



Expert Service

Hands On Advanced Analytics with Apache Spark

Curs I

Despre Traineri



Nume	Valentin Rosca
Poziție	Lead Data Engineer -» Smart Meter Analytics
Tehnologii de lucru	Python, SQL, Apache Spark, Databricks
Hobbyuri	Coding, Smart Home, Sport, Printare 3D, Boardgames

Despre Traineri



Nume	George Raut
Poziție	Data Engineer
Tehnologii de lucru	SQL, Python, Grafana, Power BI
Hobbyuri	Escalada, tenis de camp

Despre Traineri



Nume	Iuliana Lidia Sarchis
Poziție	Data Analytics Engineer
Tehnologii de lucru	SQL, Azure Cloud, Power BI

#Data

Ce sunt datele?



Expert
Services

Ce sunt Datele

În fiecare moment, fiecare acțiune, interacțiune și decizie pe care o luăm generează date.



De la ceasurile inteligente
care ne monitorizează pașii



la rețelele sociale unde împărtășim
gânduri și momente



până la senzorii urbani care
colectează informații despre mediu

Toate sunt surse de date

Acestea captează o imagine digitală a realității, oferindu-ne perspective și înțelegeri noi în lumea în care trăim. Datele sunt limbajul universal prin care povestim, înțelegem și îmbunătățim lumea înconjurătoare.

Seturi de Date

Când momentele și interacțiunile se adună, ele formează seturi de date - colecții vaste de informații care ne ajută să înțelegem și să modelăm lumea.



De la activitatea noastră personală și a celor din jur acumulată în timp



la multiple experiențele împărtășite de comunități pe rețele sociale



și până la datele ambientale colectate de diverși senzori urbani

Toate sunt seturi de date

Fiecare secvență de timp și interacțiune contribuie la crearea de seturi de date complexe și multidimensionale.

Date, Observații și Informații



Date / Observații

Fiecare set de date este o colecție de date / observații. Acestea pot fi orice, o singură înregistrare, un grup de înregistrări, chiar și un întreg fișier doar cu text (e.g. Emailuri).

 Ion  19:30  70






 Ion  19:35  65


 Ion  19:40  87

 Bea  11:30  57



 Bea  11:45  60

 Bea  12:15  59













 Ion  #1021   Super
 Vali  #1022   Nice

 Ion  #1122 

 Ion  #1130  Sus

 Ion  #1123  +1

 Bea  #1121 

 Ion  #1160   +2
 Ion  #1123    +2
 Vali  #1121  Dc?

 19:30  #112  83%

 19:30  #113  81%

 19:45  #113  81%

 20:00  #113  76%

 20:15  #112  78%

 20:30  #114  22%

 21:00  #112  50%

Informație

Fiecare dată / observație conține informație, fie ea relevantă sau nu.

Small Data vs Big Data

My data is bigger than yours






























































Expert
Services



Date, Observații și Informații în Big Data

Date / Observații

În big data, numărul de observații este de regula foarte foarte mare.

								
 Ion	 19:30	 70	 Ion	 #1021	  Super	 19:30	 #112	 83%
 Vali	 #1022	  Nice	 Vali	 #1022	  Nice	 19:30	 #113	 81%
 Ion	 19:35	 65	 Ion	 #1122		 19:45	 #113	 81%
...				
 Ion	 19:40	 87	 Ion	 #1130	 Sus	 20:00	 #113	 76%
...				
 Bea	 12:15	 59	 Ion	 #1160	  +2	 21:00	 #112	 50%
 Ion	 #1123	  +2						
			 Vali		 #1121		 Dc?	

Informație

În big data, și informația dintr-o observație poate fi mare, de exemplu un întreg fișier text cu milioane de rânduri.

Small Data vs Big Data

Small Data

Datele sunt suficient de mici pentru a fi stocate și procesate pe dispozitive cu capacitate limitată.



Sunt adesea structurate într-un format simplu și intuitiv, ușor accesibile pentru analiză.

Pot fi manipulate și înțelese fără instrumente sofisticate de analiză a datelor.



La limită

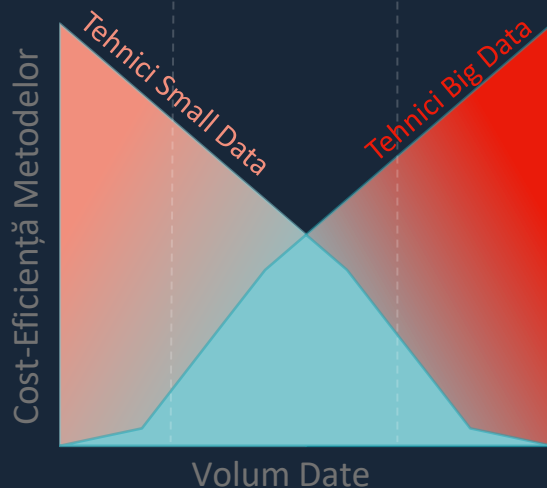
Big Data

Datele necesită infrastructuri avansate pentru stocare și procesare, depășind capacitatea dispozitivelor obișnuite.



Sunt adesea complexe și variate, necesitând formate și tehnologii avansate pentru analiză.

Necesită instrumente sofisticate și tehnici avansate de analiză pentru a fi interpretate corect.



#Big Data

Dacă datele sunt noul petrol, atunci suntem pe punctul de a deveni magnații datelor!



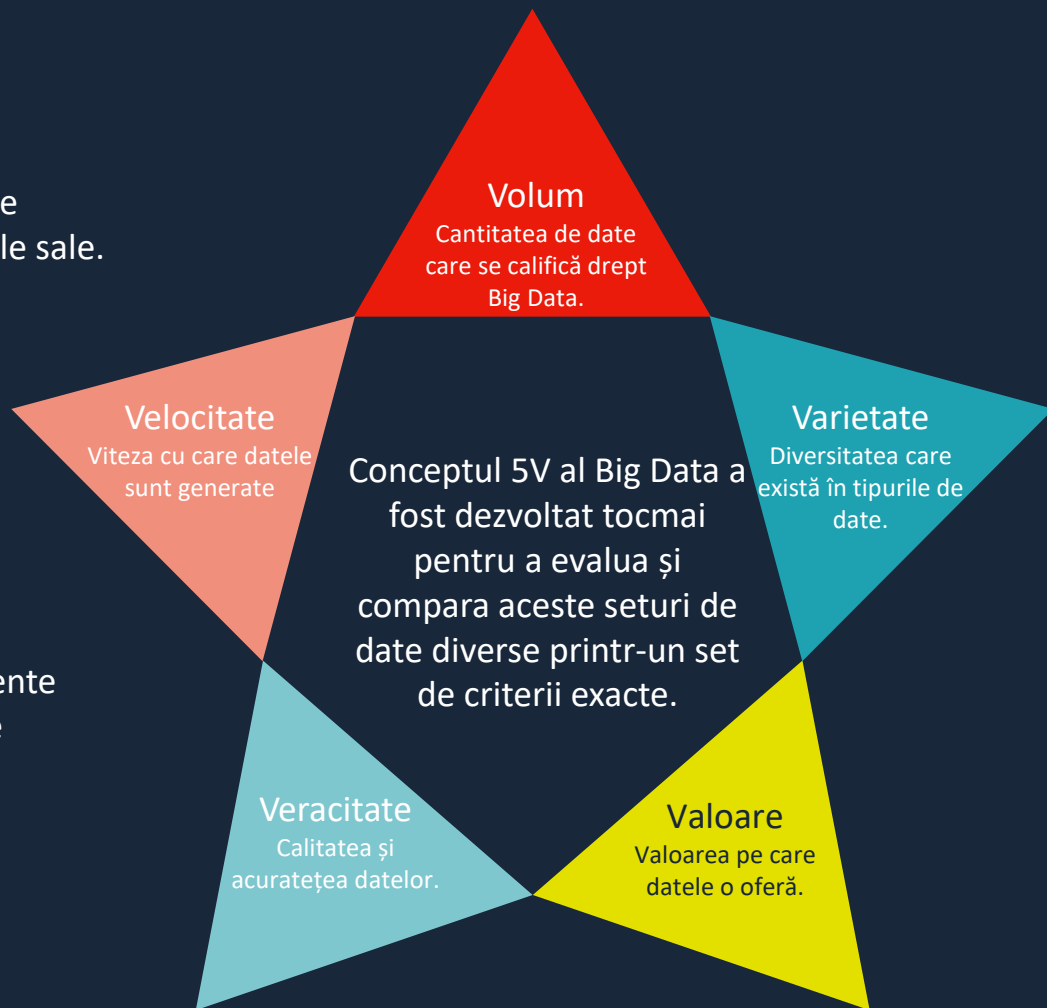
Expert
Services



Big Data 5V

Big Data cuprinde o diversitate de tipuri de seturi de date, fiecare având caracteristicile sale.

- Unele surse pot fi voluminoase, dar se acumulează lent.
- Altele, se disting prin varietatea lor bogată, incluzând o gamă largă de formate precum text și imagini.
- Date precum fișierele log sunt abundente și se extind rapid, dar adesea oferă de multe ori valoare limitată.



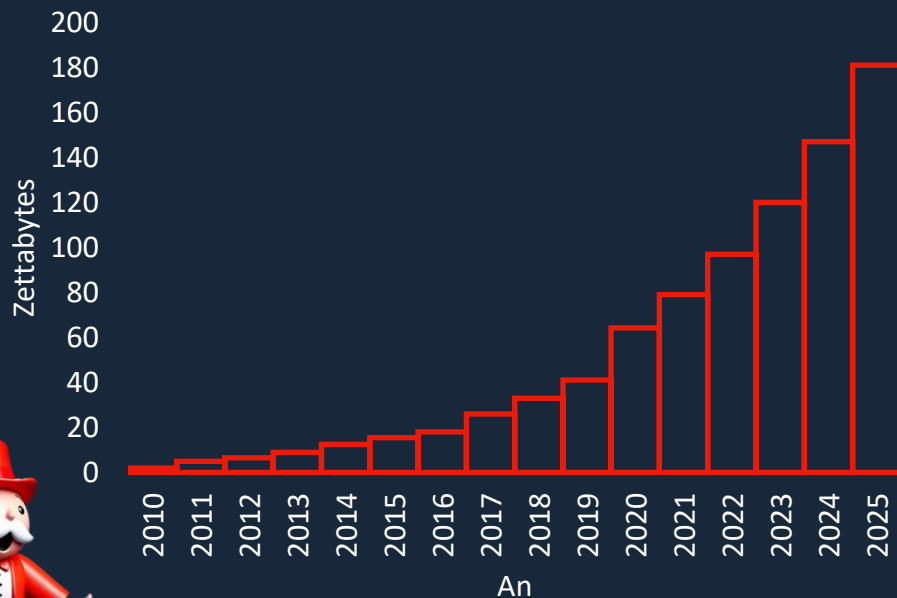
Big Data: O revoluție în date

Big Data se referă la seturi de date atât de mari și complexe încât devin dificil de procesat. Această revoluție nu înseamnă doar un volum enorm de informații, ci și viteza cu care sunt generate.

În graficul alăturat, puteți observa creșterea exponențială a volumului de date generate anual în lume.



Volum Date Generate Anual



+ Volum

+ Viteza



Big Data - Principalele surse

Social Data

Machine Data

Transactional Data



Big Data - Avantaj Competitiv in Business



<https://www.techtarget.com/searchdatamanagement/definition/5-Vs-of-big-data>



Big Data: Infrastructură

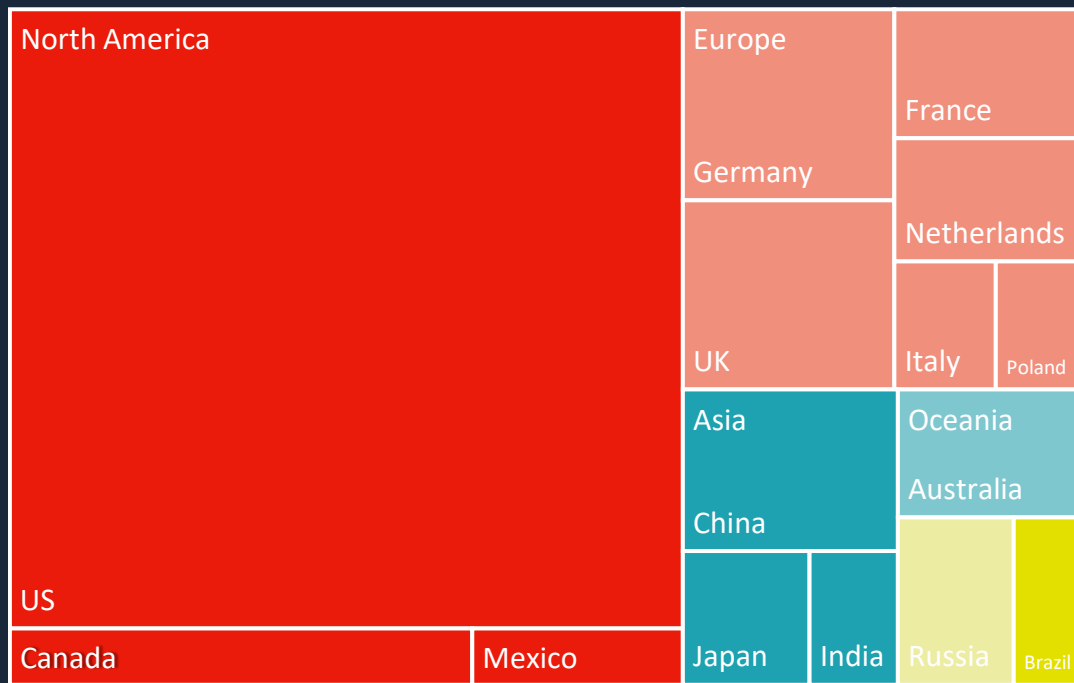
Un alt aspect crucial în înțelegerea Big Data este infrastructura - centrele de date. Aceste centre de date sunt nucleul unde informațiile sunt procesate, stocate și gestionate.



Plotul prezentat aici evidențiază distribuția centrelor de date la nivel global, concentrându-se pe top 15 regiuni și țări. Această reprezentare ne oferă și o perspectivă asupra locurilor în care sunt generate cele mai multe date.

Distribuția centrelor de date

■ North America ■ Europe ■ Asia ■ Other ■ Oceania ■ Other

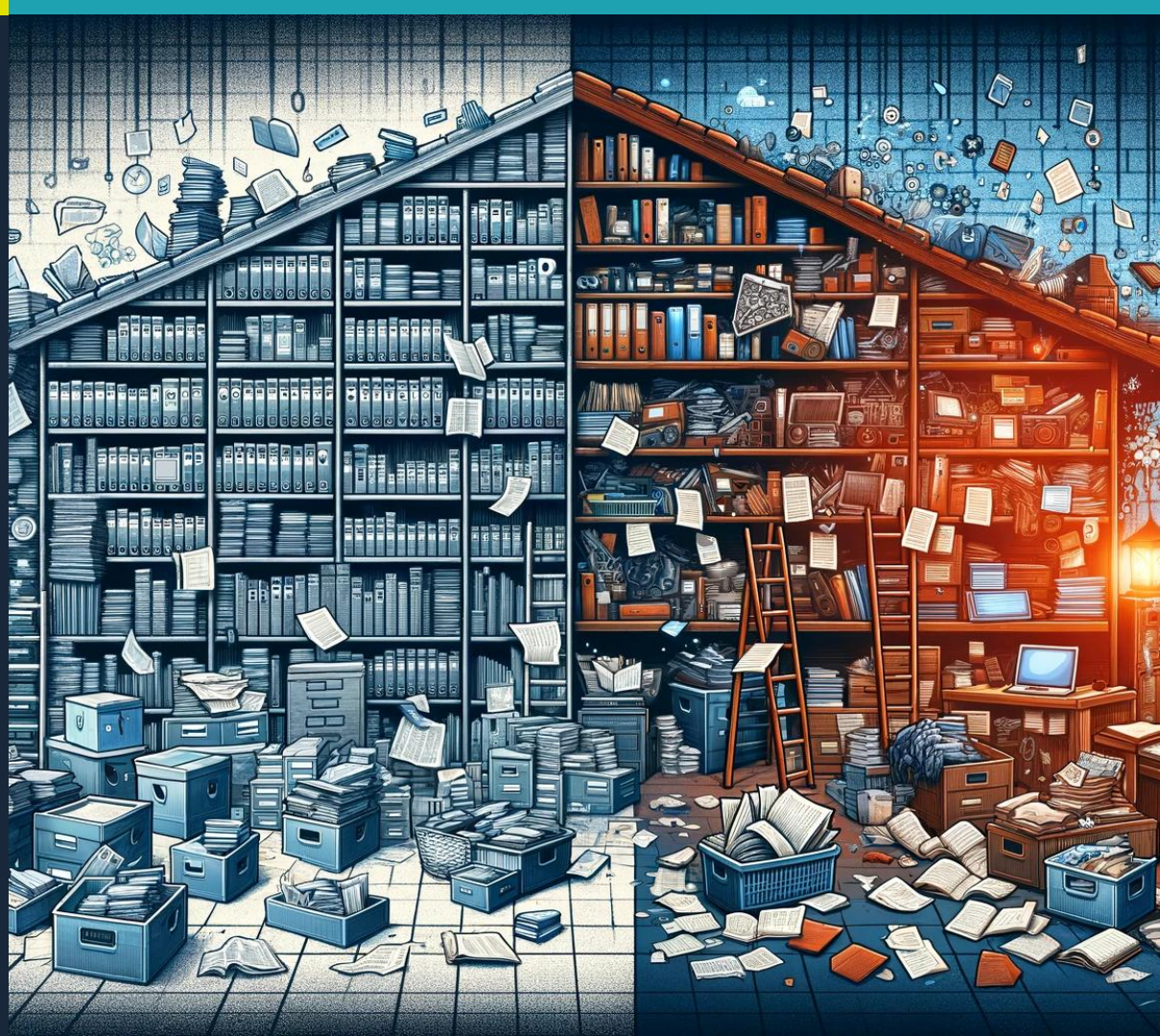


+ Varietate

+ Valoare

Structura Datelor

Încă nu ai curățat datele?



Expert
Services

Structura datelor

Informațiile în date se ascund în forme diverse, fiecare cu propria sa poveste, asemenea unui puzzle complex.

1010
1010



Structura și informația care ne este oferită nu este clară, de multe ori plină de ambiguitate.

Data nestructurate



Informațiile au aceeași structură clară și precisă, fără ambiguitate, asemenea unui tabel bine definit.

Date semi-structurate



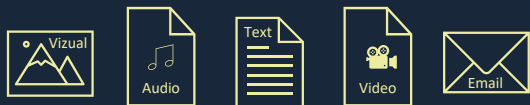
Date structurate

Tipuri de date

Date nestructurate

1010
1010

- structură neclară
- poate lua orice formă
- greu de indexat și căutat



Date semi-structurate



- structură parțial definită
- intrări cu etichete
- mai dificil de indexat și căutat



Date structurate



- structură clară și precisă
- tabel cu rânduri și coloane
- ușor de indexat și căutat



Data Lake

Date, Observații și Informații



Date Nestructurate

Universitatea are 5600 de studenți. Ion (număr de identificare: 412121), 18 ani la FEAA. Vali, cu număr de identificare 533312, este la FII și are 22 de ani.

...

Universitatea are 3800 de studenți. Vlad (număr de identificare: 121124), 19 ani la FCI. Bea, cu număr de identificare 851212, este la AC și are 24 de ani.

Date Semi-Structurate

```
{"universitate": [
  {"Id": "412121", "Nume": "Ion",
   "Vârstă": "18", "Facultate":
   "FEAA"},
  {"Id": "533312", "Nume": "Vali",
   "Vârstă": "22", "Facultate": "FII"}
]}
```

...

```
{"universitate": [
  {"Id": "121124", "Nume": "Vlad",
   "Vârstă": "19", "Facultate": "FCI"},
  {"Id": "851212", "Nume": "Bea",
   "Vârstă": "24", "Facultate": "AC"}
]}
```

Date Structurate

Id	Nume	Vârstă	Facultate
412121	Ion	18	FEAA
533312	Vali	22	FII

...

Id	Nume	Vârstă	Facultate
121124	Vlad	19	FCI
851212	Bea	24	AC

Date Nestructurate - Text



Datele nestructurate sunt colecții de date, adesea colecții de fișiere, datele nu au un model pe care îl urmează.

Vali, în vârstă de **27** de ani, ocupă poziția de **Programator** de **4** ani, arătând o pasiune pentru tehnologie și dezvoltare. În afara muncii sale, Vali are un interes deosebit pentru **sport** și **boardgames**, activități care îi oferă relaxare și distracție după orele petrecute în fața calculatorului.

Vlad, la **34** de ani, este un **Instalator** cu **11** ani de experiență în domeniu. Este cunoscut pentru dedicarea și precizia sa în munca manuală. Pentru a menține un echilibru între viața profesională și cea personală, Vlad se dedică **alergării**, activitate ce îi conferă energie și vitalitate.

Bea, de **29** de ani, își exercită talentul narativ ca **Reporter** de **7** ani, fiind apreciată pentru curajul și integritatea sa jurnalistică. Bea este actualmente în **concediu**, optând pentru un respiro bine meritat, ce îi permite să exploreze noi orizonturi și să se reîncarce pentru provocările profesionale viitoare.

Pentru analiza lor, de multe ori sunt necesari algoritmi complecși , e.g. „ChatGPT”, pentru a extrage informații.

Date Semi-Structurate – JSON Lines



Unul dintre cele mai populare și cele mai utilizate formate de date semi-structurate, este formatul JSON, o colecție de perechi **Key** <-> **Value**, unde cheia este întotdeauna un șir de caractere, și valoarea poate fi:

- Număr
- Valoare Binară
- Un alt obiect JSON
- Șir de caractere
- Listă de valori
- Valoarea "null"

În Big Data este folosită extensia acestuia, **JSON Lines**, care permite câte un JSON pe fiecare linie.



Date Semi-Structurate - CSV



Un alt format adesea utilizat, de date semi-structurate, este formatul **CSV**, Comma Separated Values.

Datele / Înregistrările sunt așezate pe rânduri, valorile dintr-o înregistrare fiind despărțite cu “,” - de aici și “Comma” din nume. Opțional, pe primul rând sunt puse numele coloanelor, separate tot cu “,”.

Header Row	->	Nume , Vârstă, Ocupație Post, Ocupație Vechime, Ocupație Concediu, Extra
Value Rows	{	Vali, 27, Programator, 4,, Sport Boardgames
		Vlad, 34, Instalator, 11,, Alergare
		Bea, 29, Reporter, 7, true,

Librăriile de parsare deduc automat tipul de date, dar permit și specificarea tipului de date explicit.

O variație de asemenea întâlnită este formatul TSV, . Tab Separated Values, în care este folosit caracterul de TAB (\t) în loc de “,” – de aici și “Tab” din nume.

Date Structurate – Baze de Date



Datele structurate sunt cele mai familiare tipuri de date, tabele. Fiecare înregistrare de date reprezintă un rând în tabel și pe coloane sunt valorile.

Nume string	Vârstă integer	Ocupație Post string	Ocupație Vechime integer	Ocupație Concediu bool	Extra array<string>
Vali	27	Programator	4	NULL	Sport, Boardgames
Vlad	34	Instalator	11	NULL	Alergare
Bea	29	Reporter	7	true	NULL

Valorile din aceeași coloană vor avea același tip de date, sau valoarea specială „NULL”.

Datele tabelare sunt stocate fie ele în fișiere [Excel](#), sau fie în formatul popular de baze de date [SQL](#). Fiecare format de stocare vine cu propriile sale tipuri de date, dar, în general, toate suportă cel puțin tipurile standard cunoscute de toată lumea.

Date Structurate – Parquet

Cu emergența Big Data, care a sporit cererea pentru o stocare mai simplă a datelor dat fiind că bazele de date sunt complexe de setat, a apărut nevoia de un format de date structurate mai simplu care este și ușor de manageriat.

Așadar, a apărut formatul "Parquet", unde datele sunt așezate în fișier asemenea parchetului dintr-o cameră.

- Mai multe rânduri consecutive sunt grupate împreună într-un **bloc**. În fișier, blocurile se află unul după altul.
- Datele din fiecare bloc sunt stocate coloană după coloană, mai întâi prima coloană, după datele de pe a doua coloană, ș.a.m.d. (nu rând după rând cum este la CSV).

❖ Fiind un format structurat, înainte de blocuri se află numele coloanelor și tipurile lor de date.

Nume	Vârstă	Ocupație			Extra	1
		Post	Vechime	Concediu		
string	integer	string	integer	bool	array<string>	
Vali	27	Programator	4	NULL	Sport, Boardgames	
Vlad	34	Instalator	11	NULL	Alergare	
Bea	29	Reporter	7	true	NULL	
Maria	33	Economist	15	NULL	NULL	
Teo	21	Programator	2	true	Sală	
Victor	24	Programator	0	NULL	Gătit	

Motoare Big Data

Motoarele Big Data sunt uneltele care nu numai ca ne permit sa lucrăm cu seturi de date uriașe, dar adesea suportă și diverse tipuri de date din familiile de date nestructurate, semi-structurate și structurate.



Deși fiecare Motor Big Data vine cu avantajele și dezavantajele lui, unele sunt mai versatile și se pot adapta la majoritatea situațiilor, altele sunt mai specializate și adesea folosite, de multe ori în combinație cu celelalte.

Apache Spark

Not the 🔥 spark



Expert
Services

Istorie



2009

2010

2013

2014



A pornit ca un
proiect in cadrul UC
Berkeley AMPLab

Algorithms, Machines, and
People



Codul sursă a
fost făcut public
sub licența BSD

Berkeley Software
Distribution



Spark a fost donat
catre Apache
Software Foundation

Software for the Public Good



A fost folosit de
Databricks pentru a
sorta un set de date
urias, setând astfel și
un record mondial.

Ce este Apache Spark



Apache Spark este un motor de procesare open-source folosit pentru a stoca și procesa date cu volum mare și / sau în timp real folosind grupuri de calculatoare, clustere, utilizând construcții simple de programare.

Suportă diverse limbaje de programare



Ingineri de date și cercetătorii de date incorporează Spark în aplicații pentru a executa rapid interogări, analize și transformări de date la scară mare



Caracteristici Apache Spark



Procesare rapida



Spark folosește resursele de care dispune cât de eficient poate pentru a prelucra datele cât mai rapid.

Procesare in memorie



In Spark, datele sunt stocate in RAM, prin urmare ele pot fi accesate rapid, mărinnd astfel viteza execuției.

Flexibil



Spark suportă diverse formate de date și mai multe limbaje de programare.

Reziliență la erori



Spark a fost proiectat să gestioneze erori fatale ale mașinilor din cluster fără pierderi de date.

Analize Complexe



Spark dispune de un catalog bogat de operații care pot fi efectuate asupra datelor, fiind capabil de analize complexe.

Componente Apache Spark

Spark Core

Motorul de Bază Spark pentru execuții paralele largi și procesarea datelor distribuite. Toate celelalte componente se bazează pe aceasta.



Managementul memoriei



Recuperare după erori



RDD - Resilient Distributed Datasets



Planificare, distribuire și monitorizare



Integrare cu sisteme de stocare



Lucru cu date nestructurate

Spark SQL

Modulul de Spark care se ocupă de procesarea datelor semi-structurate și structurate. La bază, folosește Spark Core în mod transparent.



Integrare cu diverse baze și formate de date



Data Frames - bazate pe RDD în spate



Analize Complexe

Spark Streaming

Extensie la Spark Core și Spark SQL pentru procesarea datelor în timp real.



Integrare cu diverse sisteme de stocare a fluxurilor de date



Procesarea rapidă a volumelor mari de date în timp real

Spark Mllib

Librărie low-level de învățare automată, simplă de utilizat și scalabilă. Folosește Spark Core la bază.



Clasificarea Datelor



Clusterizarea Datelor



Deep Learning

GraphX

Motorul lui Spark pentru calcul de grafuri și stocarea lor.



Gestionarea datelor de tip graf



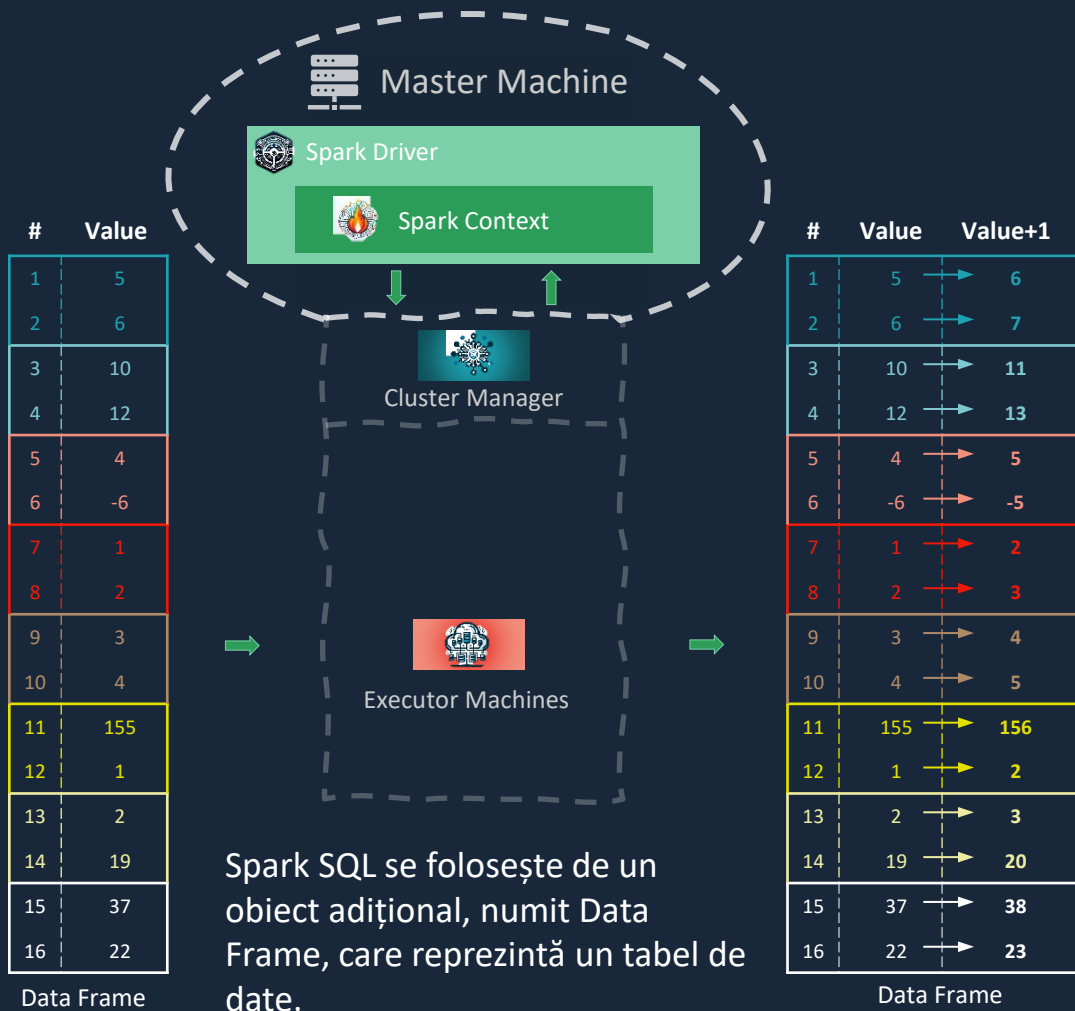
Navigare prin rețele complexe



Arhitectură

Spark folosește o arhitectură care se bazează pe o mașină principală, numită **Master**, care prin intermediul unui serviciu, numit **Cluster Manager**, dă ordine către mașinile care vor executa comenzile, numite **Executor**.

În procesul în care este rulat Spark, numit Spark **Driver**, dezvoltatorul va crea un obiect, fie el în Java, Python, etc., numit Spark **Context**, prin care librăria de Spark poate trimite instrucțiunile către executori.





Expert Service

Practice

PySpark

PySpark este librăria de Apache Spark pentru Python. Este foarte similară cu populara librărie Pandas, oferind o structură similară, asemănătoare cu interogările SQL. Ea a fost concepută pentru a oferi oricărei persoane capabilitatea de a efectua operații complexe pe date.

PySpark oferă:

- Un obiectul de tip Spark Context, folosit pentru interacțiunea cu Spark,.
- Un obiect de bază pentru codul SQL și citirea datelor, numit Spark Session.
- Un obiect de lucru cu date, numit Data Frame, având o interfață similară cu cel de Pandas.
- Funcții gata implementate pentru transformări și expresii.
- Abilitatea de a construi propriile funcții de transformare în Python.

Codul in PySpark este unul foarte simplu și foarte intuitiv, întrucât complexitatea este în ce facem cu datele.

Pregătire mediu de lucru

Pregătire mediu de lucru



<https://colab.research.google.com/>

Conectați-vă

+ Blocnotes nou

Stabilirea conexiunii dintre Google drive si Colab notebook

```
from google.colab import drive  
drive.mount('/content/drive')
```

Pregătire mediu de lucru – Instalarea PySpark in Google Colab



```
!sudo apt update
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
#Check this site for the latest download link https://dlcdn.apache.org/spark/
!wget -q https://dlcdn.apache.org/spark/spark-3.4.4/spark-3.4.4-bin-hadoop3.tgz
!tar xf spark-3.4.4-bin-hadoop3.tgz
!pip install -q findspark
!pip install pyspark
!pip install py4j

import os
import sys
# os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
# os.environ["SPARK_HOME"] = "/content/spark-3.4.4-bin-hadoop3"

import findspark
findspark.init()
findspark.find()

import pyspark
```

```
from pyspark.sql import DataFrame, SparkSession
from typing import List
import pyspark.sql.types as T
import pyspark.sql.functions as F

spark= SparkSession \
    .builder \
    .getOrCreate()

spark
```

Data Frame

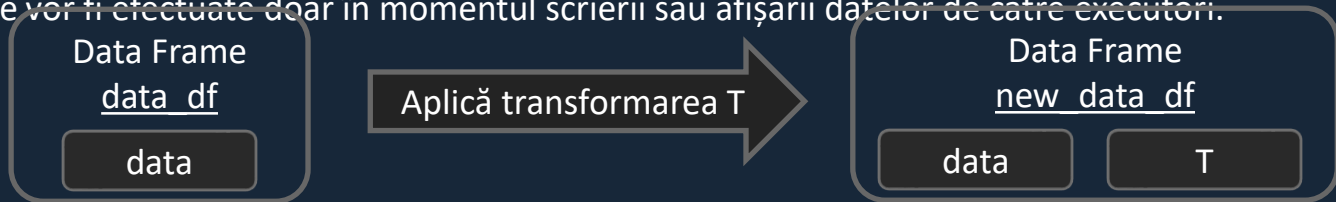
Spark SQL lucrează datele sub forma unui tabel. Bibliotecă PySpark oferă un clasă care reprezintă un tabel de date, unde fiecare coloană conține același tip de date, numită **Data Frame**, similar cu ce oferă bibliotecă Pandas.

- Crearea unui Data Frame Spark dintr-o listă de Python.

```
data = [  
    ['Vali', 23, 'Programator', 4, None, ['Sport', 'Boardgames']],  
    ['Vlad', 34, 'Instalator', 11, None, ['Alergare']],  
    ['Bea', 29, 'Reporter', 7, True, None]  
]  
  
data_df = spark.createDataFrame(data)
```

Fiind concepute pentru Big Data, un **Data Frame** nu încarcă datele. El reține doar locația și tipul lor. La fiecare transformare de date, nou Data Frame va fi creat care reține locația și tipul datelor, precum și lanțul de transformări.

Transformările vor fi efectuate doar în momentul scrierii sau afișării datelor de către executori.



Citirea Datelor

În Big Data, în general, datele sunt stocate extern în unul sau mai multe formate. PySpark oferă metode pentru citirea celor mai populare și des utilizare formate, precum JSON, Parquet sau CSV.

- Citirea fișierelor de tip JSON Lines.

```
data_df = spark.read.format('json').load('/path/to/folder/or/file')
```

- Citirea fișierelor de tip Parquet.

```
data_df = spark.read.format('parquet').load('/path/to/folder/or/file')
```

Putem schimba și setările implicite de citire. De exemplu, implicit, Spark ignoră Header-ul din fișiere CSV.

- Citirea fișierelor de tip CSV cu rând de Header.

```
data_df = spark.read.format('csv').option('header', 'true').load('/path/to/folder/or/file')
```

Dacă avem mai multe setări, adăugăm la lanț câte un apel la funcția `option` pentru fiecare setare. O listă cu toate opțiunile disponibile pentru fiecare format găsiți la <https://spark.apache.org/docs/latest/sql-data-sources.html>.

Schema Datelor

Întrucât Spark SQL a fost conceput pentru date structurate, el încearcă să determine tipurile de coloane, numită schema datelor, în mod automat, din sursele de date la citire. Acest proces nu este necesar pentru formate structurate ca Parquet, dar necesar pentru obiecte Python sau date semi-structurate, JSON și CSV.

- Această schema se poate defini de dezvoltator după un anumit format
- Pentru a dezactiva determinarea automată a tipurilor de date, la citire, se poate furniza schema
- Principalele tipuri de date din Spark (Lista completă: <https://spark.apache.org/docs/latest/sql-ref-datatypes.html>)

❖ integer	❖ string	❖ double	❖ date	❖ map<tip cheie, tip valoare>
❖ boolean	❖ array<tip element>	❖ float	❖ timestamp	❖ struct<coloana1:tip 1, coloana2:tip 2,...>

Schema Datelor

PySpark oferă un și un mod programatic de a defini o schemă a datelor:

```
from pyspark.sql import types as T

data_schema = T.StructType([
    T.StructField('nume', T.StringType(), False),
    T.StructField('varsta', T.IntegerType(), False),
    T.StructField('ocupatie', T.StringType(), False),
    T.StructField('vechime', T.IntegerType(), True),
    T.StructField('inactiv', T.BooleanType(), True),
    T.StructField('extra', T.ArrayType(T.StringType()), True)
])
```

Dacă se furnizează la citire schema, Spark va ignora sau pune NULL pe coloanele care nu se potrivesc sau nu există:

```
data_df = spark.createDataFrame(data, schema=data_schema)
```

```
data_df = spark.read.format('json').schema(data_schema).load('/path/to/folder/or/file')
```

Data Frame – Schema Datelor

Fie prin detectare automată, fie prin furnizarea la citire, fiecare Data Frame va avea întotdeauna o structură a datelor foarte bine definită. Pentru fiecare coloană avem numele, tipul de dată și dacă se permit valori de "NULL".

➤ Pentru a afișa structură a datelor / schema datelor pentru un Data Frame, folosim:

```
data_df.printSchema()
```

```
root
 |-- nume: string (nullable = false)
 |-- varsta: integer (nullable = false)
 |-- post: string (nullable = false)
 |-- vechime: integer (nullable = true)
 |-- concediu: boolean (nullable = true)
 |-- extra: array (nullable = true)
 |     |-- element: string (containsNull = true)
```

Dacă avem de lucrat cu date care nu au o schemă bine definită, fie date nestructurate, fie date semi-structurate cu multe abateri, atunci NU folosim Data Frame-ul din Spark SQL.

Data Frame – Afișarea Datelor

De multe ori ne găsim în situația în care este nevoie să depanăm procesul de transformare a datelor. Așadar, Spark oferă metode pentru acest scop.

➤ Pentru a printa datele la consolă, folosim:

```
data_df.show()
```

```
+---+-----+-----+-----+-----+-----+
|nume|varsta|      post|vechime|concediu|      extra|
+---+-----+-----+-----+-----+-----+
|Vali|    23|Programator|    4|    NULL|[Sport, Boardgames]|
|Vlad|    34|Instalator|   11|    NULL|      [Alergare]|
|Bea|    29|Reporter|    7|    true|      NULL|
+---+-----+-----+-----+-----+-----+
```

Atenție! Datele sunt transferate mai întâi de la executori pe Spark Driver înainte de afișare, așadar această metodă limitează numărul de date afișate pentru a evita prăbușirea mașinii principale.

Data Frame – Colectarea Datelor

Alte metode pe care Spark le oferă, nu numai pentru depănarea datelor, dar și pentru rarele situații când am redus datele și vrem să prelucrăm cu alte librării, sunt cele de colectare a datelor în obiecte de Python pe mașina Master.

- Pentru a colecta datele într-o listă de Python, folosim:

```
data_list = data_df.collect()
```

- Pentru a colecta datele într-un obiect de Data Frame, dar a librăriei Pandas, folosim:

```
data_pandas_pdf = data_df.toPandas()
```

Atenție! Datele sunt transferate mai întâi de la executori pe Spark Driver la colectare. Dacă sunt prea multe date, există pericolul ca procesul de Spark Driver să nu mai facă față și să fie terminat de către sistem.

Scrierea Datelor

În Big Data, după procesare, datele sunt stocate extern în unul sau mai multe formate. PySpark oferă metode pentru scrierea celor mai populare și des utilizare formate.

- Scrierea fișierelor de tip JSON Lines.

```
data_df.write.format('json').save('/path/to/save/folder/')
```

- Scrierea fișierelor de tip Parquet.

```
data_df.write.format('parquet').save('/path/to/save/folder')
```

Putem schimba și setările implicite de citire. De exemplu, implicit, Spark nu scrie Header-ul în fișiere CSV.

- Scrierea fișierelor de tip CSV cu rând de Header.

```
data_df.write.format('csv').option('header', 'true').save('/path/to/save/folder')
```

Dacă avem mai multe setări, adăugăm la lanț câte un apel la funcția `option` pentru fiecare setare. O listă cu toate opțiunile disponibile pentru fiecare format găsiți la <https://spark.apache.org/docs/latest/sql-data-sources.html>.



Expert Service

Practice Session