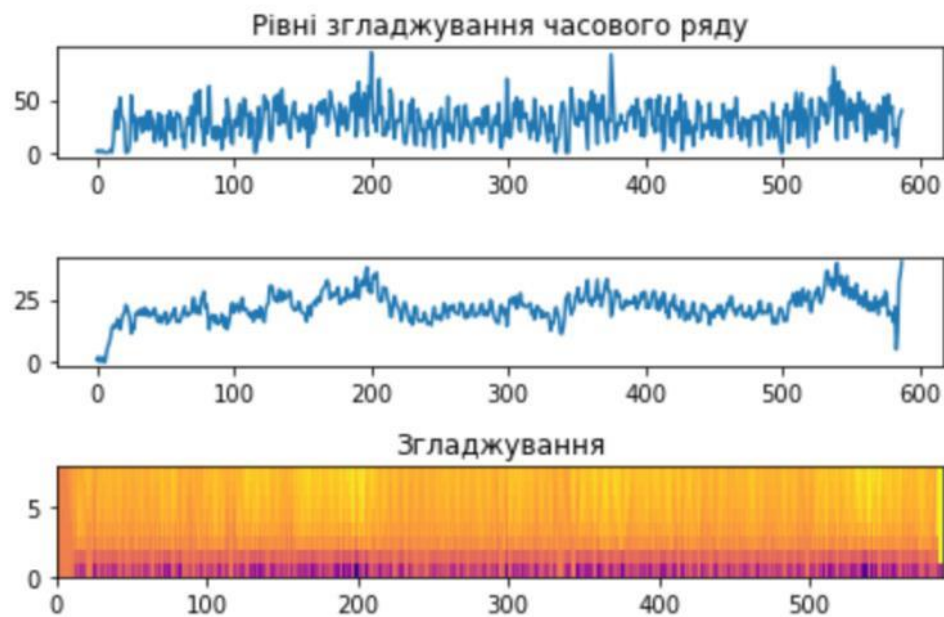


Fig. 7.3 – Results of window smoothing of the time series

The results of series smoothing can be demonstrated in a graph, in which the abscissa axis corresponds to the time axis, and along the y axis the delayed width smooths the interval. The graph shows the values - that is, the elements of the smoothed series at a point when using the width interval w .

Below is the program code in Python 3.8, which visualizes a diagram of changes in time series values with different smoothing windows (y-axis).

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import LogNorm

D=[1, 2, 0, 1,2, 0, 0, 0, 0, 1, 3,0, 11, 35,40,22,43,52,35,32,
.
.
.
38,45,23,43,23,53,10,19,55,38,37,44,16,22,5,12,32,35,40]

M = 8
N=len(D)
Z=np.random.rand(M,N)
C=D
for i in range(N):
for j in range(M):
Z[j][i]=D[i]
for j in range(M):
```

```

for k in range(j,Nj):
Z[j][k]=0;
for l in range(1,j):
Z[j][k]=Z[j][k]+D[kl]
Z[j][k]=Z[j][k]+D[k+1]

Z[j][k]=Z[j][k]-D[k]
Z[j][k]=Z[j][k]/(2*j+1)
fig,(ax0,ax1,ax2)=plt.subplots(3,1)
ax0.plot(D)
ax0.set_title('Time series smoothing levels')
fig.tight_layout()

for i in range(N):
C[i]=Z[7][i]

ax1.plot(C)
fig.tight_layout()
ax2.pcolor(Z,cmap='plasma')
ax2.set_title('Smoothing')
fig.tight_layout()
plt.show()

```

Exponential smoothing

Another frequently used method of series smoothing is exponential smoothing (Fig. 7.3). Previous values of the row are taken into account with exponentially decreasing weight values. We will denote the elements of the smoothed series s_t and immediately define $s_0 = d_0$. The following elements of the series are obtained by the recursive formula:

$$s_t = \alpha d_t + (1 - \alpha)s_{t-1}$$

where $0 \leq \alpha \leq 1$ is the smoothing coefficient. It is obvious that the $\alpha = 1$ resulting series $S = \{s_t\}$, $t = 1, \dots, N$ coincides with the original one. Thus, if the value α is close to 1, then the greatest weight in the determination s_t is assigned to the corresponding d_t , and the history of the series has less importance. On the other hand, if α equal to 0, then the entire series would be smoothed to one value $s_t = d_0$. That is, when α close to 0, the history of the series is taken into account with greater weight than the current value.

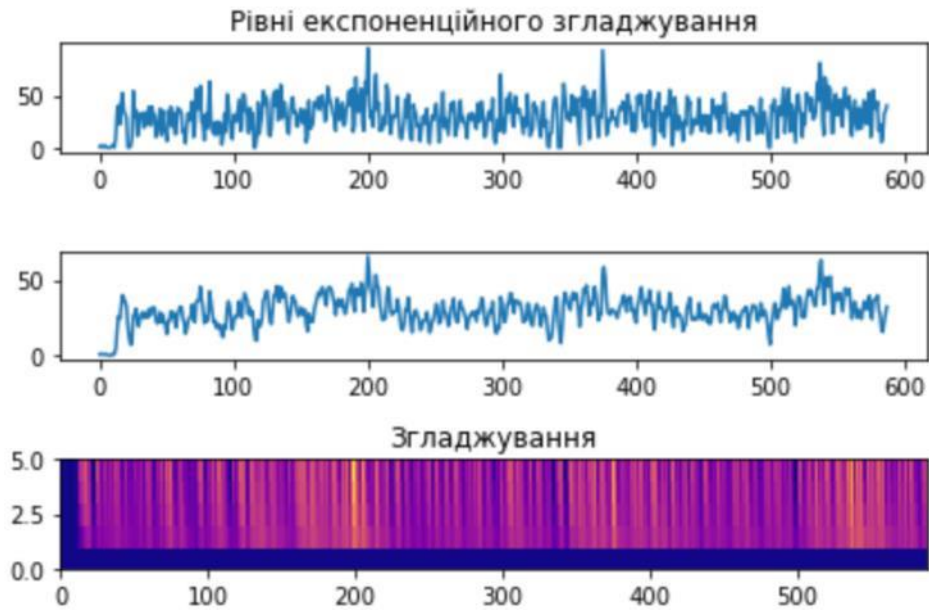


Fig. 7.3 – Results of exponential smoothing of the time series

Below is the code of the program in Python 3.8, which visualizes the diagram of changes in the values of the time series with different coefficients of exponential smoothing (ordinate axis).

```

M=5
N=len(D)
Z=np.random.rand(M,N)
C=D
for i in range(N):
    for j in range(M):
        Z[j][i]=D[i]

for j in range(M):
    alf=j/M
    for k in range(1,N):
        Z[j][k]=D[k]*alf+Z[j][k-1]*(1-alf);

fig,(ax0,ax1,ax2)=plt.subplots(3,1)
ax0.plot(D)
ax0.set_title('Exponential smoothing levels')
fig.tight_layout()
for i in range(N):
    C[i]=Z[2][i]
ax1.plot(C)
fig.tight_layout()
ax2.pcolor(Z,cmap='plasma')

```

```
ax2.set_title('Smoothing')
fig.tight_layout()
plt.show()
```

Construction of a wavelet-scalogram

Wavelet analysis also belongs to the circle of modern instrumental means of evaluating series of observations. It is especially effective in those cases when, in addition to general spectral characteristics, it is necessary to detect local time-specific features of the behavior of the process under investigation. The basis of wavelet analysis is the wavelet transform, which is a special type of linear transform whose basis functions (wavelets) have specific properties. Data analysis using wavelet transforms is a convenient, reliable and powerful tool for time series research and allows you to present the results in a visual way that is convenient for interpretation.

A wavelet (small wave) is a function that is concentrated in a small neighborhood of a point and sharply decreases to zero as it moves away from it in both the time and frequency domains. There are a wide variety of wavelets with different properties. However, all wavelets take the form of short wave packets with zero integral value, localized on the time axis, which are invariant to shift and to scaling.

Two operations can be applied to any wavelet:

- displacement, i.e. moving the area of its localization in time;
- scaling (stretching or compression) .

The main idea of the wavelet transform is that the non-stationary time series is divided into separate intervals (the so-called " observation windows"), and on each of them the scalar product (quantity showing the degree of proximity of two regularities) of the studied data with different shifts is calculated some wavelet on different scales.

Based on the basic wavelet, a family of functions is built using stretching/compression and parallel transfer. This is necessary to examine different areas of the output signal and with different degrees of detail.

With the help of a continuous wavelet transformation, the sections of the investigated series that are most similar in shape to the wavelet are detected. The idea is to compare parts of a series to some pattern at different scales.

A transform wavelet is a correlation between the original time series and some underlying wavelet. The wavelet transform generates a set of coefficients that represent the original series. They are functions of two variables: time and frequency, and therefore form a surface in three-dimensional space. Thus, the wavelet transform depends on the position of the wavelet on the time axis and its scale. The considered processes are clearly visible both on the wavelet-scalegrams and on their corresponding skeletons (graphs of extremum lines). These coefficients show how the behavior of the process at a given point is similar to a wavelet at a given scale. The closer the analyzed dependence within a given point is to the type of wavelet, the greater the absolute value of the corresponding coefficient. The application of these operations, taking into account the locality property of the wavelet in the frequency-time domain, allows analyzing data on different scales and accurately determining the position of their characteristic features in time.

On the scales you can see all the characteristic features of the original series: the scale and intensity of periodic changes, the direction and value of trends, the presence, location and duration of local features.

The most common real basis wavelet functions are constructed based on the derivatives of the Gaussian function ($g_0(t) = \exp(-t^2 / 2)$) (Fig. 7.4). This is due to the fact that the Gaussian function has the best localization indicators in both the time and frequency domains. When $n = 1$, we get a wavelet of the first order, the

so-called WAVE-wavelet with zero zero moment. When $n = 2$, we get the MHAT wavelet, the "Mexican Hat" (Mexican Hat), when $n = 3$ – the Morle wavelet.

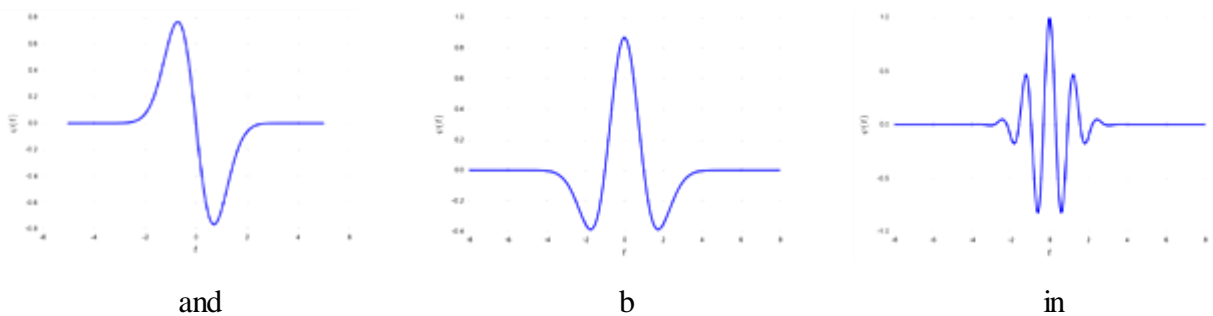


Fig. 7.4 – Examples of wavelets: (a) Gaussian wave (first derivative of a Gaussian function), (b) Mexican hat, (c) Morlet wavelet (real part)

The obtained wavelet coefficients can be presented graphically. If we set aside the wavelet shift (time axis) on one axis, and the scales on the other (scale axis), and color the points of the obtained scheme depending on the value of the corresponding coefficients (the larger the coefficient, the brighter the colors).

The wavelet spectrum $W_f(a,b)$ (wavelet spectrum, or time-scale-spectrum - scale-time spectrum) is a function of two arguments: the first argument a (time scale) is analogous to the period of oscillations, that is, the inverse of the frequency, and the second b is analogous to the displacement of the signal along the time axis.

It should be noted that $W_f(a_0,b)$ characterizes the time dependence (at $a = a_0$), while the dependence $W_f(a,b_0)$ can be matched by the frequency dependence (at $b = b_0$).

If the investigated signal $f(t)$ is a single pulse of duration τ_u , concentrated in the vicinity of $t = t_0$, then its wavelet spectrum will have the greatest value in the vicinity of the point with coordinates $a = \tau_u$, $b = t_0$.

The obtained coefficients are represented graphically by a map of conversion coefficients, or scale (Fig. 7.5). Wavelet displacements are plotted on one axis (time axis) and scales on the other (scale axis), after which the points of the obtained scheme are colored depending on the value of the corresponding coefficients (the larger the coefficient, the brighter the image colors). The scale chart shows all the characteristic features of the original series: the scale and intensity of periodic changes, the direction and magnitude of trends, the presence, location, and duration of local features.

```
import matplotlib.pyplot as plt

import pywt

f_s = 100 # Sampling rate

x=[1, 2, 0, 1,2, 0, 0, 0, 0, 1, 3,0, 11, 35,40,22,43,52,35,32,
.
.
.
45,23,43,23,53,10,19,55,38,37,44,16,22,5,12,32,35,40]

N=len(x)
t= range(N)

##### Visualization

fig, (ax1, ax4) = plt.subplots(2,1, sharex = True, figsize = (10,8))

# Signal
ax1.plot(t, x)
ax1.grid(True)
ax1.set_ylabel("Number of publications")
ax1.set_title("Dynamics of publications")

# Wavelet transform, ie scaleogram

cwtmatr, freqs = pywt.cwt(x, range(1, N), "mexh", sampling_period = 1
/ f_s)
ax4.pcolormesh(t, freqs, cwtmatr, vmin=-100, cmap = "inferno" )
ax4.set_ylim(0,10)
ax4.set_ylabel("Scale")
ax4.set_xlabel("Time")
ax4.set_title("Scalogram on MexH wavelet bases")

# plt.savefig("./fourplot.pdf")

plt.show()
```

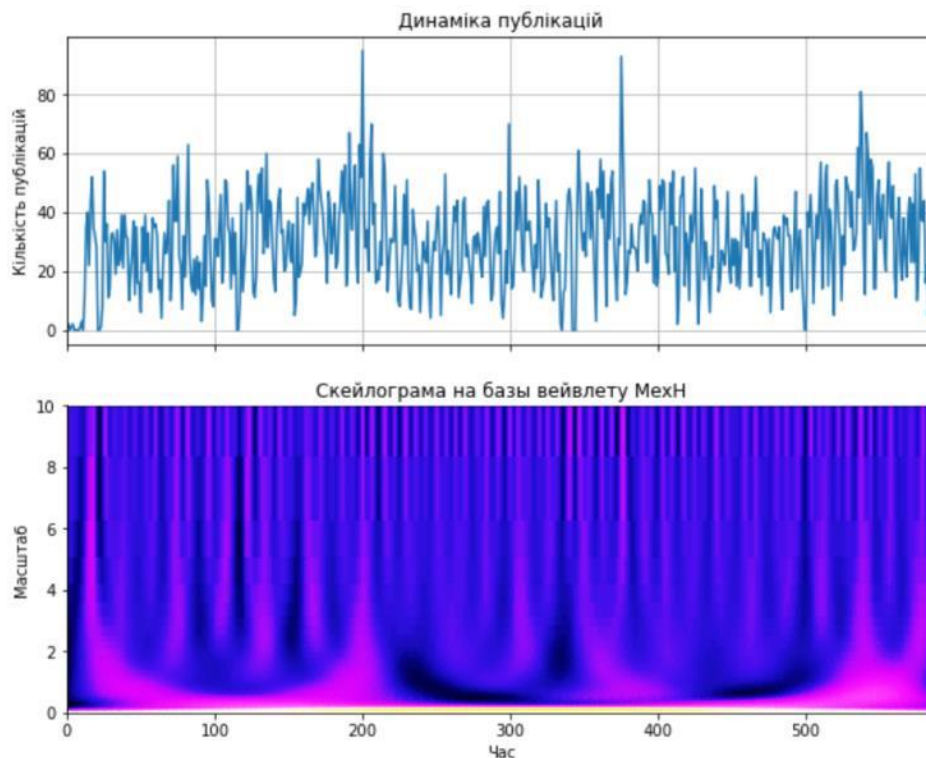


Fig. 7.5 – An example of wavelet scalelogs of the original time series

Questions for practical work

1. What is the essence of window smoothing in statistical data processing?
2. What is exponential smoothing in statistical data processing?
3. What are the advantages of wavelet analysis (wavelet transform) when studying time series and interpreting the results of *Big Data* analysis?
4. Name the most common valid basic wavelet functions?

Tasks for independent work

1. Install the Python system on the computer with support for the matplotlib graphics library.
2. Learn more about the capabilities of the matplotlib and pywt libraries.
3. Construct a wavelet scalogram with another basic wavelet function. Investigate the difference of wavelet scales with different basic wavelet functions.

literature

1. Sandro Tosi. Matplotlib for Python Developers. – Packt Publishing, 2009. – 308 p.
2. Fabio Nelli. Python Data Analytics: Data Analysis and Science Using Pandas, matplotlib, and the Python Programming Language. – Apress, 2015. – 350 p.
3. Polykar R. Introduction to wavelet transformation. – St. Petersburg: AVTEX, 2001. – 59 p.
4. Recognition of information operations. A.G. Dodonov, D.V. Lande, V.V. Tsyganok et al. – Kyiv: Engineering, 2017. – 282 p.
5. Fundamentals of the theory and practice of intelligent data analysis in the field of cyber security: a study guide / D.V. Lande, I.Yu. Subach, Yu.E. Boyarinova. – Kyiv: ISZZI KPI named after Igor Sikorskyi, 2018. – 300 p.

8. Extracting keywords from documents selected on request

The purpose of the practical work corresponding to the section is to acquire practical skills for analytical processing of text documents obtained using the Elasticsearch search engine. The result of processing the array of documents is a dictionary sorted by the frequency of word occurrence, after processing which we get the most important words related to the given topic.

Receiving an array of documents

For further processing of text documents, in particular, extraction of the most important keywords, it is necessary to obtain test documents from the Elasticsearch database. Obtaining such documents relevant to some thematic query (for example, the keyword Samsung) is carried out by means of a query to the Elasticsearch system, for example, to the information stored under the hb index, which can be entered through the Kibana system (Fig. 8.1):

```
{
"queries":
{"multi_match":
{"query" : "Samsung",
"fields":
["title","textBody"]
}
},
"size": 1000
}
```

The selected information is provided in the right frame of the Kibana interface (Console), after which it is stored in the samsung.txt file in JSON format (Fig. 8.1).

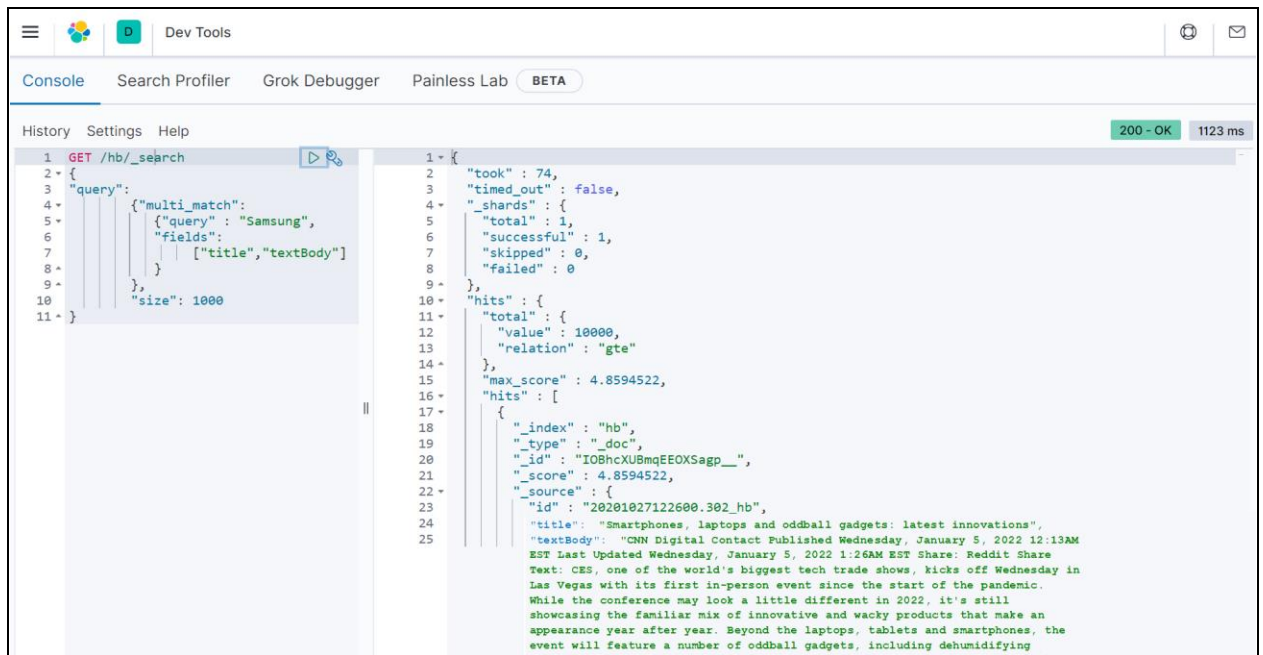


Fig. 8.1 – Selected documents in JSON format

Forming a dictionary

To form a dictionary of words from the received array of documents D placed in the `samsung.json` file, in the Python environment we will combine the contents of all the "title" and "textBody" fields, replace all separator characters with blanks, and use regular expressions to define all words. After sorting the dictionary of words, we determine only unique words and their absolute frequency of occurrence in the array D . For each word t , this value is $tf(t)$. A snippet of the Python source code is below:

```

import re
import string
f = open("G:/samsung.json", "r")
t = f.read()
f.close()

json=t.split('\n')
t=""
for i in range(len(json)):
t=t+" "+json[i]
#print(t)
title = re.findall('"title" : "(.+)"source"', t)
t=""

```

```

for i in range(len(title)):
t=t+" "+title[i]
t=re.sub('"textBody" :',' ',t)
t=re.sub('[--""[\]\\/?0-9",.()$+>><<-:;_...]', ' ',t)
t=t.upper()
t=re.sub('\s\w\s',' ',t)
t=re.sub('\s\w\w\s',' ',t)
t=re.sub('\s\s+',' ',t)
word = t.split(' ')
word.sort()

# Dictionary building
d={}
old=""
n=0;
for i in range(len(word)):
if (word[i] == old):
n=n+1;
otherwise:
#print(old,n)
d[old]=n
old=word[i]
n=1
d[old]=n
#print(d)

sorted_dict = {}
sorted_keys = sorted(d, key=d.get, reverse=True) # [1, 3, 2]

for w in sorted_keys:
sorted_dict[w] = d[w]

print(sorted_dict)

```

As a result of the program, we get a list of words sorted by frequency (Fig. 8.2). As you can see, the most frequent words also include words that do not carry a content load, the so-called "stop words". Lists of stop words for various languages compiled by linguists can be found on the Internet, for example, at <https://code.google.com/archive/p/stop-words/>.

Choosing the most important words

To determine the weight w of the word t , we will use the frequency method, namely, we will select the most frequent words from the dictionary array formed on the basis of the analysis:

$$w(t) = \sum_{\{d: t \in d\}} tf(t,d),$$

where $f(t,d)$ is the frequency of word t in document d .

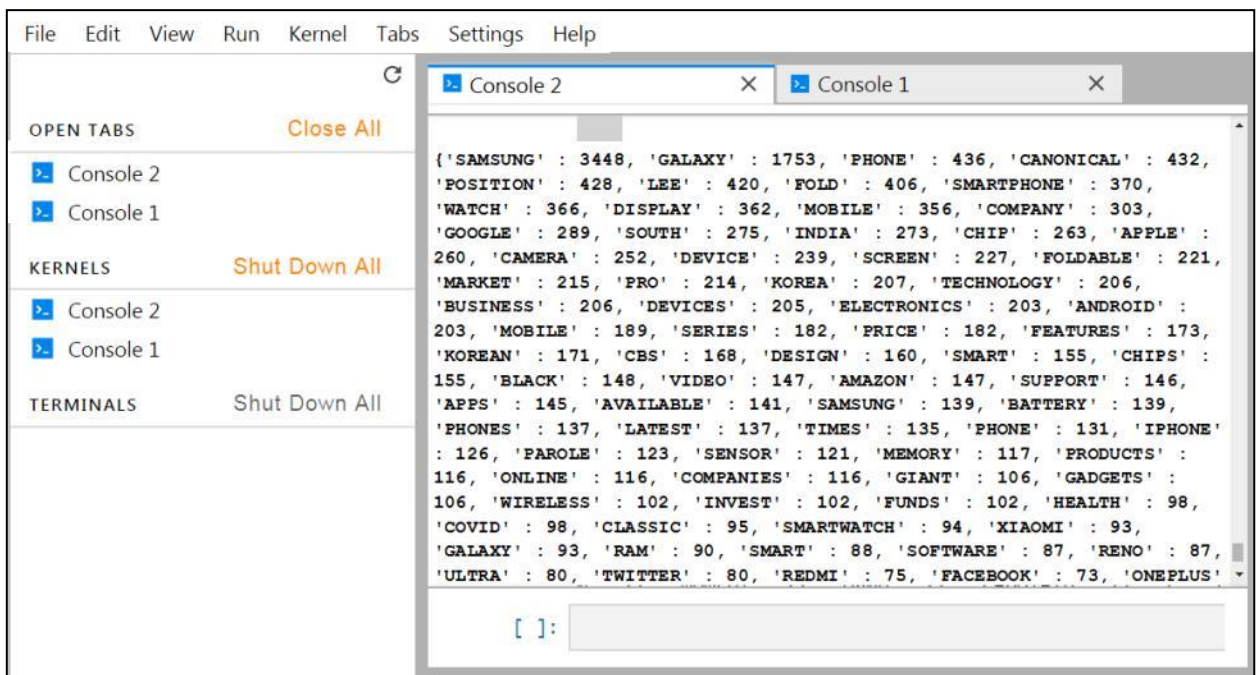


Fig. 8.2 – Selected words and their frequency of occurrence

As you can see, the most frequent words also include words that do not carry a content load, the so-called "stop words". Lists of stop words for different languages compiled by Google can be found on the Internet, for example, at <https://code.google.com/archive/p/stop-words/>

For further use, M the most important words are selected, for example, $M = 50$ such as are not included in the stop dictionary. A fragment of the program code in Python is given below:

```
f = open("G:/stop.txt", "r")
t = f.read()
f.close()
```

```

t=t.upper()
stop =t.split('\n')
M=50
j = 1
sorted_dict = {}
sorted_keys = sorted(d, key=d.get, reverse=True) # [1, 3, 2]

for w in sorted_keys:
sorted_dict[w] = d[w]
pr=0
for i in range(len(stop)):
if (stop[i] == w):
pr=1
if (pr == 0):
print(j,w,sorted_dict[w])
j=j+1
if (j > M):
break

```

Below is a list of the top 20 keywords from the Elasticsearch database documents by topic as determined by Samsung's query:

```

3448 SAMSUNG
1753 GALAXY
436 PHONE
432 CANONICAL
428 POSITIONS
420 LEE
406 FOLD
370 SMARTPHONES
366 WATCH
362 DISPLAY
356 MOBILE
303 COMPANIES
289 GOOGLE
273 INDIA
263 CHIP
260 APPLE
252 CAMERAS
239 DEVICES
227 SCREEN
215 MARKET

```

The obtained results are written to disk (file G:/words.txt).

Questions for practical work

1. How is the selection of the most important words for a given topic?
2. What is the further purpose of using the received words on the given topic?
3. What is the essence of another method of calculating the weight of

- words, in particular the TF-IDF method?
4. What is the essence of another method of calculating the weight of words, in particular, the dispersion method?

Tasks for independent work

1. Learn more about text string processing in Python.
2. Use the Python language to independently develop a program module for calculating the weight of words.
3. Familiarize yourself with other methods of calculating the weight of words, including the TF-IDF method, the dispersion method, and the horizontal visibility method (HVG).

literature

1. Lande D.V., Snarskyi A.A., Bezsudnov I.V. Internet: Navigation in complex networks: models and algorithms - M.: Librokom (Editorial URSS), 2009. - 264 p.
2. Fundamentals of the theory and practice of intelligent data analysis in the field of cyber security: a study guide / D.V. Lande, I.Yu. Subach, Yu.E. Boyarinova. – Kyiv: ISZZI KPI named after Igor Sikorskyi, 2018. – 300 p.

9. Formation of a network of concepts. Visualization in Gephi

The purpose of the practical work corresponding to the section is to gain practical skills in forming a network of keywords corresponding to the most important concepts from the text array, concepts obtained during the analytical processing of text documents, and to learn the principles of the Gephi system, designed to display graph structures in which concepts will be considered as nodes, and connections between them as edges.

Formation of the concept adjacency matrix

It is known that a graph structure or network can be defined by an adjacency matrix. Documents and arrays of keywords are analyzed to form a network of concepts, the nodes of which will be concepts, and the links between them will be the edges. The following approach is used when forming the concept adjacency matrix: two concepts are considered connected if they are part of the same sentence. A sentence is defined as a part of the text separated by appropriate punctuation marks. At the same time, the strength of the connection of concepts corresponds to the number of sentences where the corresponding words appear at the same time. By default, it is assumed that the weight of the connection of the concept with itself is zero.

Concept adjacency matrix $A = \{a_{i,j}\}$, where $a_{i,j}$ is the strength (weight) of the connection between concepts i and j .

To display the network in the Gephi graph analysis and visualization system, it is necessary to upload the adjacency matrix to it, which must be given in the following format:

$Concept_1; Concept_2; Concept_3; \dots; Concept_M$
 $Concept_1; a_{1,1}; a_{1,2}; a_{1,3}; \dots; a_{1,M}$
 $Concept_2; a_{2,1}; a_{2,2}; a_{2,3}; \dots; a_{2,M}$
 $Concept_3; a_{3,1}; a_{3,2}; a_{3,3}; \dots; a_{3,M}$
 ...
 $Concept_M; a_{M,1}; a_{M,2}; a_{M,3}; \dots; a_{M,M}$

Below is a fragment of the code in the Python programming language, with which the words.csv file is formed, using as words corresponding to the concepts, the words obtained using the tools described in the previous section (file G:/words.txt):

```

import re
import string
import numpy as np

f = open("G:/samsung.json", "r")
t = f.read()
f.close()
json = t.split('\n')
t=""
for i in range(len(json)):
t=t+" "+json[i]
title = re.findall('"title" : "(.+)"source"', t)
t=""
for i in range(len(title)):
t=t+" "+title[i]
t=re.sub('"textBody" : ', '.', t)
t=re.sub('[--%"" [\]\ /0-9", ()$+><<-:;_...]', ' ', t)
t=t.upper()
sent = t.split('.')
print(sent)
f = open("G:/word.txt", "r")
t = f.read()
f.close()
t=t.upper()
w = t.split('\n')
for i in range(len(w)):
s =w[i].split(' ')
w[i]=s[1]

mtr = np.eye(len(w))
for i in range(len(w)):
mtr[i][i]=0
stroka=""
for i in range(len(w)):
stroka=stroka+";"+w[i]

```

```

print(line)

for i in range(len(sent)):
    for j in range(len(w)):
        for k in range(len(w)):
            if(j!=k):
                if (re.search(w[j],sent[i])):
                    if (re.search(w[k],sent[i])):
                        mtr[k][j]=mtr[k][j]+1

for i in range(len(w)):
    stroka=w[i]+";"
    for j in range(len(w)):
        a=int(mtr[i][j])
        b=a.__str__()
        if (j<len(w)-1):
            stroka=stroka+b+";"
        otherwise:
            stroka=stroka+b
    print(line)

```

The result of the program is a file in the above format (Fig. 9.1).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		SAMSUNG GALAXY	NOTE	ULTRA	COMPANY IXBT	SMARTPHIFOLD	ELECTRON LINE	FLIP	IMAGES	MODEL				
2	SAMSUNG	0	2924	570	510	1008	570	1570	309	299	201	196	195	338
3	GALAXY	2924	0	684	571	379	327	1054	370	78	216	219	136	287
4	NOTE	570	684	0	183	65	68	249	61	22	59	27	31	67
5	ULTRA	510	571	183	0	47	56	148	21	8	20	17	37	66
6	COMPANY	1008	379	65	47	0	10	420	51	151	48	38	22	69
7	IXBT	570	327	68	56	10	0	115	36	8	4	23	9	10
8	SMARTPHI	1570	1054	249	148	420	115	0	163	58	133	93	73	132
9	FOLD	309	370	61	21	51	36	163	0	6	16	60	5	34
10	ELECTRON	299	78	22	8	151	8	58	6	0	9	2	8	5
11	LINE	201	216	59	20	48	4	133	16	9	0	9	8	38
12	FLIP	196	219	27	17	38	23	93	60	2	9	0	1	19
13	IMAGE	195	136	31	37	22	9	73	5	8	8	1	0	20
14	MODEL	338	287	67	66	69	10	132	34	5	38	19	20	0
15	SCREEN	430	277	52	55	97	27	216	49	9	20	33	50	54
16	DEVICE	335	175	45	12	115	8	128	23	24	17	6	9	21
17	BUDS	110	139	21	16	15	16	30	14	4	6	7	5	8
18	CAMERA	360	310	49	85	38	29	170	10	4	7	4	66	28
19	MARKET	160	44	10	6	80	10	88	13	5	6	5	8	11
20	ANDROID	137	86	21	7	18	20	87	1	4	4	4	1	6
21	PRO	158	128	55	39	19	19	54	6	3	4	5	11	23

Fig. 9.1 – The result of program execution is a file in CSV format

Basic information about the Gephi system

Gephi (<https://gephi.org/>) is currently the most popular program for visualization and analysis of networks and graphs ("network graphs"). Gephi provides fast layout, efficient filtering, and interactive data exploration, and is one of the best options for visualizing large-scale networks.

Gephi is a cross-platform open source software distributed under CDDL 1.0 and GNU General Public License v3. Mac OS X, Windows, and Linux versions of the source codes are available at <https://gephi.org/> (Fig. 9.2). The program requires Java version 7 and higher. Currently, the program is localized for the following languages: English, French, Portuguese, Russian, Chinese, Czech, and German.

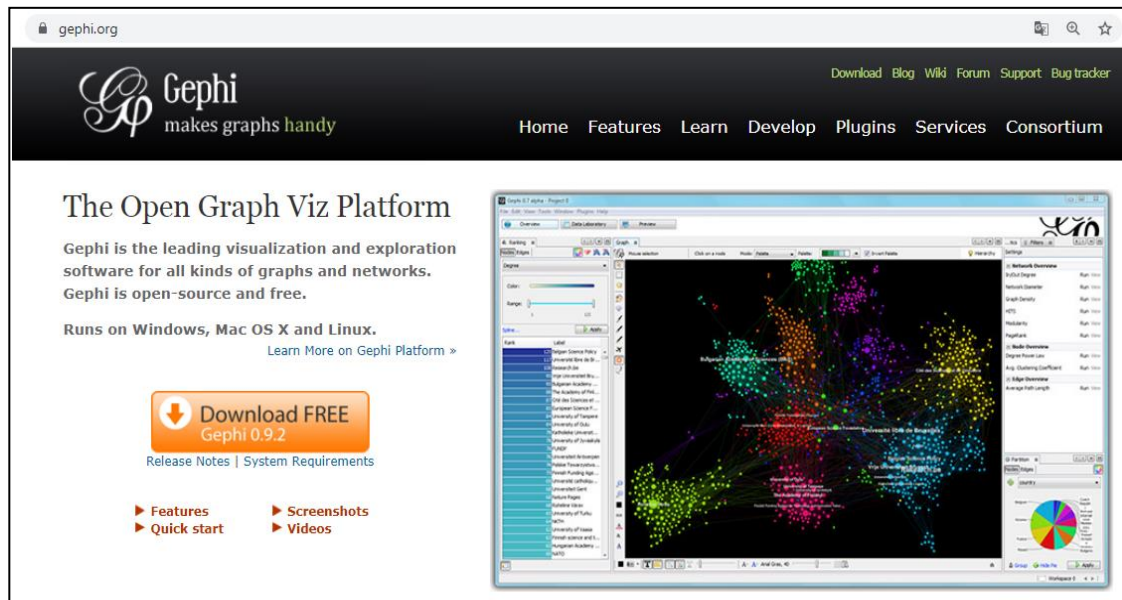


Fig. 9.2 – *Gephi* launch window – button to download the latest version

Gephi developers describe this program as "*Photoshop*, but for data".

Gephi allows you to download network data in *GEXF*, *GDF*, *GML*, *GraphML*, *Pajek (NET)*, *GraphViz (DOT)*, *CSV*, *UCINET (DL)*, *Tulip (TPL)*, *Netdraw (VNA)* and *Excel spreadsheet formats*. In addition, *Gephi* allows you to export network data in *JSON*, *CSV*, *Pajek (NET)*, *GUESS (GDF)*, *Gephi (GEXF)*, *GML* and *GraphML formats*. Thanks to this, *Gephi* can interact with other graph analysis and visualization systems.

The program includes a set of different layout algorithms (stacking graphs on a plane) and allows you to set colors, sizes and labels in graphs. *Gephi* is an interactive software and provides means to discover communities, and the ability to calculate shortest paths or the relative distance from any node to that node is also provided. Plugins from *Gephi* allow you to extend its functionality and add

new algorithms, layouts and measurement tools. *Gephi* has a multi-threaded data processing scheme, and thus allows several types of analysis to be performed simultaneously .

The user interface of the *Gephi system* includes three main sections (windows):

- "Data Lab": all raw data about networks, as well as additional calculated values, are stored here;
- "Data processing": a large part of user operations takes place here, in particular, manual editing of networks, testing layouts, setting filters;
- "Preview": here the form of the graph output is clarified, as a rule, the graph is improved, including from an aesthetic point of view, with the help of a set of tools. In the same window, the call to export the graph to *PDF*, *PNG* and *SVG formats is implemented* .

These three main sections contain multiple tabs that allow the user to implement individual functions.

Installing the Gephi system

To install the Gephi graph visualization and analysis system, you need to go to the system's website <http://gephi.org> and download the corresponding software (Download FREE button), then follow the standard steps for its installation.

Network visualization in the Gephi environment

When loading the adjacency matrix of concepts into the Gephi system (from CSV format), information about the number of nodes (actors) and the connections between them is displayed (Fig. 9.3) .

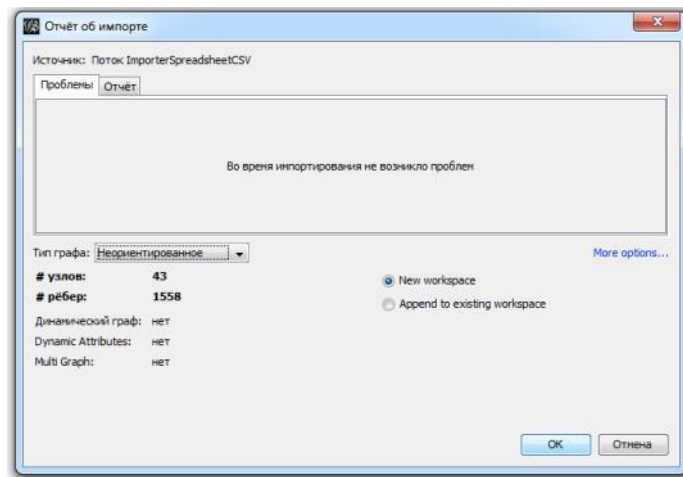


Fig. 9.3 – information on data import

After that, the file is opened, which in the "Processing" mode is displayed as a network in an unordered form. The initial display of the network is shown in Fig. 9.4.

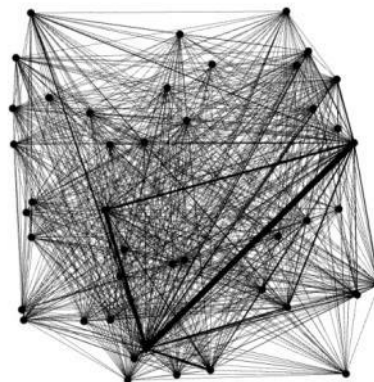


Fig. 9.4 – Initial display of the network

Next, the network is processed by arranging the nodes according to the OpenOrd algorithm, the size of the nodes - according to the degree, coloring - according to the clusters according to the modularity class (Fig. 9.5). In the "View" mode, the network of concepts has the following appearance, presented in Fig. 9.6.

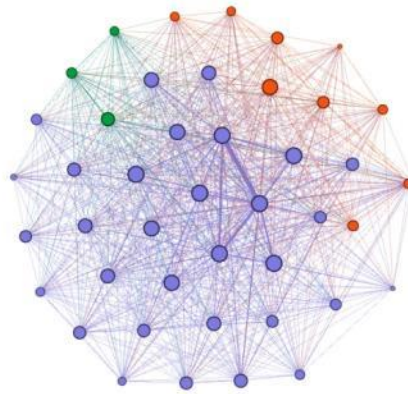


Fig. 9.5 - Visualization in "Processing" mode

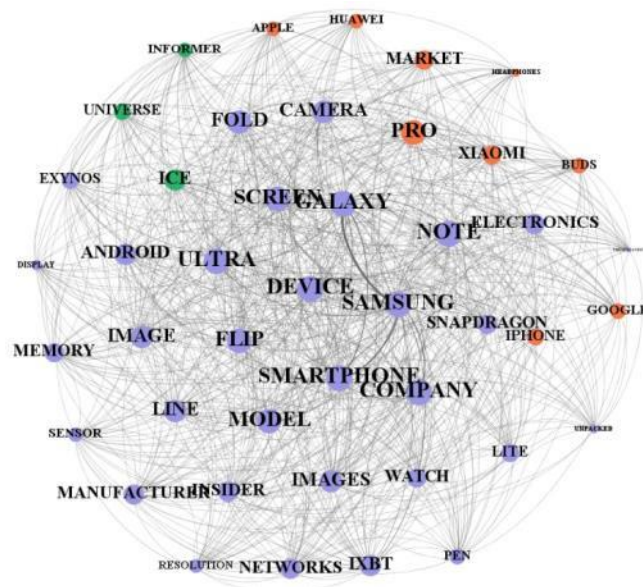


Fig. 9.6 - Visualization in "View" mode

Questions for practical work

1. What is a concept adjacency matrix?
2. Which ones do you know? means transformation types data in speech Python?
3. In which research projects is the Gephi package most widely used?
4. Name the main operating modes of the Gephi system.

Tasks for independent work

1. Learn more about data type conversion tools in Python.
2. Install the Gephi system on your computer yourself.
3. Familiarize yourself with the main modes and capabilities of the Gephi system.

literature

1. Ken Cherven. Mastering Gephi Network Visualization. - Packt Publishing, 2015. ISBN 78-1-78398-734-4.
2. John W. Foreman. Data Smart. Using Data Science to Transform Information into Insight. – Wiley, 2013. ISBN 111-8-66146-X, 978-1-11866-146-8.
3. Fundamentals of the theory and practice of intelligent data analysis in the field of cyber security: a study guide / D.V. Lande, I.Yu. Subach, Yu.E. Boyarinova. – Kyiv: ISZZI KPI named after Igor Sikorskyi, 2018. – 300 p.
4. Lande D.V., Subach I.Yu. Visualization and analysis of network structures: tutorial. – Kyiv: KPI named after Igor Sikorskyi, Polytechnic Publishing House, 2021. – 80 p. ISBN 978-966-2577-14-3

10. Apache Hadoop

The purpose of the practical work corresponding to the section is to study the principles of Hadoop - a powerful tool for working with large data from the Apache foundation, obtained from the Internet (processing of logical records of remote servers). Consider examples of the development of MapReduce programs (Google, 2004) under Hadoop.

Apache Hadoop is a software platform and basis for the organization of distributed storage and processing of extremely large data sets using the MapReduce model, in which the task is divided into smaller, separate fragments, each of which can be run on a separate node of a cluster consisting of separate servers. All modules in Hadoop are designed taking into account that hardware can fail and this should be handled automatically by the framework.

The core of the Apache Hadoop system consists of the Hadoop Distributed Filesystem (HDFS) and a computing system based on the MapReduce programming model.

A Hadoop cluster consists of many machines that store and process large data sets in parallel. Client computers send tasks to this computing cloud and receive results.

Hadoop is an open source framework for building and running distributed applications that process large amounts of data. Distributed computing is a broad and multifaceted field, but Hadoop has a number of important distinguishing features, namely:

- Availability – Hadoop runs on large clusters assembled from off-the-shelf computers or in the cloud.
- Reliability – Because Hadoop must run on off-the-shelf hardware, its architecture is designed with the possibility of frequent failures in mind.
- Scalability – Hadoop scales linearly, i.e. as the volume of data increases, it is sufficient to add new nodes to the cluster.

- Simplicity – Hadoop allows the user to quickly create efficient parallel code. Hadoop's affordability and simplicity give it a competitive edge when it comes to writing and running large distributed applications. On the other hand, reliability and scalability make Hadoop a suitable tool even for critical tasks.

Core components of Hadoop

The Apache Hadoop framework consists of the following modules:

- Hadoop Distributed File System (HDFS) is a distributed file system that allows you to store almost unlimited amounts of information.
- Hadoop YARN is a framework for managing cluster resources and task management, including the MapReduce framework.
- Hadoop Common - Linking Software - A set of infrastructure software libraries and utilities used by other modules and family projects

There are also a large number of projects directly related to Hadoop, but not included in Hadoop core:

- Hive – a tool for SQL-like queries on big data (turns SQL queries into a series of MapReduce tasks);
- Pig is a programming language for high-level data analysis. One line of code in this language can turn into a sequence of MapReduce tasks;
- Hbase – a columnar database that implements the BigTable paradigm;
- Cassandra is a high-performance distributed key-value database;
- ZooKeeper is a service for distributed configuration storage and synchronization of changes to this configuration;
- Mahout is a library and engine for machine learning on big data.

The Apache Spark project is a tool for distributed data processing. Apache Spark typically uses Hadoop components such as HDFS and YARN for its work.

Apache Hadoop and the Hadoop ecosystem

Although Hadoop is most often associated with MapReduce and the Distributed File System (HDFS, formerly NDFS), the term often refers to a family of interrelated projects united by a distributed computing and large-scale data processing infrastructure.

All underlying projects are maintained by the Apache Software Foundation, which provides community support for open source projects - including the original HTTP server from which the name derives.

As the Hadoop ecosystem expands, new projects appear that are not necessarily Apache-powered, but provide additional Hadoop functionality or build higher-level abstractions from the underlying functionality.

Installing Hadoop on a cluster using Cloudera Manager

Previously, installing Hadoop was a rather difficult task - you had to configure each machine in the cluster separately, make sure that nothing was forgotten, and carefully configure monitoring. As Hadoop has grown in popularity, companies (such as Cloudera, Hortonworks, MapR) have emerged that provide their own Hadoop builds and powerful Hadoop cluster management tools. As part of this work, we will use the Hadoop assembly from the Cloudera company.

Cloudera Hadoop consists of:

- Cloudera Hadoop (CDH) is the Hadoop distribution itself;
- Cloudera Manager is a tool for deploying, monitoring and managing a Hadoop cluster.

In order to install Hadoop on your cluster, you need to perform a few simple steps:

1. Download Cloudera Manager Express to one of the machines in your cluster from here - <https://www.cloudera.com/downloads/cdp-private-cloud-trial.html>;

2. Assign execution rights and launch;
3. Follow the installation instructions.

The cluster must run on one of the supported operating systems of the Linux family: RHEL, Oracle Enterprise Linux, SLES, Debian, Ubuntu.

After installation, the cluster management console is provided, where you can view the installed services, add/remove services, monitor the cluster status, and edit the cluster configuration (Fig. 10.1).

more detail, the process of installing Hadoop on a cluster using Cloudera Manager can be found at the link in the Quick Start section (https://docs.cloudera.com/documentation/enterprise/5-2-x/topics/cloudera_quickstart_vm.html).

In the case of familiarization and practical work, it is enough to simply download a pre-configured virtual machine and use the configured software.

Deploying a distributed environment based on Apache Hadoop

Before starting work, it is necessary to familiarize yourself with existing implementations of distributed environments (Apache Hadoop). Learn to deploy and use distributed environments to solve resource-intensive tasks.

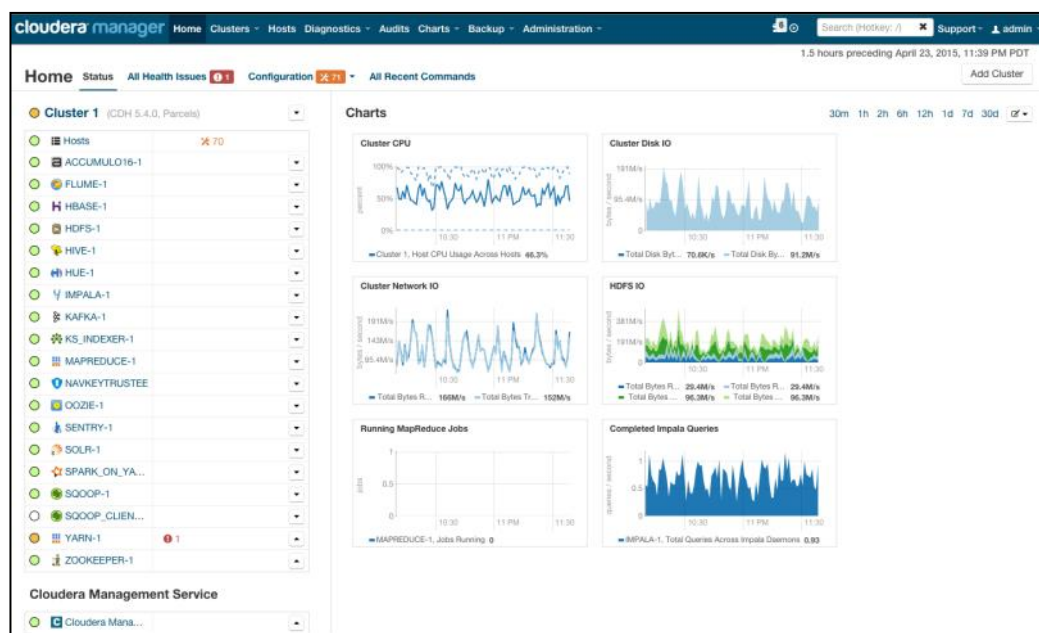


Fig. 10.1 – Cluster management console

Apache Hadoop is a project that was once part of the open search engine Nutch, but later separated from it as an implementation of the distributed file system (Distributed File System) and the MapReduce paradigm. Hadoop consists of two modules. It is an implementation of the paradigm of a distributed file system (called HDFS) and MapReduce, that is, an implementation of the MapReduce framework.

Distributed File System (DFS). Main properties: storage of large volumes of data; transparency, work as with a regular file system - with the ability to open files and not think about the implementation of distributed storage; scalability – the ability to add new nodes for data storage; reliability - the failure of a certain percentage of nodes should not damage the integrity of the entire system. DFS uses a cluster consisting of a controlling master node and slave nodes that store data. A file table is stored on the master node. Each file is divided into blocks. The master node stores information on which specific cluster machines contain file blocks. When reading/writing data, the wizard provides information about the location of the file blocks: on which machine specific blocks are stored and their serial number. To ensure reliability, each block is stored in multiple copies, on multiple machines.

The order of work

Setting up a single-node cluster.

1. Repository Index Update:

```
$ sudo apt-get update
```

2. Installing Java:

```
$ sudo apt-get install default-jdk $ update-alternatives --config java
```

Java version must be at least 1.7.

Audit:

```
$ java -version
```

3. Installing SSH:

```
$ sudo apt-get install ssh
```

4. Installing RSYNC:

```
$ sudo apt-get install rsync
```

5. Key generation for SSH:

```
$ ssh-keygen -t dsa -P ' ' -f ~/.ssh/id_dsa $ cat ~/.ssh/id_dsa.pub >>  
~/.ssh/authorized_keys
```

6. Download Hadoop (TAR.GZ archive) from the official website.

7. Unpacking Hadoop:

```
$ sudo tar -zxvf hadoop-.tar.gz
```

8. Adding environment variables to ~/.bashrc

```
#Hadoop Variables
```

```
export JAVA_HOME=<path to JAVA directory>
```

```
export HADOOP_HOME=<path to unpacked Hadoop directory>
```

```
export PATH=$PATH:$HADOOP_HOME/bin
```

```
export PATH=$PATH:$HADOOP_HOME/sbin
```

```
export HADOOP_MAPRED_HOME=$HADOOP_HOME
```

```
export HADOOP_COMMON_HOME=$HADOOP_HOME
```

```
export HADOOP_HDFS_HOME=$HADOOP_HOME
```

```
export YARN_HOME=$HADOOP_HOME
```

```
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
```

```
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

9. Updating the configuration after making changes:

```
$ source ~/.bashrc
```

10. In the hadoop directory, edit hadoop-env.sh

```
export JAVA_HOME=<path to Java directory>
```

11. Editing core-site.xml

```
<configuration>  
<property>  
<name>fs.defaultFS</name>  
<value>hdfs://localhost:9000</value>  
</property>  
</configuration>
```

12. Editing yarn-site.xml

```
<configuration>  
<property>  
<name>yarn.nodemanager.aux-services</name>  
<value>mapreduce_shuffle</value>  
</property>  
<property>  
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>  
<value>org.apache.hadoop.mapred.ShuffleHandler</value>  
</property>  
</configuration>
```

13. Copying the template:

```
$ sudo cp mapred.site.xml.template mapred-site.xml
```

14. Editing the template (mapred-site.xml):

```
<configuration>  
<property>  
<name>mapreduce.framework.name</name>  
<value>yarn</value>  
</property>  
</configuration>
```

15. Editing hdfs-site.xml

```
<configuration>  
<property>  
<name>dfs.replication</name>  
<value>1</value>  
</property>  
<property>  
<name>dfs.namenode.name.dir</name>  
<value>  
file:<Hadoop directory>/hadoop_data/hdfs/namenode  
</value>  
</property>  
<property>  
<name>dfs.datanode.data.dir</name>  
<value>  
file: <Hadoop directory>/hadoop_store/hdfs/datanode  
</value>  
</property>
```

```
</configuration>
```

16. Creating node files (nodes):

```
$ mkdir -p <Hadoop directory>/hadoop_data/hdfs/namenode
```

```
$ mkdir -p <Hadoop directory>/hadoop_data/hdfs/datanode
```

17. Changing the current user and directory group from Hadoop:

```
$ sudo chown <user>:<group> -R <hadoop directory>
```

18. Formatting and Launching

```
$ hdfs namenode -format
```

```
$ start-all.sh
```

19. Status check:

```
$ jps
```

After you download Hadoop, you can manage your Hadoop cluster in one of three supported modes:

Local/Offline Mode – After Hadoop is loaded by default, it is configured in offline mode and can run as a separate Java process.

Pseudo-distributed mode is a distributed simulation on a single machine. Each Hadoop daemon, such as hdfs, yarn, MapReduce, etc., will run as a separate Java process. This mode is useful for development.

Fully Distributed Mode - This mode is fully distributed with a minimum of two or more computers as a cluster.

Installing Hadoop offline

When installing Hadoop 2.4.1 offline, no daemons are started and everything runs in a single Java virtual machine. Offline mode is suitable for running MapReduce applications during development because they are easy to test and debug.

To configure Hadoop, you can set the Hadoop environment variables by adding the following commands to the `~/.bashrc` file:

```
export HADOOP_HOME=/usr/local/hadoop
```

You can verify that Hadoop is working fine. Just type the following command:

```
$ hadoop version
```

If everything is fine with the settings, you should see the result (with version accuracy):

```
Hadoop 2.4.1  
Subversion https://svn.apache.org/repos/asf/hadoop/common -r 1529768  
Compiled by hortomu on 2013-10-07T06:28Z  
Compiled with protocol 2.5.0  
From source with checksum 79e53ce7994d1628b240f09af91e1af4
```

This means that Hadoop's offline mode setup is working fine. By default, Hadoop is configured to run in non-distributed mode on a single machine.

Questions for practical work.

1. What is Apache Hadoop Big Data Platform?
2. What are the main components (protocols and specifications) that make up the Apache Hadoop stack?
3. Why can extremely large data sets be processed in blocks by the MapReduce processing algorithm?

Tasks for independent work

1. Install the Apache Hadoop platform on the computer and study the composition of its components and the values of its main parameters.
2. Check the performance of Hadoop.
3. Familiarize yourself with the working procedure of the MapReduce big data processing engine.

literature

1. Andrew S. Tanenbaum, Maarten van Steen. Distributed Systems: Principles and Paradigms. - Upper Saddle River, NJ: Pearson Prentice Hall, 2nd edition, 2006. - 686 p.
2. Nancy A. Lynch. Distributed Algorithms. - San Francisco, Calif.: Morgan Kaufmann Publishers, 1996. - 872 p.
3. Ajay D. Kshemkalyani, Mukesh Singhal. Distributed Computing: Principles, Algorithms and Systems. - Cambridge University Press, 2008. - 736 p.
4. Mikito Takada. Distributed Systems for Fun and Profit (Electronic resource <http://book.mixu.net/distsys/ebook.html>).
5. Apache Hadoop (Electronic resource <http://hadoop.apache.org/>).
6. Torque Resource Manager (Electronic resource <http://www.adaptivecomputing.com/products/open-source/torque/>).

11. MapReduce programming

The purpose of the practical work related to the section is to learn how to program simple MapReduce applications for practical problems and chaining simple problems using Python commands and a simple structure of key-value data stores, to learn the principles of processing text documents (language corpora, csv server logs) using the Hadoop tool and Python language commands to write short processing programs.

MapReduce is a concept of distributed computing. MapReduce operation consists of two main steps: Map and Reduce. In the first step (Map), the input data is pre-processed. For this, one of the computers (master-node) receives the input data of the task, divides them into parts and transmits them to other computers (worker-nodes). In the second step (Reduce) there is aggregation of pre-processed data in the Map step. The main node receives responses from working nodes and based on them forms a result - a solution to the problem.

Programming the WordCount problem

Let's consider how to run a MapReduce task in Python on Hadoop. As a task, we will use the classic example of WordCount. In order to experiment with real data, you need to use a prepared news archive.

Formulation of the task: there is a set of documents. It is necessary for each word found in the set of documents to count how many times this word occurs in the archive on the server (lenta_articles.tar.gz).

Let's consider the solution of the problem. If we have a large corpus of documents - let one document be one input record for the MapReduce task. In MapReduce, we can only specify user functions, which we will do (we will use python-like pseudocode). Map splits the document into words and returns a set of pairs of the form (word, 1).

```
def map(doc):  
    for word in doc:
```

```
yield word, 1
```

Reduce summarizes the occurrence of each word:

```
def reduce(word, values):  
    yield word, sum(values)
```

Now the task is to program this solution as code that can be executed on Hadoop and run.

The easiest way to run a MapReduce program on Hadoop is to use Hadoop's streaming interface. The Streaming interface assumes that Map and Reduce are implemented as programs that accept data from stdin and output the result to stdout.

The program that performs the Map function is called Mapper. The program that executes Reduce is called, accordingly, Reducer.

The Streaming interface assumes by default that one line included in the Mapper or Reducer corresponds to one record included in the Map.

The conclusion of the Mapper enters the input of the Reducer in the form of pairs (key, value), with all pairs corresponding to the same key:

- Guaranteed to be processed by one run of Reducer;
- They will be submitted to the input consecutively (that is, if one Reducer processes several different keys, the input will be grouped by key).

So, let's implement the Mapper and Reducer in the Python programming language:

```
#mapper.py  
  
import sys  
  
def do_map(doc):  
    for word in doc.split():  
        yield word.lower(), 1  
  
    for line in sys.stdin:  
        for key, value in do_map(line):  
            print(key + "\t" + str(value))
```

```

#reducer.py

import sys

def do_reduce(word, values):

return word, sum(values)

prev_key = None

values = []

for line in sys.stdin:

key, value = line.split("\t")

if key != prev_key and prev_key is not None:

result_key, result_value = do_reduce(prev_key, values)

print(result_key + "\t" + str(result_value))

values = []

prev_key = key

values.append(int(value))

if prev_key is not None:

result_key, result_value = do_reduce(prev_key, values)

print(result_key + "\t" + str(result_value))

```

The data that will be processed by Hadoop must be stored on HDFS. Let's download the job description and put it in HDFS. To do this, you need to use the `hadoop fs` command:

```

wget https://www.dropbox.com/s/opp5psid1x3jt41/lenta_articles.tar.gz

tar xzvf lenta_articles.tar.gz

hadoop fs -put lenta_articles

```

Hadoop fs utility supports a large number of methods for manipulating the file system, many of which replicate standard linux utilities one-to-one. You can learn more about its capabilities at the link - <https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html>.

Now let's run the streaming task:

```
yarn jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar\  
  
-input lenta_articles\  
  
-output lenta_wordcount\  
  
-file mapper.py\  
  
-file reducer.py\  
  
-mapper "python mapper.py"\  
  
-reducer "python reducer.py"
```

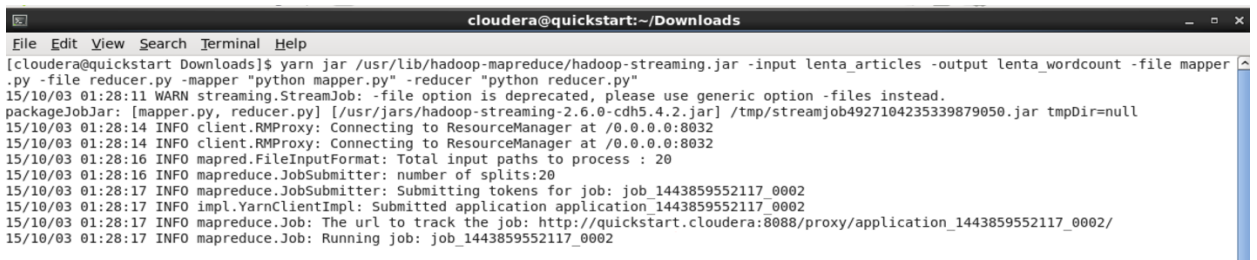
The yarn utility serves to launch and manage various applications (including map-reduce based) on the cluster. Hadoop-streaming.jar is just one example of such a yarn application.

Next are the launch options:

- input – folder with output data on hdfs;
- output – a folder on hdfs where you need to put the result;
- file – files that are needed during the map-reduce task;
- mapper – the console command that will be used for the map stage;
- reduce - a console command that will be used for the R educe stage.

After starting, the console will show the progress of the task and the URL for viewing more detailed information about the task (Fig. 11.1).

In the interface available from this URL, you can get a more detailed status of the task execution, view the logs of each M apper and R educe r (Fig. 11.2).



```
cloudera@quickstart:~/Downloads  
File Edit View Search Terminal Help  
[cloudera@quickstart Downloads]$ yarn jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar -input lenta_articles -output lenta_wordcount -file mapper  
.py -file reducer.py -mapper "python mapper.py" -reducer "python reducer.py"  
15/10/03 01:28:11 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.  
packageJobJar: [mapper.py, reducer.py] [/usr/jars/hadoop-streaming-2.6.0-cdh5.4.2.jar] /tmp/streamjob4927104235339879050.jar tmpDir=null  
15/10/03 01:28:14 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032  
15/10/03 01:28:14 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032  
15/10/03 01:28:16 INFO mapred.FileInputFormat: Total input paths to process : 20  
15/10/03 01:28:16 INFO mapreduce.JobSubmitter: number of splits:20  
15/10/03 01:28:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1443859552117_0002  
15/10/03 01:28:17 INFO impl.YarnClientImpl: Submitted application application_1443859552117_0002  
15/10/03 01:28:17 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1443859552117_0002/  
15/10/03 01:28:17 INFO mapreduce.Job: Running job: job_1443859552117_0002
```

Fig. 11.1 - Task execution protocol

Job Overview			
Job Name:	streamjob4927104235339879050.jar		
State:	RUNNING		
Uberized:	false		
Started:	Sat Oct 03 01:28:28 PDT 2015		
Elapsed:	1mins, 18sec		

ApplicationMaster			
Attempt Number	Start Time	Node	Logs
1	Sat Oct 03 01:28:20 PDT 2015	quickstart.cloudera:8042	logs

Task Type	Progress	Total	Pending	Running	Complete
Map	<div style="width: 20%;"></div>	20	9	5	6
Reduce	<div style="width: 1%;"></div>	1	0	1	0

Attempt Type	New	Running	Failed	Killed	Successful
Maps	9	5	0	0	6
Reduces	0	1	0	0	0

Fig. 11.2 – Task status

The result of the work after successful execution is stored in HDFS in the folder that we specified in the output field. You can view its contents using the "hadoop fs -ls lenta_wordcount" command.

The result itself can be obtained as follows:

```
hadoop fs -text lenta_wordcount/* | sort -n -k2,2 | tail -n5
p.      41
that   43
on     82
and   111
in    194
```

The "hadoop fs -text" command outputs the contents of the folder in text form. We sorted the result by the number of word occurrences. As expected, the most frequent words in the language are prepositions.

Map only job

In fact, there are a number of tasks in which you can do without only the Map stage. Here are examples of such tasks:

1. Data filtering (for example, "Find all entries from IP address 123.123.123.123" in web server logs);
2. Data conversion ("Delete column in csv-logs");
3. Loading and unloading of data from an external source ("Insert all entries from the log into the database").

Such tasks are solved using the Map-Only method. When creating a Map-Only task in Hadoop, you need to specify zero number of R educers:

In Hadoop, you can define the Combine function for this, that is, it will process the output of part of the M appers. The combining function is very similar to the Reduce function - it accepts the output of a part of M apper's as an input and gives an aggregated result for these M apper's, so very often R educer is also used as a combiner. Its important difference from Reduce is that the Combine function receives all the values corresponding to one key.

Moreover, Hadoop does not guarantee that the combine function will be performed at all for M apper's output. Because the combining function cannot always be applied, for example, in the case of searching for the median value by key. However, in those tasks where the combining function can be applied, its use allows to achieve a significant increase in the speed of execution of the MapReduce task.

The most difficult stage when performing a Map-Reduce task is the Shuffle stage. This happens because the intermediate results (M apper's outputs) are written to disk, organized and transmitted over the network. However, there are tasks in which such behavior does not seem very reasonable. For example, in the task of counting words in documents, it is possible to pre-aggregate the output results of several M apper's on one node of the map-reduce task, and transfer the already summed values for each machine to the R educer.

Chains of MapReduce tasks

There are situations when MapReduce alone cannot be used to solve a problem. For example, consider a slightly modified WordCount task: there is a set of text documents, you need to count how many words are found from 1 to 1000 times in the set, how many words from 1001 to 2000, how many from 2001 to 3000, and so on. To solve this task, we will need 2 MapReduce tasks:

1. Modified Word Count, which for each word will calculate which of the intervals it falls into ;
2. MapReduce that counts how many times each of the intervals was encountered in the output of the first MapReduce.

Pseudocode solution:

```
#map1
def map(doc):
    for word in doc:
        yield word, 1

#reduce1
def reduce(word, values):
    yield int(sum(values)/1000), 1

#map2
def map(doc):
    interval, cnt = doc.split()
    yield interval, cnt

#reduce2
def reduce(interval, values):
    yield interval*1000, sum(values)
```

In order to execute a sequence of MapReduce tasks on hadoop, it is quite simple to specify the folder that was specified as output for the first task as input data for the second task and run them one by one.

In practice, chains of MapReduce tasks can be quite complex sequences in which MapReduce tasks can be connected both sequentially and in parallel to each other (Fig. 11.3). To simplify the management of such job plans, there are separate tools such as the oozie scheduler (<http://oozie.apache.org/>) and luigi (a Python package that helps create complex batch job pipelines).

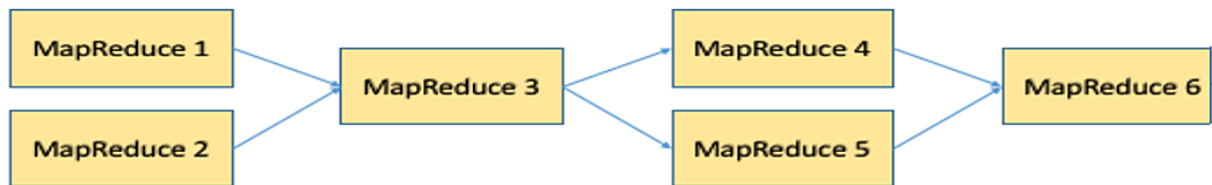


Fig. 11.3 – Example of a chain of MapReduce tasks.

Distributed cache

An important mechanism in Hadoop is Distributed Cache. Distributed Cache allows you to add files (for example, text files, archives, jar files) to the environment in which the MapReduce job is executed.

You can add files stored on HDFS, local files (local to the machine from which the task is run). Distributed Cache can be used together with Hadoop streaming: by adding the mapper.py and reducer.py files via the -file option. You can add not only mapper.py and reducer.py, but any files in general, and then use them as if they were in a local folder.

Reduce Join

When working with relational databases, the Join operation is often used, which allows you to jointly process the contents of some tables by combining them by some key. When working with big data, such a task also sometimes arises. Consider the following example:

Consider an example when there are logs of two web servers, each of which has the following form:

```

1446792139      178.78.82.1/sphingosine/unhurrying.css
1446792139      126.31.163.222/accidentally.js
1446792139      154.164.149.83/pyroacid/unkemptly.jpg
1446792139      202.27.13.181/Chawia.js
1446792139      67.123.248.174/morphographical/dismain.css
1446792139      226.74.123.135/phanerite.php
1446792139      157.109.106.104/bisonant.css
  
```

It is necessary to calculate for each IP address which of the 2 servers it visited more often. The result should be presented in the form:

```
178.78.82.1    first
126.31.163.222 second
154.164.149.83 second
226.74.123.135 first
```

Unlike relational databases, in general, joining two logs by key (by IP address) is a rather complex operation and is solved using 3 MapReduce and the Reduce Join pattern (Fig. 11.4).

ReduceJoin works as follows:

A separate MapReduce (Map only) is launched for each of the input logs, which transforms the input data into the following form:

`key -> (type, value),`

where key is the key by which you need to join the tables, type is the table type (first or second in our case), and value is any additional data tied to the key.

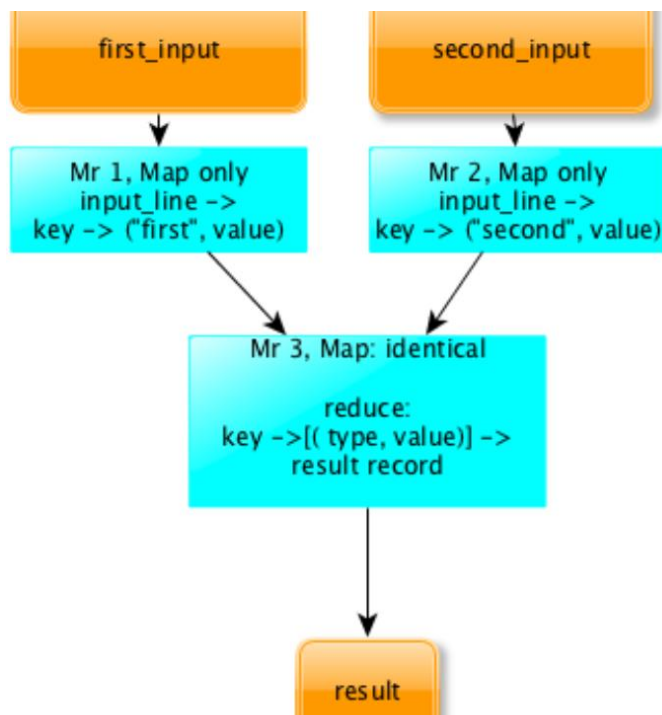


Fig. 11.4 – General scheme of ReduceJoin

The outputs of both MapReduce are fed to the input of the 3rd MapReduce, which, in fact, performs the union. This MapReduce contains an empty Mapper that simply copies the input data. Next, Shuffle decomposes the data by keys and submits it to the Reducer input in the form:

```
key -> [(type, value)]
```

It is important that at this moment records from both logs are received by the Reducer, and at the same time, by the type field, it is possible to identify from which of the two logs a specific value was received. So the data is enough to solve the original problem. In our case, the Reducer should simply count for each key the records with which the type met more and output this type.

MapJoin

The ReduceJoin pattern describes the general case of joining two logs by key. However, there is a special case in which the task can be significantly simplified and accelerated. This is a case in which one of the logs is significantly smaller than the other.

Consider the following problem:

There are 2 logs. The first log of the web server (the same as in the previous task), the second file (100 KB in size) contains the correspondence of the →Topic URL.

Example of the 2nd file:

```
/toyota.php auto  
/football/spartak.html sport  
/ Cars auto  
/Finances/money business
```

For each IP address, it is necessary to calculate the pages of which category were downloaded most often from this IP address.

In this case, we also need to join 2 logs by URL. However, in this case, we do not need to run 3 MapReduce, since the second log will fit completely in memory.

In order to solve the problem using the 1st MapReduce, we can load the second log into the Distributed Cache, and when initializing Mapper, simply read it into memory by putting it in the \rightarrow topic dictionary (Fig. 11.5) .

Then the task is solved as follows:

Maps:

```
# We find the topic of each of the pages of the first log
input_line -> [ip, topic]
```

Reduce:

```
ip -> [topics] -> [ip, most_popular_topic]
```

Reduce receives an ip input and a list of all topics, it simply calculates which of the topics was encountered most often. In this way, the problem is solved with the help of the 1st MapReduce, and the actual Join generally takes place inside the Map (therefore, if additional aggregation by key was not needed, it would be possible to do without the MapOnly task).

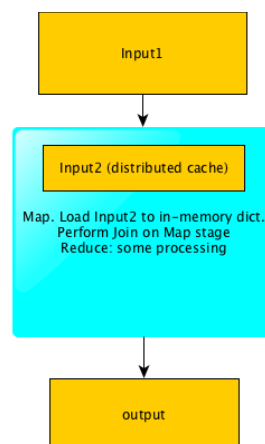


Fig. 11.5 – MapJoin operation scheme

In this work, we considered Hadoop - a software stack for working with big data, described the Hadoop installation process using the Slouder distribution as an example, showed how to write mapreduce programs using the streaming interface and Hadoop's API.

Questions for practical work:

1. What is the essence of the Map task in the MapReduce algorithm?
2. What is the essence of the Reduce task in the MapReduce algorithm?
3. How many stages does the MapReduce algorithm perform when processing Big Data?
4. What is the main task of the split function in Hadoop when organizing Big Data processing for MapReduce jobs?
5. What are the most common applications of MapReduce for processing *Big Data* that you know?

Tasks for independent work

1. Using programming patterns in MapReduce, create a text file processing program.
2. Develop a log processing program using programming patterns in MapReduce.
3. Familiarize yourself with the principles of the MapReduce big data processing algorithm.
4. Create a Hadoop big data processing environment on your own computer using virtualization technology.

Literature

1. Thomas Earle, Wajid Khattak, Paul Buhler Fundamentals of Big Data: Concepts, Algorithms and Technologies/ Trans. with English, K.: Publishing House Business, Balance, Books - BBB, 2016.-342p.
2. Nathan Martz, James Warren Big Data: Principles and Practice of Building Scalable Data Processing Systems in Real Time: Trans. with English - Kyiv. "Williams". 2016. - 308 p.
3. Tom White Hadoop. Detailed instructions. Trans. with English St. Petersburg "Peter". 2013. - 672 p.

4. Hadoop technology. Practical experience and expertise DISgroup, 2012.
(URL: <http://www.dis-group.ru>)
5. Chuck Lam. Hadoop in action. - M.: DMK Press, 2012. - 424 p. ISBN 978-5-94074-785-7
6. <https://habr.com/ru/company/dca/blog/268277/>
7. <https://habr.com/ru/post/270453/>
8. <https://habr.com/ru/post/345672/>

12. DBMS MongoDB

The purpose of the practical work corresponding to the section is to gain experience working with the NoSQL DBMS MongoDB, installing the server and the line client of this system, working in command line mode, uploading external files, in particular, in CSV format to the databases of this DBMS, as well as exporting data to external files.

NoSQL technologies are replacing long-known relational databases. Unlike relational databases, NoSQL DBMSs offer document-oriented data models, thanks to which many of these DBMSs work faster, have better scalability, and are easier to use.

MongoDB, one of the NoSQL databases, implements a new approach to building databases, where there are no tables, schemas, SQL queries, foreign keys, and many other things inherent in object-relational databases.

MongoDB is not a relational, but a document-oriented database management system. The main reason for abandoning the relational model is to simplify horizontal scaling, but there are other advantages. MongoDB is written in C++, so it is easy to port to different platforms.

MongoDB can be deployed on Windows, Linux, MacOS, Solaris platforms.

The six main concepts of MongoDB are:

1. MongoDB is conceptually the same as an ordinary, familiar database. Inside MongoDB, there can be zero or more databases, each of which is a container for other entities.

2. A database can have zero or more collections. The collection is so similar to a traditional "table" that you can safely consider them one and the same.

3. Collections consist of zero or more "documents". Again, the document can be thought of as a "string".

4. The document consists of one or more "fields", which are similar to a "column".

5. Indexes in MongoDB are almost identical to indexes in relational databases.

6. "Cursors" are different from the previous five concepts, but they are very important (although sometimes they are overlooked) and deserve a separate discussion.

Installation of the server and the line client

To install the MongoDB server, you need:

1. Go to the official download web page (Fig. 12.1) and download the binary file (recommended stable version). Both 32-bit and 64-bit versions can be used.

2. Unpack (anywhere) and go to the bin folder. Two executables to work with: `mongod` is the server and `mongo` is the client console.

3. Create a new file in the bin folder and name it `mongodb.config`

4. Add one line to `mongodb.config`: `dbpath = <Path to database files> .`

For example, in Windows you can write `dbpath=c:\mongodb\data` and in Linux - `dbpath=/etc/mongodb/data .`

5. Start `mongod` with the parameter `--config /path/to/your/mongodb.config .`

Some commands of the string client

To start the MongoDB string client, it is enough to run the command:

```
$ mongo
```

After that, the MongoDB database will respond with information about the version, server address, ID number, date, time, etc.:

```
MongoDB shell version v4.4.6
connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("f2faf2a9-87ac-4d71-91cc-a77d00275e4a") }
MongoDB server version: 4.4.5
```

```
The server generated these startup warnings when booting:
2021-05-28T12:14:01.468+00:00: Using the XFS file system is strongly recommended with
the WiredTiger storage engine.
```

```
>
```

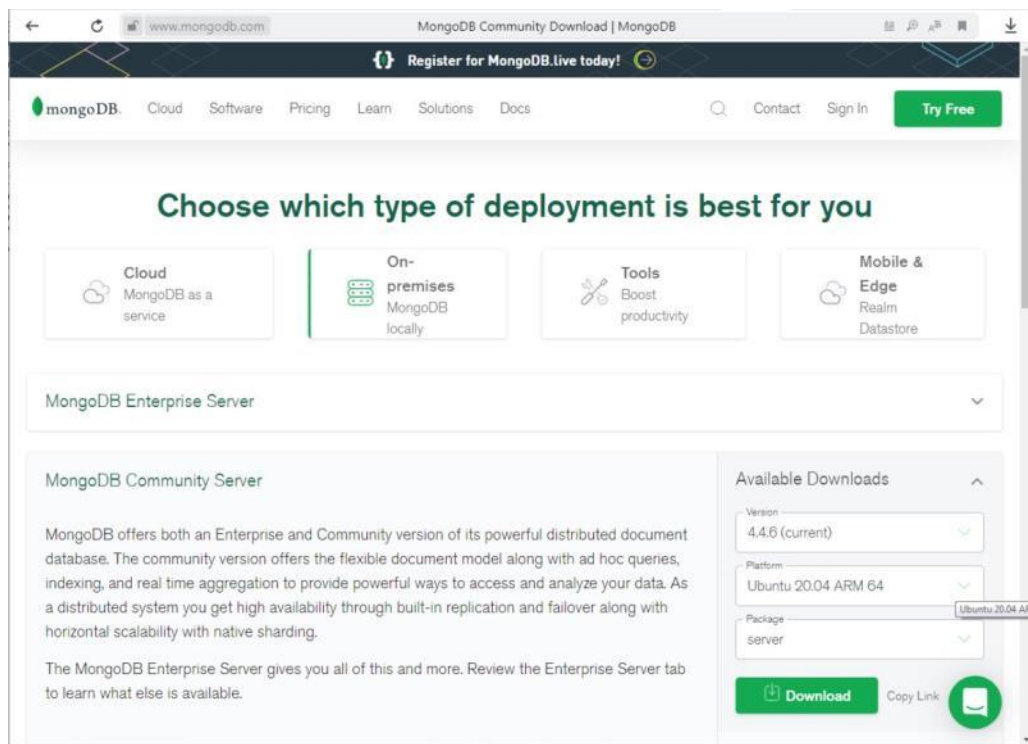


Fig. 12.1 - MongoDB Home Page (<https://www.mongodb.com>)

To get a list of active databases, enter the following command:

```
>db.getMongo().getDBNames()
```

OR

```
>db.adminCommand('listDatabases')
```

To connect to the database, enter the following command:

```
>use <database name>
```

For example, when the name of the database is "sources" (sources, for example, social network accounts for further monitoring):

```
> use sources
```

To get a list of collections in the database, enter the following command:

```
>db.getCollectionNames()
```

The result will look like this (example):

```
[ "tg", "ytc" ]
```

To get the number of documents in the collection (for example, the name of the ytc collection), you need to enter the command:

```
>db.ytc.find().count()
```

To view the content of all documents in the collection (for example, the name of the ytc collection), it is enough to enter the search command without the query parameter:

```
>db.ytc.find()
```

The result will look like this (example):

```
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e79"), "saddr" : "UCopcQiBf6yqBPSNFz9x4OYw", "name" : "Klimenko Time"
}
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e7a"), "saddr" : "UCQwVj4PyS5leCgEJY4I2t1Q", "name" : "DW Ukr" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e7b"), "saddr" : "UCXoAjrdHFazhEL3Ug8REClw", "name" : "DW Rus" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e7c"), "saddr" : "UCD7dXP0dxuf2b34lBcFb4IA", "name" : "TIS TV" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e7d"), "saddr" : "UCsxd-Drk7iYG50Ly-LVNWaw", "name" : "REN TV" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e81"), "saddr" : "UCdRzHJ8wUmDCr0wYM2Zi07Q", "name" : "Kryminform" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e82"), "saddr" : "UCw5pE2GE1cHeErKVyF76jJg", "name" : "TV Center" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e83"), "saddr" : "UCUjpfPulcVwbWD8yyCe5Zew", "name" : "Factorium Room"
}
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e84"), "saddr" : "UCJ9e_x5_ZuR9TwZwzvqRsIA", "name" : "Rogandar News"
}
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e85"), "saddr" : "UCPMf85Kr0Xtlr3bIYXdsZ3w", "name" : "RGD News" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e86"), "saddr" : "UCgS19QtJ5NqQi_ErxYVoIpA", "name" : "News 7" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e87"), "saddr" : "UCbJYQHINGGcuoDYKIQZ816A", "name" : "Gleb Volyn" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e88"), "saddr" : "UCBi2mrWuNuyYy4gbM6fU18Q", "name" : "ABC News" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e89"), "saddr" : "UC7aFhtshxOnaqioFAJ0ZSvA", "name" : "Vovan222prank"
}
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e8a"), "saddr" : "UC6ZFN9Tx6xh-skXCuRHCDpQ", "name" : "PBS NewsHour" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e8b"), "saddr" : "UCupvZG-5ko_eiXAupbDfxWw", "name" : "CNN" }
{ "_id" : ObjectId("60cb88b9b8ca2f6093347e8c"), "saddr" : "UCvjqjOZN3ZvDEmZfHuUKCIQ", "name" : "CGTN Live"
}
```

Type "it" for more

If we just want to see one document from the collection, we can use the findOne function:

```
> db.movies.findOne()
{
  "_id" : ObjectId("5721794b349c32b32a012b11"),
  "title" : "Star Wars: Episode IV - A New Hope",
  "director" : "George Lucas",
  "year" : 1977
}
```

To search for documents in the collection (for example, the name of the tg collection), you need to enter a command with the query parameter (for example, by the name field):

```
>db.tg.find(name='@Aavst')
```

The result will look something like this:

```
{ "_id" : ObjectId("60cb8af025ad9284143be469"), "saddr" : "@Aavst" }
```

To search, for example, by another field (the `_id` field), you need to enter a query:

```
> db.tg.find({"_id": ObjectId("60cb8af025ad9284143be46c")})
```

To exit the line editor, enter the following command:

```
>exit
```

To delete a collection, enter the following command:

```
>db.<collection name>.drop()
```

To delete the database, enter the following command:

```
>db.dropDatabase()
```

Import a database from a table in an external format

If the user has a data array (table) at his disposal, presented, in particular, in CSV format, then he can create and load the MongoDB database with this data. It can use the `mongoimport` utility to import the corresponding table. Below is an example of creating and loading a database from a table in CSV format. For this, it is enough to apply the command specified from the command line of the operating system:

```
$mongoimport -d <base name> -c <collection name>  
--type <file type> -file <file name> --fields <comma separated fields>
```

Example command:

```
$mongoimport -d sources -c ytc --type csv -file ytc.csv --fields  
saddr,name
```

Example result:

```
2021-06-17T20:39:05.040+0300 connected to: mongodb://localhost/  
2021-06-17T20:39:05.221+0300 104 document(s) imported successfully.  
0 document(s) failed to import.
```

Export the database to a table in an external format

If the user has at his disposal a database in the MongoDB DBMS environment, then he has the option of exporting (unloading) information from this database to an external table presented, in particular, in CSV format, which can be used as input data for other programs. It can use the mongoexport utility to import the corresponding table. Below is an example of exporting information to a table in CSV format. For this, it is enough to apply the command specified from the command line of the operating system:

```
$mongoexport --<file type> -o <file name> -d <base name> -c  
<collection name> -f =<comma fields>
```

Example command:

```
$mongoexport --csv -o /tmp/ytc1.csv -d sources -c ytc -f saddr,name
```

Example of a report on the result:

```
2021-06-17T20:43:13.164+0300 csv flag is deprecated; please use --  
type=csv instead  
2021-06-17T20:43:13.197+0300 connected to: mongodb://localhost/  
2021-06-17T20:43:13.247+0300 exported 104 records
```

Questions for practical work

1. What are NoSQL DBMSs and what are their main properties?
2. How is MongoDB installed?
3. What are the main commands for interacting with MongoDB from the command line?
4. In what formats can information be imported or exported to/from MongoDB databases?

Tasks for independent work

1. Install the MongoDB system
2. Familiarize yourself with the basic concepts of the MongoDB DBMS. What is the difference from SQL-DBMS?

3. Familiarize yourself with the features of data import/export from JSON format.

literature

1. Karl Seguin. "The Little MongoDB Book". -34 p. URL: <http://github.com/karlseguin/the-little-mongodb-book>
2. Shannon Bradshaw, Jon Brazil, Christina Khodorov. "MongoDB: The Complete Guide. A powerful and scalable database management system" - Moscow: DMK Press, 2020. - 540 p.
3. Markin A.V. Post-shooting data bases. MongoDB: textbook / Markyn A.V. — Moscow: Ai Pi Ar Media, 2020. — 383 c. — ISBN 978-5-4497-0632-4.
4. Import and Export MongoDB database. URL: <https://betacode.net/10279/importing-and-exporting-mongodb-database>

13. The Compass client of the MongoDB DBMS

The purpose of the practical work corresponding to the section is to gain experience with the Compass system - the graphical user interface (GUI) of the MongoDB database, the installation of this GUI, working with data presented by them in JSON format, CRUD commands (Create - creation, Read - reading, Update – modification, Delete – deletion).

Since MongoDB 3.2, MongoDB Compass has been shipped as a graphical shell.

Installing the Compass GUI

To download MongoDB Compass, you need to go to the official address of this GUI <https://www.mongodb.com/try/download/compass>. On this page, you can select the download options – Compass version and target operating system (Figure 13.1).

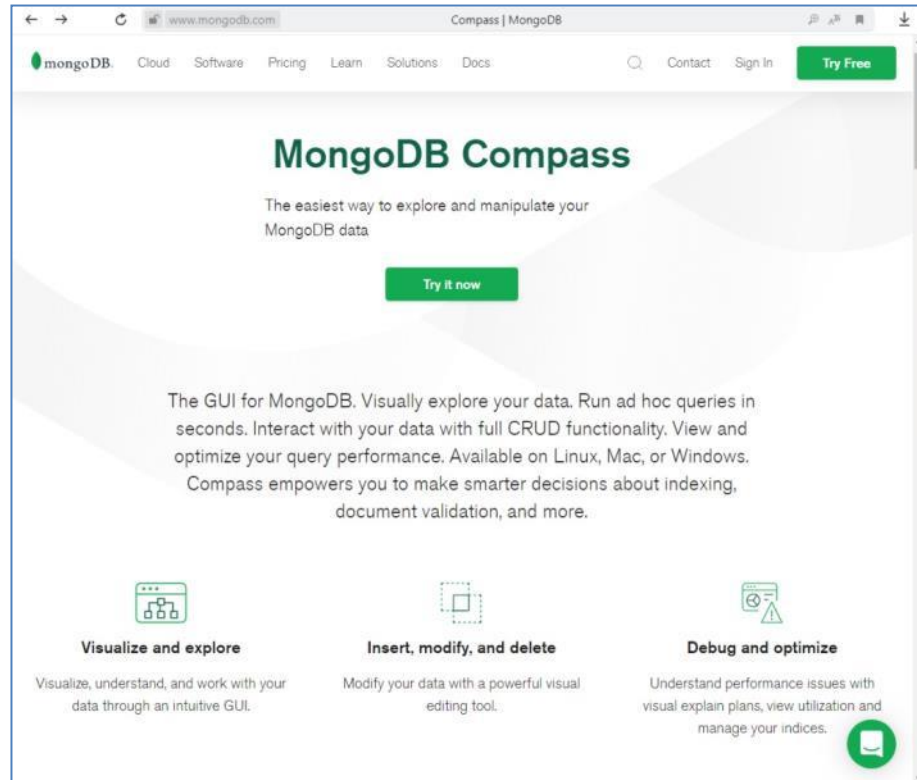


Fig. 13.1 - MongoDB Compass Start Page

After downloading and unpacking the MongoDB Compass archive, we get a directory (Fig. 13.2).

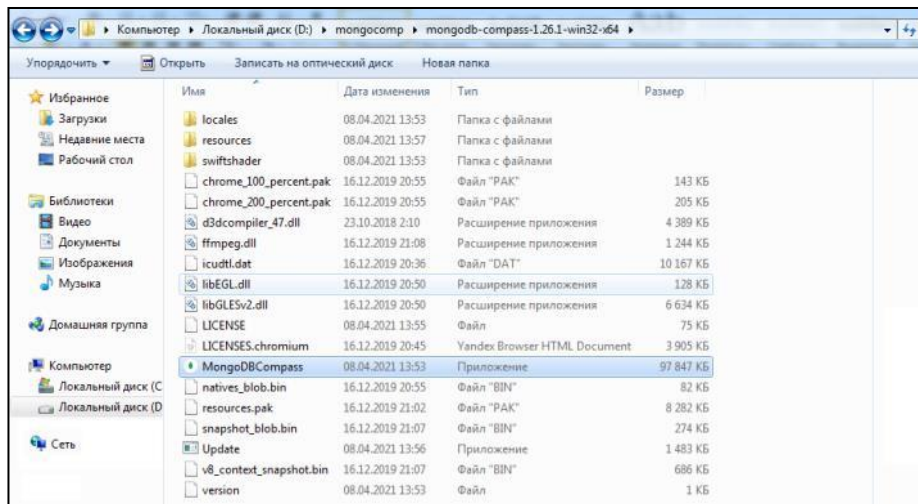


Fig. 13.2 – The MongoDB Compass directory on the client computer

Starting the MongoDB Compass GUI

Here you can find the MongoDBCompass.exe file. It is he who represents the program module. After activating this module, we will get the system interface, where you need to enter the address of the server and the port where the mongod service is already activated (Fig. 13.3).

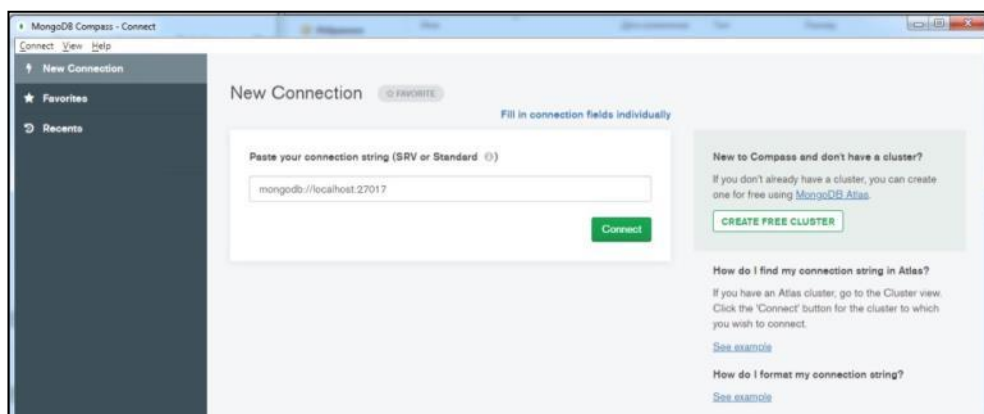


Fig. 13.3 - Configuring the MongoDB Compass client

After going to the specified address (Connect button), the transition to the list of active databases is displayed (Fig. 13.4).

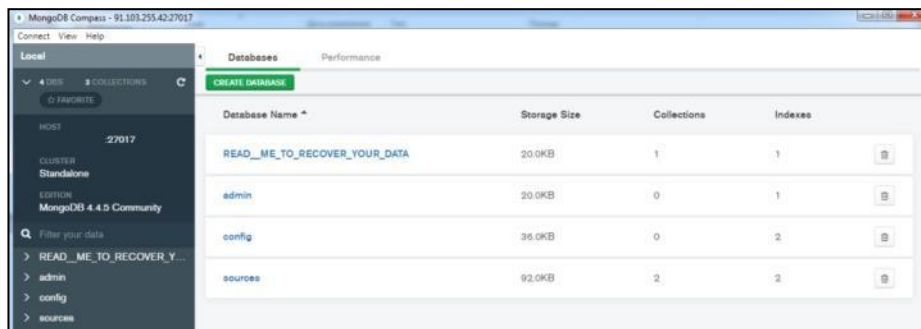


Fig. 13.4 - List of MongoDB Compass databases

The previous section provided information on creating the sources database by importing from a table. Let's go to a detailed review of this database by activating the appropriate hyperlink (Fig. 13.5).

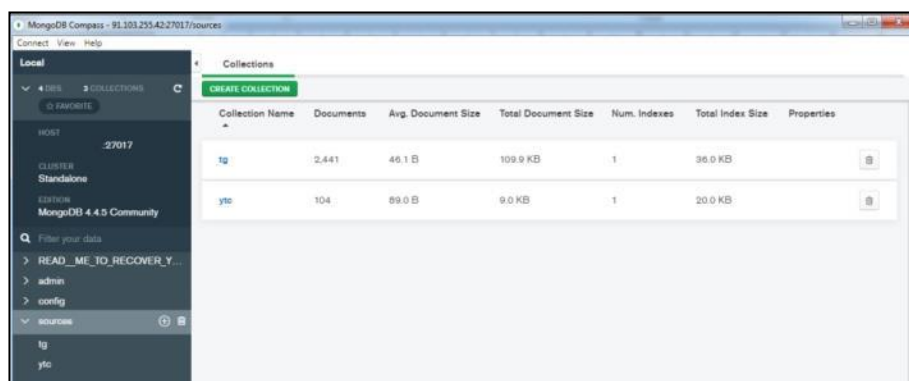


Fig. 13.5 – List of collections of the sources database

In the given example, you can see that the sources database contains two collections - tg and ytc (collections of addresses of Telegram and Youtube channels, respectively). In the given table, you can see the number of documents in each collection (2441 and 104), the average length of the document, the amount of disk space occupied by the collections, the number of indexes (auxiliary search files), and the volume of indexes.

Implementation of CRUD commands

Switching to viewing a specific collection (for example, ytc, Youtube channels) is carried out by activating the corresponding hyperlink.

As can be seen in Fig. 13.6, MongoDB Compass implements access to view individual collection documents listed in JSON format.

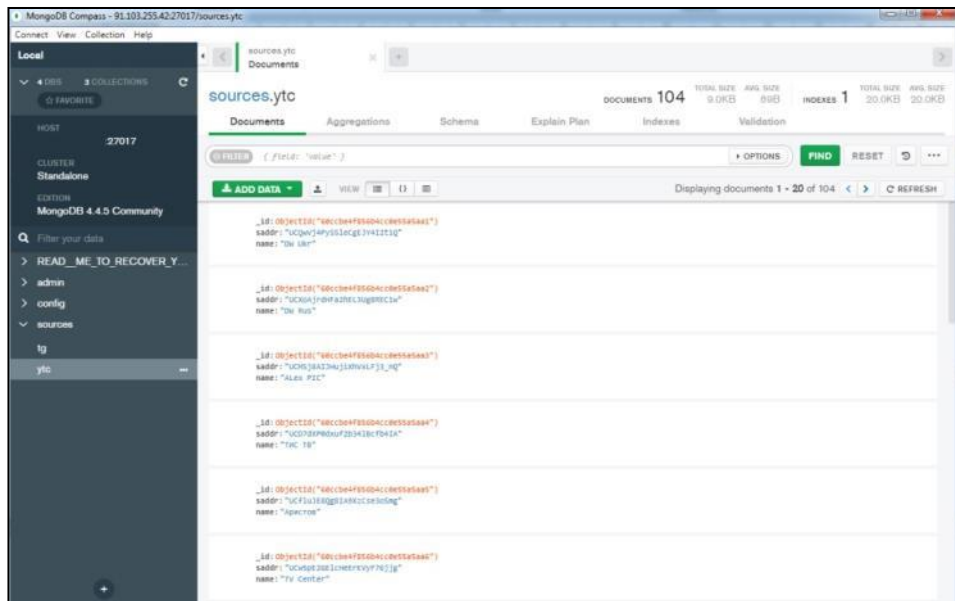


Fig. 13.6 - Content of the ytc Collection

In addition, the implementation of the CRUD - R (Read) function is also possible in the search mode. To do this, it is enough to enter a query in JSON format in the Filter field, for example `{name: 'Baltiy nedeli'}`. The result is shown in Fig. 13.7.

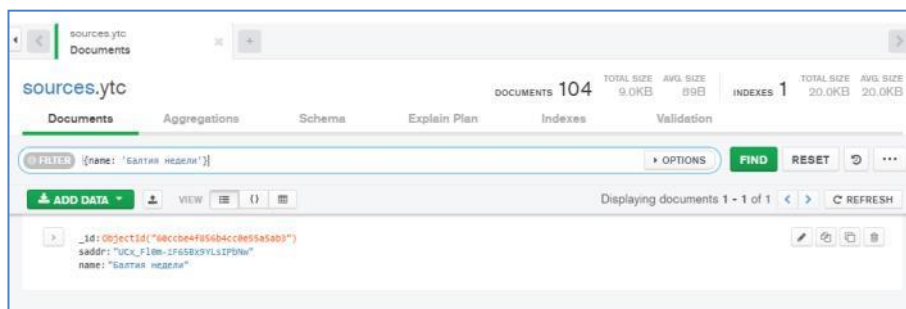


Fig. 13.7 – Search result

Implementation D (Delete): As we can see, the deletion operation for the entire collection (similar to `db.<collection name>.drop()`) is implemented on the interface (Fig. 13.5) in the form of the "Trash can" icon.

The same operation for the entire database (similar to `db.drop.Database()`) can be seen in Fig. 13.5 in the form of the same icon.

For a separate record, it is enough to activate the similar icon shown in Fig. 13.7.

To implement the U (Update) function for a separate document, you need to activate the Edit icon, after which you can edit individual fields of the document and add new ones (Fig. 13.8).

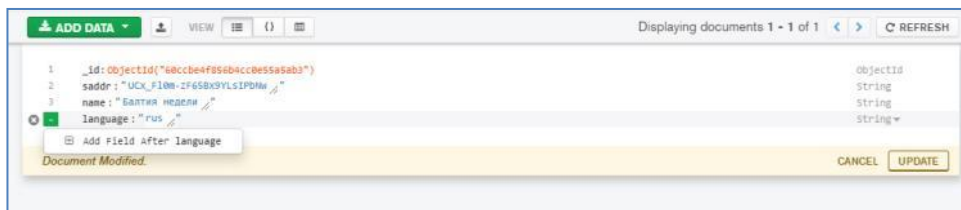


Fig. 13.8 – Search result

C(Create) – creation of new documents (ADD DATA mode) is possible in two options (manual input or file import - Fig. 13.9, 13.10).

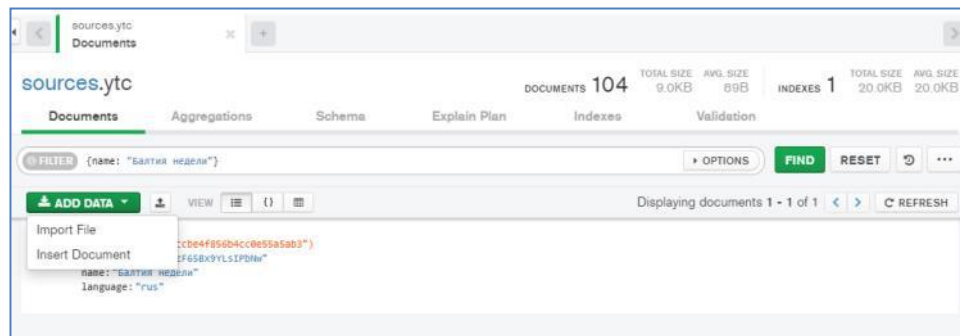


Fig. 13.9 – ADD DATA mode

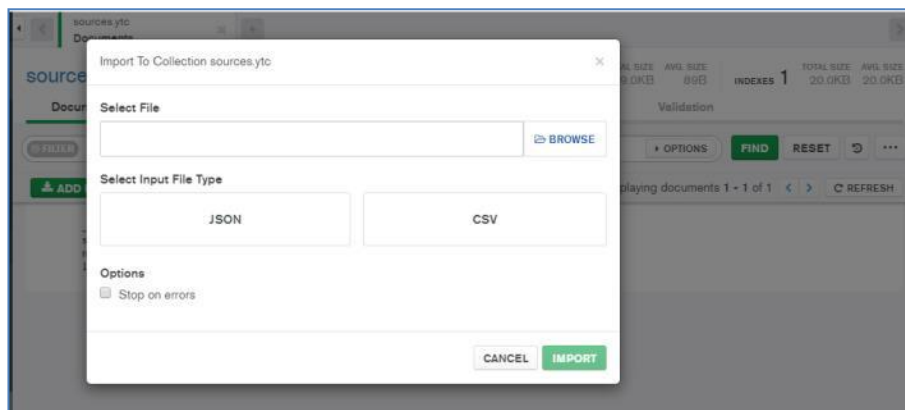


Fig. 13.10 – Creating a new document

When importing data from the download, it is carried out by running the `mongoimport` command in the background after activating the `IMPORT` key.

Questions for practical work

1. What is CRUD?
2. How to install the MongoDB Compass system ?
3. How are the main commands for interaction with MongoDB implemented through MongoDB Compass ?
4. In what formats can information be imported or exported to/from MongoDB databases?

Tasks for independent work

1. Install the MongoDB Compass GUI
2. Familiarize yourself with the main features of the MongoDB Compass GUI.
3. CRUD capabilities in the MongoDB Compass environment.

literature

1. Karl Seguin. "The Little MongoDB Book". -34 p. URL: <http://github.com/karlseguin/the-little-mongodb-book>
2. Shannon Bradshaw, Jon Brazil, Christina Khodorov. "MongoDB: The Complete Guide. A powerful and scalable database management system". - M.: DMK Press, 2020. - 540 p.
3. A.N. Filippov, Ya.A. Povolotskyi Application of the non-relational DBMS MongoDB in CAD TP / Methodical manual // Application description. St. Petersburg: ITMO University, 2018. – 46 p. URL: <https://books.ifmo.ru/file/pdf/2441.pdf>
4. MongoDB for Giant Ideas (Electronic resource <https://www.mongodb.com/>).

14. Basics of working with Neo4j

The purpose of the practical work corresponding to the section is to gain experience with the Neo4j graph DBMS using the Neo4j Browser and using the Cypher query language, which provides a human-readable SQL-like query language to graph databases.

Of course, the concept of *Big Data* is directly related to network tasks, since the number of possible connections in networks significantly exceeds the number of nodes. Special software systems are created to work with such tasks. Including Neo4j — an open-source graph database management system implemented in Java. It is considered the most common graph DBMS.

Data in Neo4j is stored in its own format, specially adapted for the presentation of graph information, this approach, in comparison with the modeling of a graph database by means of a relational DBMS, allows applying additional optimization in the case of data with a more complex structure.

To process a graph, it is not necessary to place it entirely in the RAM of the computing node, thus, it is possible to process fairly large graphs.

The application programming interface for DBMS is implemented for many programming languages, including Java, Python, Clojure, Ruby, PHP, and a REST-style API is also implemented.

DBMS uses its own query language - Cypher, which is not only a query language, but also a data manipulation language, as it provides CRUD functions for graph storage.

This section is devoted to the basics of working with the Neo4j graph DBMS, using its most important part - the Neo4j Browser.

The start page of the web server (<https://neo4j.com>) is shown in Fig. 14.1.

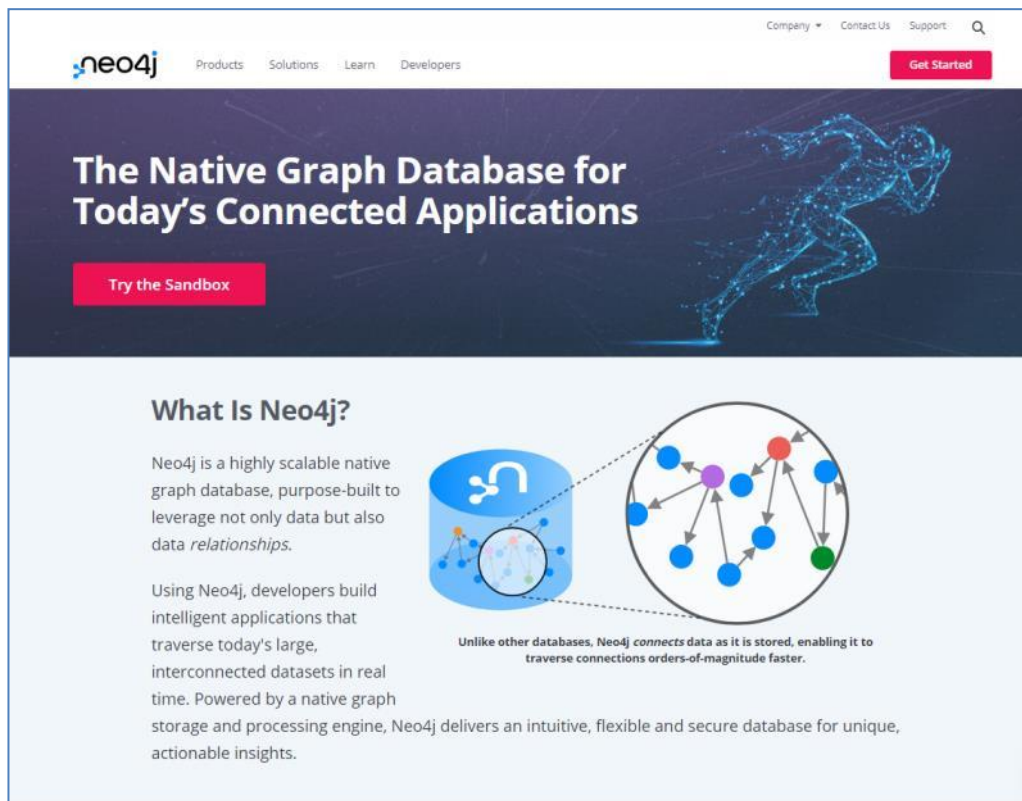


Fig. 14.1 - Neo4j start web page

Installing Neo4j

To install Neo4j, you need to go to the download center at the address: <https://neo4j.com/download-center/> (Fig. 14.2).

For further operation of Neo4j, the following initial requirements must be met:

- A minimum of 2Gb of RAM is required for Neo4j DBMS to work, and 16Gb is recommended for stable operation.
- As a disk array, it is recommended to use SSD disks.
- Neo4j is implemented in Java, so you need to install JVM8.

Before installing the system itself, you must first download the encrypted GPG key and add the repository to the local list:

```
wget --no-check-certificate -O -  
https://debian.neo4j.org/neotechnology.gpg.key | sudo apt-key add  
-  
  
echo 'deb http://debian.neo4j.org/repo stable/' >  
/etc/apt/sources.list.d/neo4j.list
```

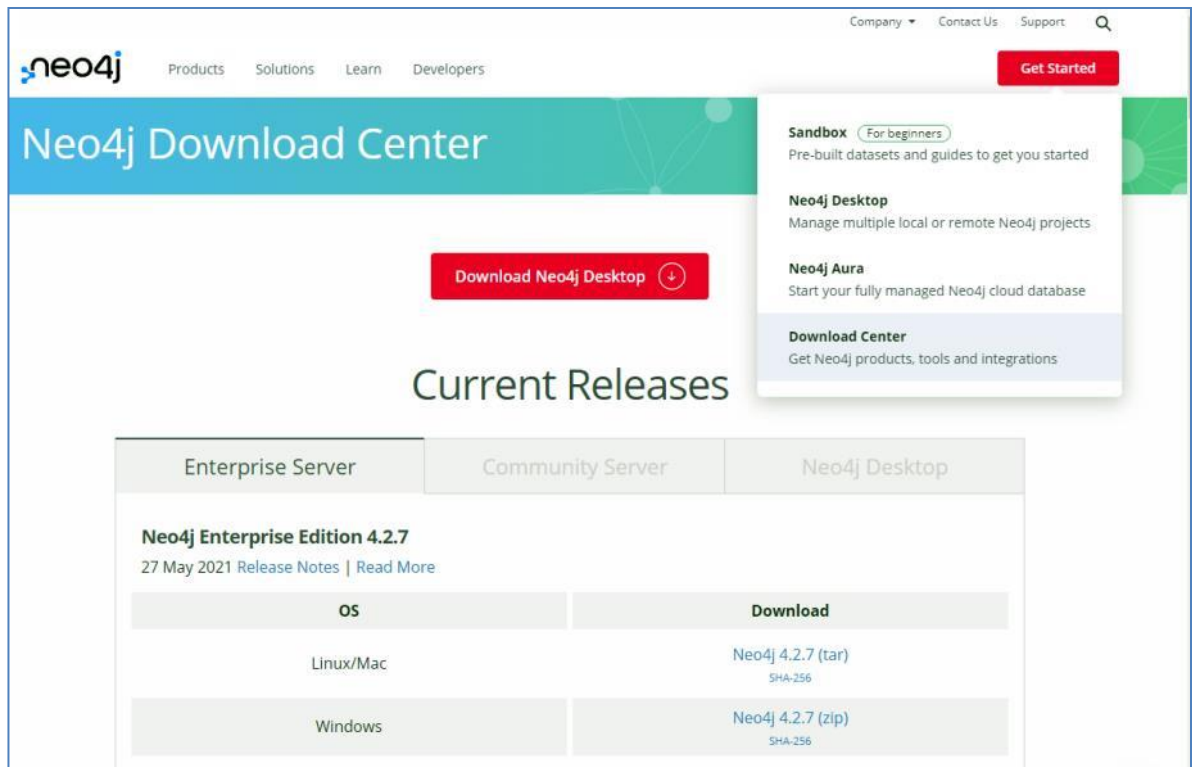


Fig. 14.2. Neo4j Download Center

After that, you need to update the local package database and install the DBMS:

```
apt update apt install neo4j
```

In order to connect to the database from any IP address, and not only on the local host, some lines must be changed in the configuration file. To do this, you need to open the settings file using a text editor:

```
/etc/neo4j/neo4j.conf
```

You need to find the lines, they are in different parts of the file:

```
#dbms.connector.http.listen_address=:7474
#dbms.connector.bolt.listen_address=:7687
```

You need to uncomment as shown below:

```
dbms.connector.http.listen_address=0.0.0.0:7474
dbms.connector.bolt.listen_address=0.0.0.0:7687
```

After saving the changes, you need to restart Neo4j:

```
service neo4j restart
```

If necessary, configure the firewall for remote access:

```
iptables -A INPUT -p tcp --dport 7474 -j ACCEPT
iptables -A INPUT -p tcp --dport 7687 -j ACCEPT
```

Connection in the browser

It is necessary to open a browser and connect to the DBMS at the following address:

<IP address of Neo4J>: 7474

To work with the database, the bolt network protocol is used - a highly efficient and lightweight client-server protocol designed for database applications.

When connecting for the first time, replace the localhost value with your IP address for the bolt protocol, and use the database name as the password, then specify and confirm the password for the Neo4J user (Fig. 14.3).

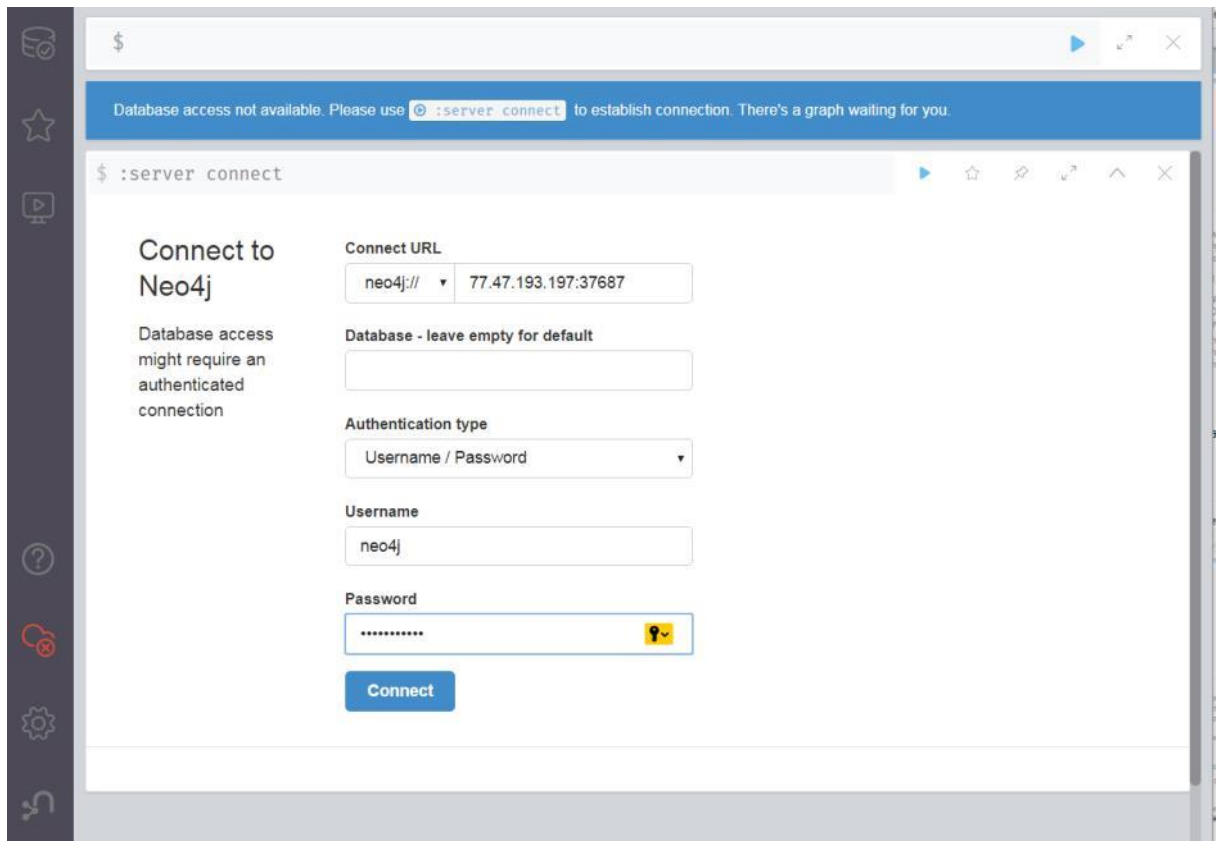


Fig. 14.3 – Connecting to Neo4J

After that, you can start working with Neo4J graph DBMS.

Starting the system

To start working with Neo4J, you need to check that the DBMS is running (example for Linux):

```
service -status-all | grep neo4j  
[ + ] neo4j
```

The "plus" sign means that the DBMS is already running, the "minus" sign means that it is not yet running. To start Neo4J, execute the command:

```
sudo service neo4j start
```

After starting, you can go to the link <http://localhost:7474/browser/> using a regular browser, after which the Neo4j Browser interface should appear (Fig. 14.4).

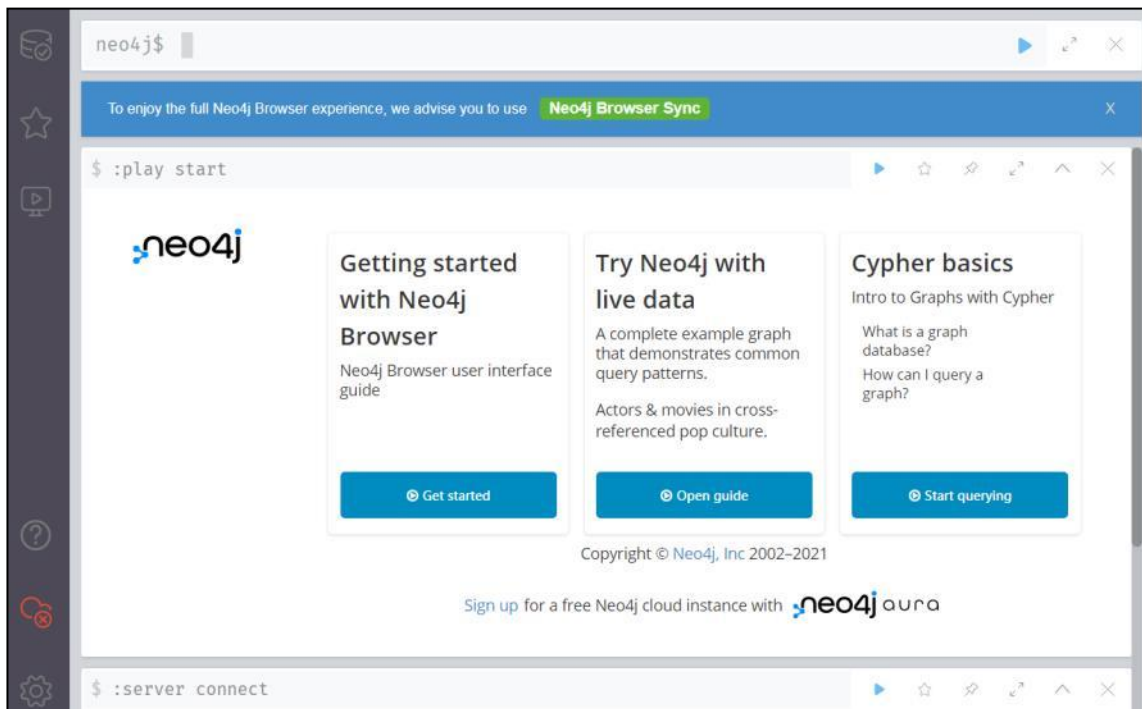


Fig. 14.4 – Neo4j Browser interface

In the upper part of the Neo4j Browser window there is a row of the so-called editor. Starting a set of commands with a colon, we will see a list of all available commands with a brief description (Fig. 14.5).

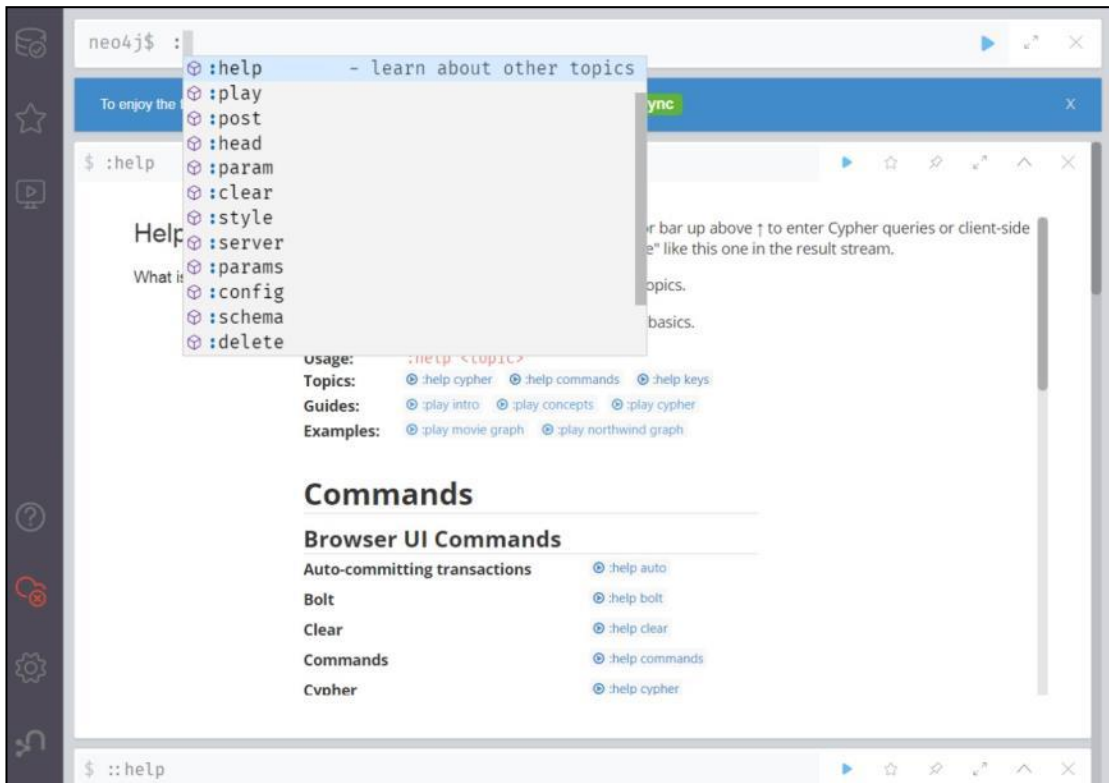


Fig. 14.5 - List of Neo4j Browser commands

A detailed description of the commands can be obtained by calling the `:help` command (Fig. 14.6).

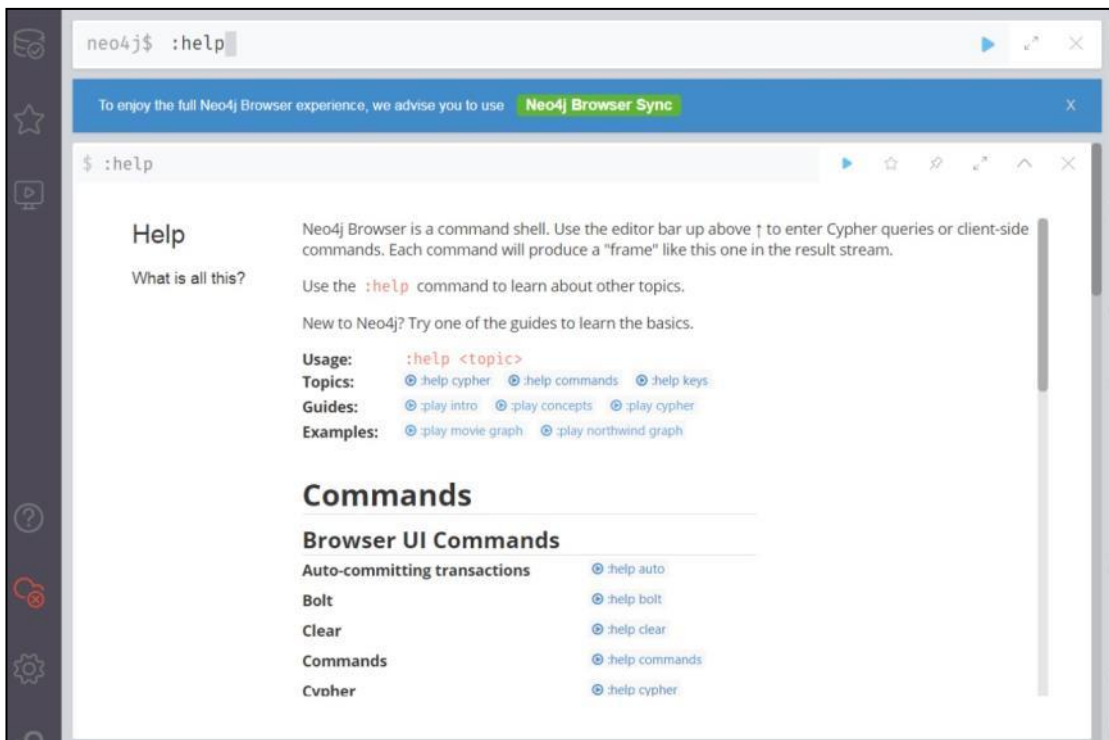


Fig. 1 4.6 – Command :help

Creating a graph

To create and further process graphs, we will use the Cypher language, which is a data manipulation language and also provides CRUD functions for the graph store.

For a brief introduction to the Cypher language, you can call the command (Fig. 14.7):

```
:play cypher
```

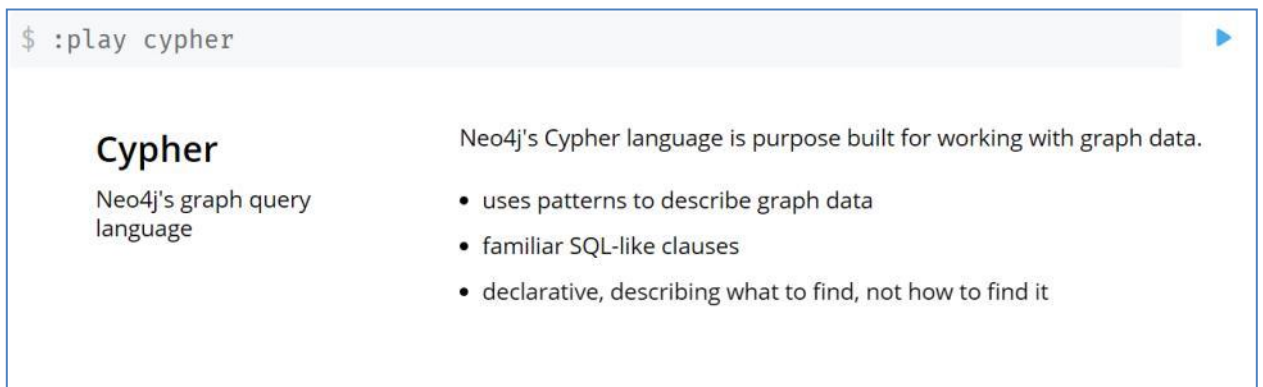


Fig. 14.7 – Description of the Cypher language

To start creating a small graph of social orientation, let's go to the editor and type the first command in the Cypher language:

```
CREATE (u1:Person {name: "Vasy1", from: "Zhytomyr"})
```

After executing the Browser command, the result is displayed (Fig. 14.8).



Fig. 14.8 – Creating a node

Let's add another node:

```
CREATE (u2:Person {name: "Dmytro", from: "Kyiv"})
```

Now we will query all nodes of the Person type and output the value of the name property (Fig. 14.9):

```
MATCH (ee:Person) RETURN ee.name
```

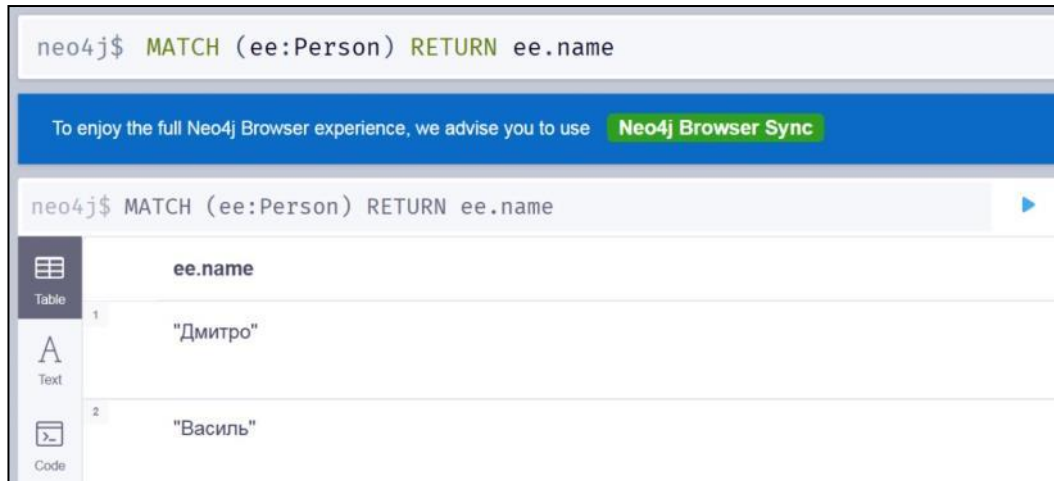


Fig. 14.9 – Displaying the list of nodes

It is possible to sort the extracted data by some field:

```
MATCH (ee:Person) RETURN ee.name ORDER BY ee.name
```

Next, we can request all nodes of this type (Fig. 14.10):

```
MATCH (ee:Person) RETURN ee
```

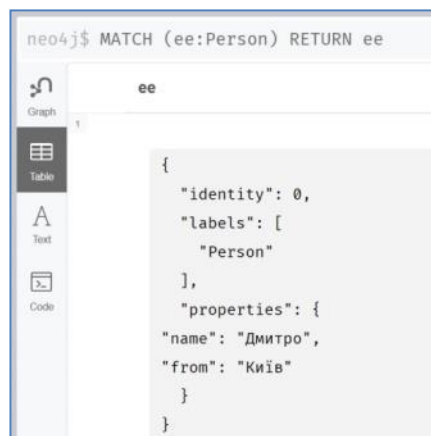


Fig. 14.10 – Full description of the node

By clicking on the Graph button, you can see our nodes in a graphical form (14.11).



Fig. 14.11 – Graphical representation of nodes

Connections between nodes in Cypher can be added using the CREATE command (Fig. 14.12, 14.13).



Fig. 14.12 – Graphical representation of nodes

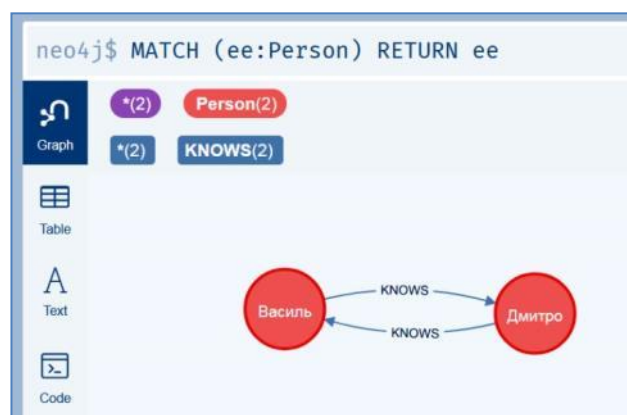


Fig. 14.13 – Display graph with connections

With the help of Cypher, you can also perform various operations on graphs, for example, query adjacent vertices, friends in a social graph, delete edges and vertices, and much more, but this is a topic for a separate conversation.

You can also configure Neo4j Browser to display nodes and connections in a different style depending on the labels assigned to them.

Deleting a graph (Fig. 14.14):

```
Match (n)-[r]-()
Delete n, r;
```

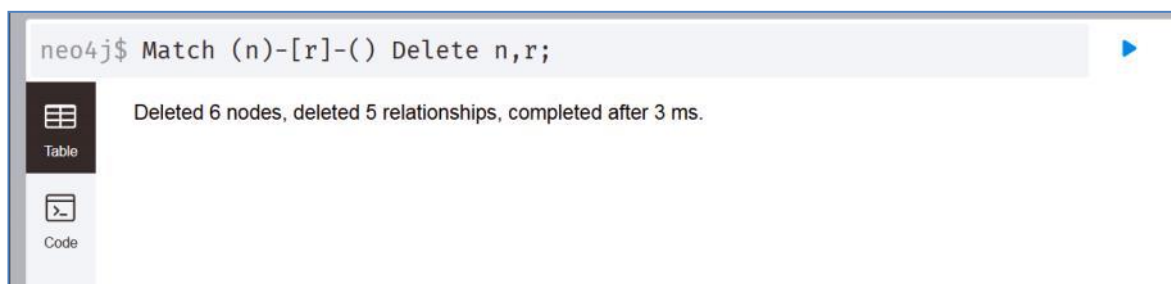


Fig. 14.14 – Deleting a graph

Import data from .csv format into a neo4j graph

We present the connection between the named nodes (in the example, we will mark them with numbers). Consider, for example, a set of strings in which each string expresses an adjacency relation between two nodes, for example, 1->2, 2->3, 1->3, etc.

```
1,2
2,3
1.3
1.4
2.5
3,4
3.5
4.5
```

This file containing these lines should be placed on the local machine (for a standard neo4j installation)

```
/var/lib/neo4j/import/
```

The name of the file, for example csv

Then the commands for downloading the file, forming and displaying the graph will look like this:

```

LOAD CSV FROM 'file:///csv' AS line
MERGE (a:node {name:line[0]})
MERGE (b:node {name:line[1]})
MERGE (a)-[:connects]->(b);

```

After that, the network can be displayed (Fig. 14.15) by entering the command:

```
MATCH (ee:name) RETURN ee
```

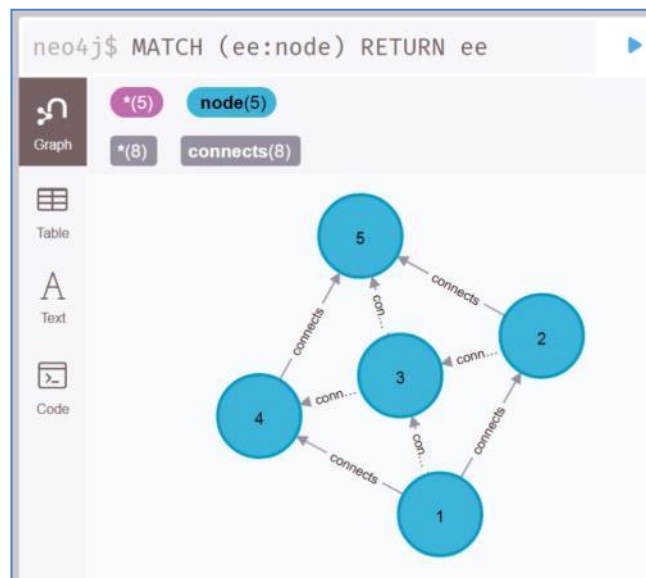


Fig. 14.15 – Display of the loaded graph

Questions for practical work

1. What are graph DBMSs and what are their main features?
2. What other DBMS besides Neo4J does the Cypher query language support?
3. What CRUD- functions can be used to work with graph storage?

Tasks for independent work

1. Familiarize yourself with the main capabilities of the graph DBMS in detail. Neo4J and Cypher languages
2. Get acquainted with the possibilities of loading graphs from JSON format
3. Familiarize yourself with the options for graph visualization using the :style command.

literature

1. Robinson Ian, Weber Jim, Eifrem Emil. Graph data bases: new opportunities for working with connected data / trans. with English R. N. Raghimova; science ed. A. N. Kysylev. - 2nd ed. - M.: DMK Press, 2016. - 256 p. ISBN 978-5-97060-201-0
2. Eric Redmond, Jim. R. Wilson. Seven databases in seven weeks. Introduction to modern databases and NoSQL ideology. Edited by Jacqueline Carter / Trans. with English A. A. Slinkin - Moscow: DMK Press, 2013. - 384 p. ISBN 978-5-94074-866-3
3. Holzschuher, Florian and Peinl, Rene (2013). "Performance of Graph Query Languages: Comparison of Cypher, Gremlin and Native Access in Neo4J" in EDBT '13. Proceedings of the Joint EDBT/ICDT 2013 Workshops: 195-204, Genoa, Italy: ACM. DOI:10.1145/2457317.2457351

Addition. A minimum of Python is required

Python is a universal programming language that supports object-oriented programming, created in 1991 by Guido van Rossum. The official website of the programming language is <http://python.org>. Python is freely distributed under a license similar to the GNU General Public License.

Python is an interpreted language: the source code is converted to machine code in the process of interpretation.

Data types

The following data types are used in Python:

1. None (undefined variable value)
2. Logical variables (Boolean Type)
3. Numbers (Numeric Type)
 - int is an integer
 - float – a floating point number
 - complex is a complex number
4. Lists (Sequence Type)
 - list – a list
 - tuple is a tuple
 - range - range
5. Lines (Text Sequence Type)
 - str

In some cases, the program receives data in the form of strings, and must operate with numbers (or vice versa). In this case, special functions (special operators) are used to convert one type of data to another. Thus, the `int()` function converts the string (or floating-point number) passed to it into an integer, the `str()` function turns the argument passed to it into a string, and `float()` into a fractional number.

Strings as sequences of characters

A string is a complex data type that is a sequence of characters. Strings in Python can be enclosed in both single and double quotes. However, the beginning and end of a line must be surrounded by the same type of quotes.

A string is a basic type that represents an unchanging sequence of characters (str from the English "string").

Functions and methods of working with strings

A function or method	Appointment
S1 + S2	Concatenation (combination of strings)
S1 * 3	Row repetition
S[i]	Reference by index
S[i:j:step]	Section extraction
len (S)	String length
S.join (list)	Concatenates strings from the sequence str using the delimiter specified by string
S1.count (S[, i, j])	Number of occurrences of substring s string s 1. The result is a count. You can specify the starting position of search i and the end of search j
S.find (str, [start], [end])	Search for a substring in a string. Returns the number of the first occurrence or -1
S.index (str, [start], [end])	Search for a substring in a string. Returns the number of the first occurrence or throws a ValueError
S.rindex (str, [start], [end])	Search for a substring in a string. Returns the number of the last occurrence or throws a ValueError
S.replace (template, replacement)	Template replacement
S.split (symbol)	Splitting a line by a delimiter
S.upper ()	Convert string to upper case
S.lower ()	Convert string to lower case

There is a function len() that allows you to measure the length of a string. The result of its execution is a number that shows the number of characters in the line.

There are also concatenation (+) and duplication (*) operations for strings.

Examples:

```
>> len('It is a long string')
```

Result:

```
19
```

```
>> '!!!!' + 'Hello World' + '!!!!'
```

Result:

```
'!!!! Hello World !!!!'
```

```
>> '-' * 20
```

Result:

```
'-----'
```

In sequences, the order of symbols is important, each symbol in a line has a unique serial number - an index. You can refer to a specific character in a string and extract it using the indexing operator, which is square brackets with the character number in them.

```
>>> 'morning, afternoon, night'[1]
```

Result:

```
'o'
```

```
>>> tday = 'morning, afternoon, night'
```

```
>>> tday[4]
```

Result:

```
'i'
```

In the example, the expression 'morning, afternoon, night'[1] resulted in the extraction of the second character. The fact is that indexing does not start from one, but from zero. You can also extract symbols starting from the end. And here the countdown starts from -1 (the last character).

You can extract not one character from the line, but several, i.e. get slice (substring). The operator for extracting a slice from a string has the following

form: [X:Y). X is the index of the beginning of the section, and Y is its end; and the symbol with the number Y is no longer included in the slice. If there is no first index, then the slice is taken from the beginning to the second index; in the absence of the second index, the slice is taken from the first index to the end of the line.

```
>>> tday = 'morning, afternoon, night'
>>> tday[0:7]
'morning'
>>> tday[9:-7]
'afternoon'
>>> tday[-5:]
night
```

Lists are changing sequences

Python lists, like strings, are ordered sequences. However, unlike strings, lists do not consist of symbols, but of various objects (values, data), and are enclosed not in quotation marks, but in square brackets []. Objects are separated from each other with a comma.

Lists can consist of various objects: numbers, strings, and even other lists. In the latter case, the lists are called nested.

```
[23, 656, -20, 67, -45] # list of integers
[4.15, 5.93, 6.45, 9.3, 10.0, 11.6] # list of fractional numbers
["Katy", "Sergei", "Oleg", "Dasha"] # a list of strings
["Moscow", "Tytova", 12, 148] # mixed list
[[0, 0, 0], [0, 0, 1], [0, 1, 0]] # a list consisting of lists
```

As with strings, you can perform concatenation and repetition operations on lists:

```
>>> [45, -12, 'april'] + [21, 48.5, 33]
[45, -12, 'april', 21, 48.5, 33]
>>> [[0,0],[0,1],[1,1]] * 2
[[0, 0], [0, 1], [1, 1], [0, 0], [0, 1], [1, 1]]
```

Arrays of elements in the form of rectangular tables, which define the rules of mathematical processes, are called matrices. Matrix elements can be numbers, algebraic symbols, or mathematical functions.

Python also uses lists to work with matrices. Each element of the matrix list contains a nested list.

Thus, a structure of nested lists is formed, the number of which determines the number of matrix columns, and the number of elements inside each nested list indicates the number of rows in the original matrix.

A two-dimensional list cannot be created by repeating a single row. To create a list, you first create a list of n elements (just n zeros to begin with), then make each element of the list a reference to another one-dimensional list of m elements.

Example:

```
n=3
m=3
A=[0]*n
for i in range(n):
    a[i]=[0]*m
print('A: ',A)
```

Result:

```
A: [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Output of nested list (two-dimensional array)

Two nested loops are usually used to process and output the list. The first cycle is by row number, the second cycle is by elements within the row. For example, you can display a two-dimensional numerical list on the screen line by line, separating the numbers with spaces within one line, as follows:

```
for i in range(n):
    for j in range(n): print(A[i][j], end = ' ') print()
```

Dictionaries

One of the complex data types (along with strings and lists) in the Python programming language are dictionaries. A dictionary is a mutable (like a list) unordered (as opposed to strings and lists) collection of key:value pairs.

The syntax of a dictionary in Python can be described by the following scheme:

```
{ key: value, key: value, key: value, ...}
```

In the dictionary, values are accessed by keys enclosed in square brackets.

```
>>> dic
={ 'cat': 'cat', 'dog': 'dog', 'bird': 'bird', 'mouse': 'mouse' }
>>> dic['cat']
'cat'
>>> dic['bird']
'bird'
```

Dictionaries, like lists, are a mutable data type: elements (key:value pairs) can be modified, added, and deleted. The dictionary can be initially created empty (for example, `d = {}`) and then filled with elements. Adding and modifying have the same syntax: `dictionary [key] = value`.

The key can be either existing (then the value changes) or new (the dictionary element is added). Deleting a dictionary element is done using the `del ()` function.

```
>>> dic
={ 'cat': 'cat', 'dog': 'dog', 'bird': 'bird', 'mouse': 'mouse' }
>>> dic['elephant'] = 'elephant'
>>> dic['fox'] = 'fox'
>>> Del
```

Operators

Conditional operator. The if instruction

Sometimes a piece of code must be executed only for a certain value of a particular variable. Usually in programming languages, something like this construction is used:

```
if a logical operator b:
    an expression that is executed if the result is True
```

Implementation example:

```
if numbig < 100: # if the value of the variable numbig is less
than 100, then...
    c = a**b # reduce the value of the variable a to the power
of b,
    # assign the result to c.
```

(# is a comment sign, everything that follows it is not interpreted programmatically)

There is also a more complex form of branching: if–else. If the condition in the if statement is false, then the block of code in the else statement is executed.

An example of code with an else branch:

```
print "Hello"
tovar1 = 50
tovar2 = 32
if tovar1+ tovar2 > 99 :
print "The amount is not enough"
otherwise:
print "Check paid"
print "Goodbye"
```

Python provides a special extension of the if statement that allows you to direct the flow of program execution along one of many branches. This extended instruction, in addition to the optional else part, contains a number of elif branches (abbreviation from "else if" - "else if").

The syntax of the if – elif – else statement looks like this:

```
if logical expression_1:
command_1
team_2
...
command_n
elif logical expression _2:
command_1
team_2
...
command_n
elif logical expression _3:
team_1
team_2
...
team_n
else:
command_1
team_2
...
team_n
```

Examples of a script using the if-elif-else instruction:

```
x = -10
if x > 0:
print 1
elif x < 0:
print -1
```

```
otherwise:  
print 0
```

Cycles

The while loop

Loops are instructions that perform the same sequence of actions while a given condition is true.

The universal loop organizer in the Python programming language (as in many other languages) is the while construct.

The while construction in Python can be described by the following scheme:

```
while a logical operator b:  
action  
change a
```

The logical expression in the header of the while loop can be complex, and the variable (or expression) b can change.

When the execution of the program code reaches the while loop, the logical expression in the header is executed, and if True, the nested expressions are executed. After the execution of the program, the flow returns to the head of the while loop and the condition is checked again.

The simplest loop can look like this:

```
str1 = "+"  
i = 0  
while i < 10:  
print (str1)  
i = i + 1
```

The for loop

In programs written in Python, the for loop is widely used, which is a loop that goes around a given set of elements and performs various operations on them in its body. For example, if there is a list of numbers, and you need to increase the value of each element by two units, you can iterate through the list using a for loop, performing the corresponding action on each of its elements.

```
>>> spisok = [0,10,20,30,40,50,60,70,80,90]  
>>> i = 0  
>>> for element in list:
```



```

spisok[i] = element + 2
i = i + 1
>>> list
[2, 12, 22, 32, 42, 52, 62, 72, 82, 92]

```

When all elements are processed, the for loop ends its work. The absence of the next element is the termination condition of the for loop (for comparison: in the while loop, the termination condition is the false result of the logical expression in the header).

The for loop is used to work with dictionaries:

```

>>> d = {1:'one',2:'two',3:'three',4:'four'}
>>> for key in d:
d[key] = d[key] + '!'
>>> d
{1: 'one!', 2: 'two!', 3: 'three!', 4: 'four!'}

```

It is worth remembering that Python's for loop is not analogous to for loops in many other programming languages, as it is not a loop with a counter.

```

{'fox': 'fox', 'dog': 'dog', 'cat': 'cat', 'elephant':
'elephant', 'mouse': 'mouse', 'bird': 'bird'}
>>> dic['elephant'] = 'elephant'
>>> del(dic['bird'])
>>> Dic
{'fox': 'fox', 'dog': 'dog', 'cat': 'cat', 'elephant':
'elephant', 'mouse': 'mouse'}

```

Data input and output

Data is entered using the input function (input list):

```

a = input()
print(a)

```

In the brackets of the function, you can specify a message - a comment on the entered data :

```

a = input ("Enter quantity: ")

```

The input() command accepts input as a character string by default. Therefore, to enter an integer value, you should specify the int() function:

```

a = int (input())

```

The float() function is used to enter real numbers

```
a=float(input())
```

Data output is carried out using the print command (output list):

```
a = 1
b = 2
print(a)
print(a + b)
print('sum = ', a + b)
```

It is possible to write commands in one line, dividing them by ;. However, you should not use this method often, it reduces the convenience of reading:

```
a = 1; b = 2; print(a)
print (a + b)
print ('sum = ', a + b)
```

Mathematical operations in Python

Integers (int)

Python supports a set of the simplest mathematical operations:

$x + y$	Addition
$x - y$	Subtraction
$x * y$	Multiplication
x / y	Division
$x // y$	Obtaining a whole part from division
$x \% y$	Remainder from division
$-x$	Changing the sign of a number
$\text{abs}(x)$	The module of a number
$\text{divmod}(x, y)$	Pair ($x // y, x \% y$)
$x ** y$	Reduction to degree
$\text{pow}(x, y[, z])$	x : The number to be raised to a power. y : A number that is the measure to which the first argument needs to be reduced. If the number is negative or one of the numbers "x" or "y" is not an integer, then the argument "z" is not accepted. z : The number to divide by modulo. If a number is

specified, "x" and "y" are expected to be positive and of type int.

Library (module) math

Python comes standard with the math library, which contains a large number of frequently used mathematical functions.

To work with this module, you need to import it:

```
Import math
```

The most frequently used functions of the math module

math.ceil(x)	Returns the nearest integer greater than x
math.fabs(x)	Returns the absolute value of x
math.factorial(x)	Calculates the factorial of x
math.floor(x)	Returns the nearest integer less than x
math.exp(x)	Calculates e^{**x}
math.log2(x)	Logarithm to base 2
math.log10(x)	Logarithm to base 10
math.log(x[, base])	By default, it calculates the logarithm based on the base e. Additionally, you can specify the base of the logarithm
math.pow(x, y)	Calculates the value of x to the power of y
math.sqrt(x)	The square root of x

Trigonometric functions of the math module

math.cos(x)	Returns the cos of X
math.sin(x)	Returns the sin of X
math.tan(x)	Returns the tan of X
math.acos(x)	Returns the acos of X
math.asin(x)	Returns the asin of X

<code>math.atan(x)</code>	Returns the atan of X
---------------------------	-----------------------

Constants:

- `math.pi` - the number π .
- `math.e` - number e (exponent).

Regular expressions

Regular expressions, also called regex, are used in almost all programming languages. In Python, they are implemented in the standard `re` module.

A regular expression pattern is a special language used to represent common text, numbers, or symbols, extracting texts that match this pattern.

Regular expressions are patterns used to find a matching piece of text and match characters.

Regular expressions contain special characters/operators to create complex patterns.

Operator “[]”

Any of the characters in square brackets .

`[abc]` – will find a, b and c.

`[az]` – will find all values from a to z.

`[a-zA-Z0-9]` – will find all values from a to z, from A to Z and from 0 to 9.

The dot operator

operator is used to match any single character except the newline character.

This operator can be used in combination with others.

Metasequences

Some patterns are used all the time. There are shortcuts for them. Here are the most commonly used:

`\w` — matches any letter, number, or underscore, equivalent to `[a-zA-Z0-9_]`.

`\W` - Matches any character except alphanumeric characters and underscores.

`\d` — matches any numeric character,
equivalent to `[0–9]`.

`\D` matches any non-numeric character.

Operators “+” and “*”

The `+` character is used for 1 or more values of the left character.

The `*` character is used for 0 or more left character values.

For example, if we want to find all substrings starting with "d" and ending with "e", we may encounter zero or more characters between "d" and "e". Used by: `d\w*e`.

If you want to find all substrings starting with "d" and ending with "e" with at least one character between "d" and "e", use: `d\w+e`.

Repetition:

`\w{n}` — repeats `\w` exactly `n` times.

`\w{n,}` — repeats at least `n` times or more.

`\w{n1, n2}` - repeats `\w` at least `n1` times, but not more than `n2` times.

Operators “^” and “\$”

The “`^`” operator highlights the beginning of a line, and the “`$`” highlights the end of a line.

Search codes

A short list with a brief explanation of each code:

`\d` corresponds to a number

`\D` does not match a number

`\s` matches an empty field (space)

`\S` corresponds to a filled field

`\w` corresponds to an alphanumeric value

`\W` matches a non-alphanumeric value

Python uses the `re` module to work with regular expressions.

The main functions of the re module:

The compile() function of the re module

The compile() function of the re module compiles the regular expression pattern pattern into a regular expression object that can be used to find matches.

The search() function of the re module

The search() function of the re module scans the string for the first match with the regular expression pattern and returns the corresponding match object.

The match() function of the re module

The match() function of the re module will return a matching match object if zero or more characters at the beginning of string match the regular expression pattern.

The fullmatch() function of the re module

The fullmatch() function of the re module will return a match object if the entire string matches the regular expression pattern.

The finditer() function of the re module

The finditer() function of the re module returns an iterator of match objects over all non-overlapping matches for the regular expression pattern in the string.

The split() function of the re module

The split() function of the re module splits a string when the regular expression pattern appears and returns a list of substrings.

The findall() function of the re module

The findall() function of the re module returns all matches of the pattern in the string string as a list of strings. The string is scanned from left to right and matches are returned in the order found.

The sub() function of the re module

The sub() function of the re module returns the string obtained by replacing the leftmost occurrence of the regular expression pattern in the string string with the

replacement string repl. If the regular expression pattern is not found, the string is returned unchanged.

The subn() function of the re module

The subn() function of the re module performs the same operation as the sub() function, but returns a tuple (new_string, number_of_subs_made)

The escape() function of the re module

The escape() function of the re module escapes special characters in a pattern. This is useful when you want to match an arbitrary string of literals that may contain regular expression metacharacters.

The purge() function of the re module

The purge() function of the re module clears the cache of regular expressions.

Error() exception of module re

The error() exception of the re module is raised when the string passed to one of the module's functions is not a valid regular expression, for example the pattern may contain mismatched parentheses, or when any other error occurs during the compilation of the pattern or mapping to the string.

Regular expression object of the Pattern module re

A Pattern regular expression object is the result of compiling a regular expression pattern. Compiled regular expression objects support the methods and attributes discussed below.

Match object with the Match template of the re module

A regular expression mapping object to a string always has a boolean value of True. You can check if there was a match using a simple if...else statement. Mapping objects support methods and attributes.

The datetime module

The datetime module provides classes for handling time and date in various ways. The standard way of representing time is also supported, but more emphasis is placed on the ease of manipulation of the date, time and their parts.

Classes provided by the datetime module:

Class `datetime.date(year, month, day)` is a standard date. Attributes: `year`, `month`, `day`. Immutable object.

Class `datetime.time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None)` – standard time, independent of date. Attributes: `hour`, `minute`, `second`, `microsecond`, `tzinfo`.

class is the difference between two points in time, to the nearest microsecond.

class is an abstract base class for time zone information (for example, to account for time zone and/or daylight saving time).

Class `datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None)` - combination of date and time.

Mandatory arguments:

- `datetime.MINYEAR` $(1) \leq \text{year} \leq \text{datetime.MAXYEAR}$ (9999)
- $1 \leq \text{month} \leq 12$
- $1 \leq \text{day} \leq \text{number of days in this month and year}$

Optional:

- $0 \leq \text{minutes} < 60$
- $0 \leq \text{second} < 60$
- $0 \leq \text{microsecond} < 1000000$

Datetime class methods:

`datetime.today()` – a datetime object from the current date and time. Works like `datetime.now()` with `tz=None`.

`datetime.fromtimestamp(timestamp)` - a date from a standard representation of time.

`datetime.fromordinal(ordinal)` – a date from a number that is the number of days that have passed since 01/01/1970.

`datetime.now(tz=None)` – a datetime object from the current date and time.

`datetime.combine(date, time)` – a datetime object from a combination of date and time objects.

`datetime.strptime(date_string, format)` – converts a string to a datetime (same as the `strptime` function from the `time` module).

`datetime.date()` – date object (with time cutoff).

`datetime.time()` – time object (with date cutoff).

`datetime.replace([year[, month[, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]]]])` – returns a new datetime object with changed attributes.

`datetime.timetuple()` - Returns a `struct_time` from a datetime.

`datetime.toordinal()` – the number of days that have passed since 01/01/1970.

`datetime.timestamp()` – returns the time in seconds since the beginning of the epoch.

`datetime.weekday()` – day of the week as a number, Monday – 0, Sunday – 6.

`datetime.isoweekday()` – day of the week as a number, Monday – 1, Sunday – 7.

`datetime.isocalendar()` – tuple (year in ISO format, ISO week number, ISO day of the week).

`datetime.isoformat(sep='T')` is a nice string of the form "YYYY-MM-DDTHH:MM:SS.mmmmmm" or, if `microsecond == 0`, "YYYY-MM-DDTHH:MM:SS".

Module sys

The `sys` module provides access to some variables and functions that interact with the python interpreter.

`sys.argv` – list of command line arguments passed to the Python script.
`sys.argv[0]` is the name of the script (an empty string in an interactive shell).

`sys.byteorder` – byte order. It will have the value 'big' if the order of passing bits is from the oldest to the youngest, and 'little' if it is the other way around (the youngest byte is the first).

`sys.builtin_module_names` is a string tuple containing the names of all available modules.

`sys.call_tracing(function, arguments)` - calls the function with arguments and tracing enabled while tracing is enabled.

`sys.copyright` - A string containing copyrights relating to the Python interpreter.

`sys._clear_type_cache()` - clears the internal type cache .

`sys._current_frames()` - returns a dictionary representation of the ID for each thread in the top frame of the stack currently in that thread at the time the function is called.

`sys.dllhandle` is an integer specifying the Python (Windows) DLL handle.

`sys.exc_info()` - Returns a tuple of three values that provide information about the exceptions currently being handled.

`sys.exec_prefix` – Python installation directory.

`sys.executable` is the path to the Python interpreter.

`sys.exit([arg])` - exit from Python. Throws a `SystemExit` exception that can be caught.

`sys.flags` – command line flags. Attributes are read-only.

`sys.float_info` – information about the float data type.

`sys.float_repr_style` – information about the use of the built-in `repr()` function of the float type.

`sys.getdefaultencoding()` – returns the used encoding.

`sys.getdlopenflags()` – flag values for `dlopen()` calls.

`sys.getfilesystemencoding()` – returns the file system encoding.

`sys.getrefcount(object)` – returns the number of references to the object. The `hitcount` function argument is another reference to the object.

`sys.getrecursionlimit()` – returns the recursion limit.

`sys.getsizeof(object[, default])` – returns the size of the object (in bytes).

`sys.getswitchinterval()` - thread switching interval.

`sys.getwindowsversion()` - Returns a tuple describing the version of Windows.

`sys.hash_info` – information about hashing parameters.

`sys.hexversion` – python version as a hexadecimal number (for 3.2.2 final this will be 30202f0).

`sys.implementation` - an object containing information about the running python interpreter.

`sys.int_info` – information about the int type.

`sys.intern(string)` - returns the intern string.

`sys.last_type`, `sys.last_value`, `sys.last_traceback` - information about processed exceptions. Its content resembles `sys.exc_info()`.

`sys.maxsize` – the maximum value of a number of type `Py_ssize_t` (231 on 32-bit and 263 on 64-bit platforms).

`sys.maxunicode` – the maximum number of bits for storing a Unicode character.

`sys.modules` – a dictionary of names of loaded modules. We are changing it, so you can have fun :)

`sys.path` - a list of module search paths.

`sys.path_importer_cache` – dictionary-cache for searching objects.

`sys.platform` – information about the operating system.

`sys.prefix` – python interpreter installation folder.

`sys.ps1`, `sys.ps2` - primary and secondary request of the interpreter (defined only if the interpreter is in interactive mode). By default, `sys.ps1 == ">>>"` and `sys.ps2 == "..."`.

`sys.dont_write_bytecode` - if true, python will not write `.pyc` files.

`sys.setdlopenflags(flags)` – set flag values for `dlopen()` calls.

`sys.setrecursionlimit(limit)` - set the maximum depth of recursion.

`sys.setswitchinterval(interval)` - Set the thread switching interval.

`sys.settrace(tracefunc)` – set the "trace" of the function.

`sys.stdin` is standard input.

`sys.stdout` – standard output.

`sys.stderr` – standard error stream.

`sys.__stdin__`, `sys.__stdout__`, `sys.__stderr__` - output values of input, output and error streams.

`sys.tracebacklimit` – the maximum number of traceback levels.

`sys.version` – python version.

`sys.api_version` – C API version.

`sys.version_info` – A tuple containing the five components of the version number.

`sys.warnoptions` – implementation of warnings.

`sys.winver` is the python version number used to generate the Windows registry.

A NumPy module

NumPy is an extension of the Python language that adds support for large multidimensional arrays and matrices, along with a large library of high-level math functions for operations on these arrays.

NumPy's main object is a uniform multidimensional array (numpy is called `numpy.ndarray`). This is a multidimensional array of elements (usually numbers) of the same type.

The most important attributes of `ndarray` objects:

`ndarray.ndim` - the number of dimensions (they are more often called "axes") of the array.

`ndarray.shape` – dimensions of the array, its shape. This is a tuple of natural numbers showing the array length of each axis. For a matrix with n rows and m columns, shape will be (n,m) . The number of elements of the shape tuple is `ndim`.

`ndarray.size` – the number of array elements. Obviously, is equal to the product of all elements of the shape attribute.

`ndarray.dtype` – an object that describes the type of array elements. You can define `dtype` using standard Python data types. NumPy here provides a whole bunch of built-in capabilities, for example: `bool_`, `character`, `int8`, `int16`, `int32`, `int64`, `float8`, `float16`, `float32`, `float64`, `complex64`, `object_`, and the ability to define your own data types, including composition.

`ndarray.itemsize` – the size of each element of the array in bytes.

`ndarray.data` is a buffer containing the actual elements of the array. Usually you don't need to use this attribute, because the easiest way to access array elements is by using indexes.

There are many ways to create an array in NumPy. One of the easiest is to create an array from ordinary Python lists or tuples using the `numpy.array()` function (array is a function that creates an object of type `ndarray`):

```
>>> import numpy as np
>>> a = np.array([1, 2, 3])
>>> a
array([1, 2, 3])
>>> type(a)
<class 'numpy.ndarray'>
```

The `array()` function transforms nested sequences into multidimensional arrays. The type of elements of the array depends on the type of elements of the original sequence (can be overridden at the time of creation).

```
>>> b = np.array([[1.5, 2, 3], [4, 5, 6]])
>>> b
array([[ 1.5,  2.,  3. ],
       [ 4.,  5.,  6.]])
```

You can also override the type at creation time:

```
>>> b = np.array([[1.5, 2, 3], [4, 5, 6]], dtype=np.complex)
>>> b
array([[ 1.5+0.j,  2.0+0.j,  3.0+0.j],
       [ 4.0+0.j,  5.0+0.j,  6.0+0.j]])
```

In particular, the `numpy.random` module is used to create arrays with random elements.

The easiest way to specify an array with random elements is to use the `sample` function (or `random`, or `random_sample`, or `ranf` is the same function).

```

>>> np.random.sample()
0.6336371838734877

>>> np.random.sample(3)
array([ 0.53478558, 0.1441317, 0.15711313])

>>> np.random.sample((2, 3))
array([[ 0.12915769, 0.09448946, 0.58778985],
       [ 0.45488207, 0.19335243, 0.22129977]])

```

Without arguments, it returns a simple number in the interval [0, 1), with one integer - a one-dimensional array, with a tuple - an array with the dimensions specified in the tuple (all numbers are from the interval [0, 1)).

You can create an array of integers using the `randint` or `random_integers` function. Arguments: `low`, `high`, `size`: from which, to which number (`randint` does not include this number, but `random_integers` does), and `size` - array sizes.>>>

```

np.random.randint(0, 3, 10)

array([0, 2, 0, 1, 1, 0, 2, 2, 2, 0])
>>> np.random.random_integers(0, 3, 10)
array([2, 2, 3, 3, 1, 1, 0, 2, 3, 2])
>>> np.random.randint(0, 3, (2, 10))
array([[0, 1, 2, 0, 0, 0, 1, 1, 1, 2],
       [0, 0, 2, 2, 2, 0, 1, 2, 2, 1]])

```

String module

The `string` module contains various tools for working with strings, many of which duplicate the capabilities of the standard `str` type. The `string` module also contains a set of constants:

```

string.ascii_lowercase # 'abcdefghijklmnopqrstuvwxyz'
string.ascii_uppercase # 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
string.ascii_letters
# 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
string.digits # '0123456789'
string.punctuation # !"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~

```

In the **string module**, among others, the `capwords()` function is defined, which converts the first letter of each word in the string to upper case.

```

import string
s = 'It was nice talking to you! Thank you!'
print(s)
print(string.capwords(s))

```

The result of the function will be as follows:

```
It was nice talking to you! Thank you!  
It Was Nice Talking To You! Thank you!
```

string module defines a number of constants related to the ASCII table and character sequences.

The string module contains several types of tools, such as functions, methods, and classes. It also contains the most general string constants.

Functions

`index (s, sub, i=0)`

`lower (s)`

`splitfields (s, sep)`

`joinfields (<words>, <distributor>)`

`strip(s)`

`upper (s)`

Result

returns the index of the first occurrence of the substring sub in string s, starting at position i.

returns the string s in lowercase letters.

returns a list of substrings of s strings separated by the sep character.

concatenates a list of <words> using <separator>.

returns the string obtained from s by eliminating blanks.

returns the string s in uppercase letters.

Matplotlib: Scientific Graphics in Python

Matplotlib is a Python library for data visualization, plotting functions and equations. Matplotlib is used in scientific research and conferences to display the obtained data.

To build graphs, you need to import the Pyplot module. Pyplot is a module for working with graphs in Python.

Installation:

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


To graph a Python function, you need to specify the function itself. It can be set using the lambda function. A lambda function is a short way to write a regular function in one line.

Consider the construction of a sinusoid in Python $f(x) = \sin(x)$:

```
y = lambda x: np.sin(x)
```

y is the designation of the function, lambda is the keyword that means the beginning of the task of the lambda function, x is the argument used in the function, after the colon the function is specified. Since standard Python does not have a function that returns the sine of x, it is given by m using NumPy.

To plot a Python function, you must first set the coordinate grid using the `plt.subplots()` command.

```
fig = plt.subplots()
```

Linspace is used to define the range of values on which the graph of the function is built in Python:

```
x = np.linspace(-3, 3, 100)
```

In this example, `linspace` creates an array with a lower bound of -3 and an upper bound of 3, the created array will have 100 elements. The larger the last number, the more values the function will calculate, the more accurately the plot will be displayed in Python.

To plot a Python function directly, use the `plt.plot(x, y(x))` command, where x is an argument and y(x) is a function of x given by a lambda expression. Once the plot is constructed, it must be displayed using the `plt.show()` command.

Complete code for a Python program to display a graph of a function:

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

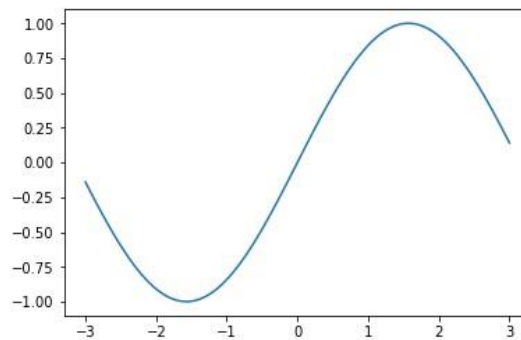
y = lambda x: np.sin(x)

fig = plt.subplots()

x = np.linspace(-3, 3, 100)

plt.plot(x, y(x)) plt.show()
```

As a result, a sine wave graph is formed in a separate window.



Display multiple graphs in one area in Python

You can display the graphs of several functions in one area of Python . Let's add `aeYrwb`. $y=x$ and display it together with the sinusoid.

To do this, we will introduce another function using `lambda`.

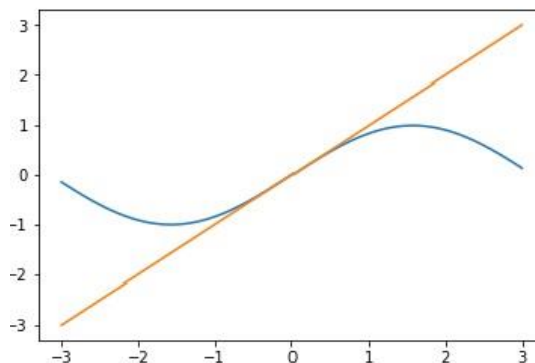
```
y1=lambda x: x
```

Let's plot the graph of this function

```
plt.plot(x,y1(x))
```

A program for constructing graphs of two functions in one window:

```
import numpy as np
import matplotlib.pyplot as plt
y1=lambda x: np.sin(x)
y2=lambda x: x
x = np.linspace(-3, 3,100)
plt.plot(x, y1(x))
plt.plot(x, y2(x))
plt.show()
```



3D surfaces in Python

In three-dimensional space, two arguments are required for a function assignment.

Let's set, for example, a function of two arguments:

$$f(x,y)=x^2-y^2.$$

In accordance:

```
f = lambda x, y: x ** 2 - y ** 2
```

First, the plot area is defined using the `plt.figure` function, in which the `figsize(x, y)` parameter defines the width and height of the figure in inches.

Definition of the area:

```
fig = plt.figure(figsize = (12, 6))
```

Let's create a picture in which the three-dimensional space with coordinate axes and the surface itself will be displayed. For this, `fig.add_subplot()` is used.

```
ax = fig.add_subplot(1, 1, 1, projection = '3d')
```

The Python function `fig.add_subplot()` divides the plot area into cells and specifies in which cell to draw a 3D graph.

The command `ax = fig.add_subplot(1, 1, 1, projection = '3d')` divides the plot area into two cells and the first cell will display a three-dimensional plot thanks to the argument `projection = '3d'`.

Let's enter the display areas of the function for each argument:

```
xval = np.linspace(-5, 5, 100)
```

```
yval = np.linspace(-5, 5, 100)
```

To create a surface, use:

```
surf = ax.plot_surface(x, y, z, rstride = 4, cstride = 4, cmap =  
cm.plasma)
```

Here, `x` and `y` are arguments, `z` is the objective function, `rstride` and `cstride` are responsible for the draw step. The smaller these values are, the smoother the gradient will look on the surface. Due to the `cmap.plasma` value, the surface will be displayed in the plasma color scheme. There are other color schemes such as `viridis` and `magma`.

An example of a program for building a surface in three-dimensional space:

```
from mpl_toolkits.mplot3d import Axes3D  
import numpy as np  
from matplotlib import cm  
import matplotlib.pyplot as plt
```

```
f = lambda x, y: x ** 2 - y ** 2
```

```

fig = plt.figure(figsize = (10, 10))

ax = fig.add_subplot(1, 1, 1, projection = '3d')

xval = np.linspace(-4, 4, 100)yval = np.linspace(-4, 4, 100)

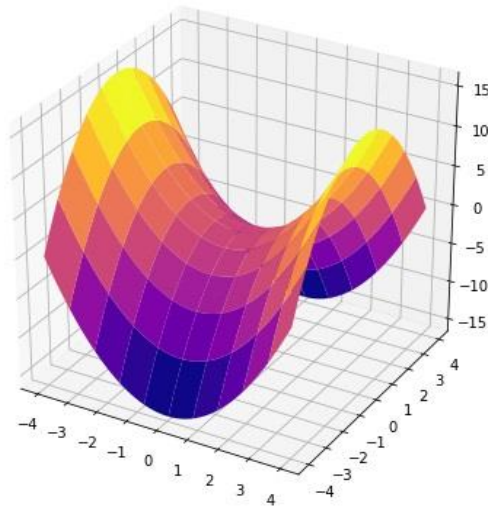
x, y = np.meshgrid( xval, yval)

z = f(x, y)

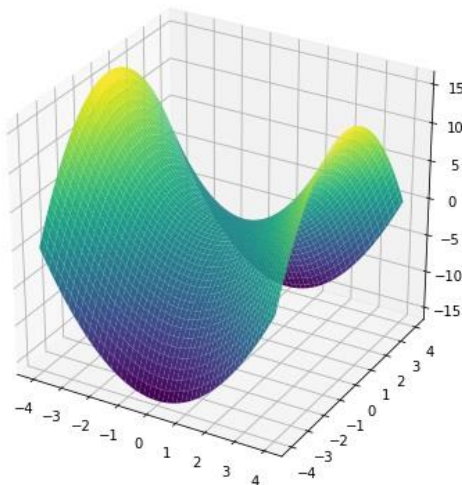
surf = ax.plot_surface(x, y, z, rstride = 10,cstride = 10, cmap =
cm.plasma)

```

We get a graph of the three-dimensional surface in the plasma color gamut in a special window.



When changing the parameters `rstride = 2`, `cstride = 2`, `cmap = cm.viridis`, the graph of the three-dimensional surface in Python is more accurate and in a different color scheme.



PyWavelets : Wavelet transforms in Python

PyWavelets is an open source wavelet transform software for Python. The basic idea of the wavelet transformation corresponds to the specificity of many time series that demonstrate the evolution in time of their main characteristics - the average value, dispersion, periods, amplitudes and phases of harmonic components. The vast majority of processes studied in various fields of knowledge have the features listed above.

The continuous wavelet transform (CWT) is performed by calling the function:

```
pywt.cwt(data, scales, wavelet)
```

Parameters:

data : input signal;

scales : wavelet scale;

wavelet : the name of the wavelet

Subject index

Apache Hadoop 86, 88, 89
Apache Spark 88
API 10, 27, 35, 47, 123
Atom 12
Big Data 8, 9, 11
CGI (Common Gateway Interface) 40
CDH (Cloudera Hadoop) 89
Cloudera 88, 89
Cloudera Manager 88,89
CRUD 35, 119, 120
CSV 57, 58, 79, 83, 114, 115
cURL 15, 16, 31, 32
Cypher 129, 132
Data Science 9, 10
DFS (Distributed file system) 87, 90
Elastic 26, 30
Elastic Stack 9, 10, 26, 27
Elasticsearch 21, 26, 27, 30, 37
Excel 58
Gephi 78, 80, 81
GUI 117, 118
Hadoop 78, 80, 81
Hadoop Common 87
Hadoop YARN 87
HDFS (Hadoop Distributed Filesystem) 87
JSON (JavaScript Object Notation JavaScript) 19
Jupyter Lab 61, 62
Kibana 26, 45
Kibana Console 47
Kibana Discover 49
Kibana Visualizations 51
Mapper 98, 100
MapReduce 86, 87, 88
MHAT Wavelet 68
MongoDB 110, 117
MongoDB Compass 117, 118
Morle Wavelet 68
Neo4j 123, 124, 127
Neo4j Browser 123, 127, 132
NoSQL 111
OSINT (Open Source Intelligence) 8
Python 23, 37, 135
RDF (Resource Description Framework) 12
Reducer 98
REST API 47
RSS 11, 12
Wavelets 66, 67, 68, 69
WAVE Wavelet 68
Wget 16
WinPython 61
WordCount 97, 100, 103
XML 11, 12
XGET 33, 55
X-Pack 26, 45
XPOST 31, 32
YAML Ain't Markup Language 29

Educational edition

Dmytro Volodymyrovych **Lande**

Ihor Yuriyovych **Subach**

Anatoly Yasonovych **Gladun**

**PROCESSING OF EXTREMELY LARGE
DATA SETS (BIG DATA)**

Tutorial

In the author's edition

Institute of Special Communication and Information Protection
National Technical University of Ukraine
"Ihor Sikorsky Kyiv Polytechnic Institute"

Sub to be printed on 12/30/2021. Format 60x84 ^{1/16}. Office paper. Times headset.
The printing method is risography. Mind. printing. sheet 10.5. Region-issue sheet 24,36.
Circulation of 100 pr.
Deputy No. 15-200.

LLC "Engineering" ISBN 978-966-2344-83-7