

17-SparkDataFrames

August 11, 2020

1 Spark DataFrames

- Enable wider audiences beyond “Big Data” engineers to leverage the power of distributed processing
- Inspired by data frames in R and Python (Pandas)
- Designed from the ground-up to support modern big data and data science applications
- Extension to the existing RDD API

1.1 References

- [Spark SQL, DataFrames and Datasets Guide](#)
- [Introduction to DataFrames - Python](#)
- [PySpark Cheat Sheet: Spark DataFrames in Python](#)

1.1.1 DataFrames are :

- The preferred abstraction in Spark
- Strongly typed collection of distributed elements
- Built on Resilient Distributed Datasets (RDD)
- Immutable once constructed

1.1.2 With Dataframes you can :

- Track lineage information to efficiently recompute lost data
- Enable operations on collection of elements in parallel

1.1.3 You construct DataFrames

- by parallelizing existing collections (e.g., Pandas DataFrames)
- by transforming an existing DataFrames
- from files in HDFS or any other storage system (e.g., Parquet)

1.1.4 Features

- Ability to scale from kilobytes of data on a single laptop to petabytes on a large cluster
- Support for a wide array of data formats and storage systems
- Seamless integration with all big data tooling and infrastructure via Spark
- APIs for Python, Java, Scala, and R

1.1.5 DataFrames versus RDDs

- Nice API for new users familiar with data frames in other programming languages.
- For existing Spark users, the API will make Spark easier to program than using RDDs
- For both sets of users, DataFrames will improve performance through intelligent optimizations and code-generation

1.2 PySpark Shell

Run the Spark shell:

pyspark

Output similar to the following will be displayed, followed by a >>> REPL prompt:

```
Python 3.6.5 |Anaconda, Inc.| (default, Apr 29 2018, 16:14:56)
```

[GCC 7.2.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

```
2018-09-18 17:13:13 WARN NativeCodeLoader:62 - Unable to load native-hadoop library for your platform
```

Setting default log level to "WARN".

To adjust logging level use `sc.setLogLevel(newLevel)`. For SparkR, use `setLogLevel(newLevel)`.

Welcome to

```

      _--_
     /  _/  _-   _--_   _--_/_/_-
    _\  \/_-  \/_-  \/_-/_/_/'/_/
   /__/_/_-./_\_,/_/_/_/_-\_\_   version 2.3.1
      //

```

Using Python version 3.6.5 (default, Apr 29 2018 16:14:56)

```
SparkSession available as 'spark'.
```

>>>

Read data and convert to Dataset

```
df = sqlContext.read.csv("/tmp/irmar.csv", sep=';', header=True)
```

```
>>> df2.show()
```

	name	phone	office	organization	position	hdr	team1	team2
0	Alphonse Paul	+33223235223	214	R1	DOC	False	EDP	NA
1	Ammari Zied	+33223235811	209	R1	MC	True	EDP	NA

```

.
.
.
| 18|    Bernier Joachim |+33223237558|    214|          R1|    DOC|False|    ANANUM|    NA|
| 19|    Berthelot Pierre |+33223236043|    601|          R1|    PE| True|    GA|    NA|
+---+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

```

1.3 Transformations, Actions, Laziness

Like RDDs, DataFrames are lazy. Transformations contribute to the query plan, but they don't execute anything. Actions cause the execution of the query.

1.3.1 Transformation examples

- filter
- select
- drop
- intersect
- join ### Action examples
- count
- collect
- show
- head
- take

1.4 Creating a DataFrame in Python

```
[1]: import sys, subprocess
import os

os.environ["PYSPARK_PYTHON"] = sys.executable
```

```
[2]: from pyspark import SparkContext, SparkConf, SQLContext
# The following three lines are not necessary
# in the pyspark shell
conf = SparkConf().setAppName("people").setMaster("local[*]")
sc = SparkContext(conf=conf)
sc.setLogLevel("ERROR")
sqlContext = SQLContext(sc)
```

```
[3]: df = sqlContext.read.json("data/people.json") # get a dataframe from json file

df.show(24)
```

firstname	lastname	login
Simon	Uzel	uzel_s
Perrine	Moreau	moreau_p
Elise	Negri	negri_e
Camille	Cochet	cochet_c
Nolwenn	Giguelay	giguelay_n
Youen	Meyer	meyer_y
Emilie	Lacoste	lacoste_e
Pia	LeBihan	lebihan_p
Yann	Evain	evain_y
Camille	Guyon	guyon_c
Mathilde	LeMener	lemener_m
Gildas	LeGuilly	liguilly_g
Pierre	Gardelle	gardelle_p
Christophe	Boulineau	boulineau_c
Omar	Aitichou	aitichou_o
Lijun	Chi	chi_l
Jiawei	Liu	lin_j
Irvin	Keraudren	keraudren_i
Bryan	Jacob	jacob_b
Raphael	Guillerm	guillerm_r
Bruno	Queguiner	queguiner_b
Yingshi	Zeng	zeng_y

1.5 Schema Inference

In this exercise, let's explore schema inference. We're going to be using a file called `irmar.txt`. The data is structured, but it has no self-describing schema. And, it's not JSON, so Spark can't infer the schema automatically. Let's create an RDD and look at the first few rows of the file.

```
[4]: rdd = sc.textFile("data/irmar.csv")
    for line in rdd.take(10):
        print(line)
```

```
Alphonse Paul;+33223235223;214;R1;DOC;False;EDP;NA
Ammari Zied;+33223235811;209;R1;MC;True;EDP;NA
André Simon;+33223237555;301;R1;DOC;False;THEO-ERG;NA
Angst Jurgen;+33223236519;320;R1;MC;False;PROC-STOC;NA
Bailleul Ismaël;+33223236369;302;R1;MC;True;THEO-ERG;NA
Baker Mark;+33223236028;835;R1;PR;True;GAN;NA
Balac Stephane;+33223236274;110;R1;MC;False;ANANUM;NA
Bauer Max;+33223236675;734;R1;MC;False;GAN;NA
Bavard Juliette;+33223236724;331;CNRS;CR;False;GAN;THEO-ERG
```

Beauchard Karine;+33223236164;235;R1;PR;True;ANANUM;NA

1.6 Hands-on Exercises

You can look at the DataFrames API documentation

Let's take a look to file `"/tmp/irmar.csv"`. Each line consists of the same information about a person:

- name
- phone
- office
- organization
- position
- hdr
- team1
- team2

```
[5]: from collections import namedtuple

rdd = sc.textFile("data/irmar.csv")

Person = namedtuple('Person', ['name', 'phone', 'office', 'organization',
                                'position', 'hdr', 'team1', 'team2'])

def str_to_bool(s):
    if s == 'True': return True
    return False

def map_to_person(line):
    cols = line.split(";")
    return Person(name      = cols[0],
                  phone     = cols[1],
                  office    = cols[2],
                  organization = cols[3],
                  position   = cols[4],
                  hdr        = str_to_bool(cols[5]),
                  team1      = cols[6],
                  team2      = cols[7])

people_rdd = rdd.map(map_to_person)
df = people_rdd.toDF()
```

```
[6]: df.show()
```

```
+-----+-----+-----+-----+-----+-----+
+-----+
|          name|      phone|office|organization|position|  hdr|    team1|
team2|
```

Alphonse Paul	+33223235223	214	R1	DOC	false	EDP
NA						
Ammari Zied	+33223235811	209	R1	MC	true	EDP
NA						
André Simon	+33223237555	301	R1	DOC	false	THEO-ERG
NA						
Angst Jurgen	+33223236519	320	R1	MC	false	PROC-STOC
NA						
Bailleul Ismaël	+33223236369	302	R1	MC	true	THEO-ERG
NA						
Baker Mark	+33223236028	835	R1	PR	true	GAN
NA						
Balac Stephane	+33223236274	110	R1	MC	false	ANANUM
NA						
Bauer Max	+33223236675	734	R1	MC	false	GAN
NA						
Bavard Juliette	+33223236724	331	CNRS	CR	false	
GAN THEO-ERG						
Beauchard Karine	+33223236164	235	R1	PR	true	ANANUM
NA						
Bekka Bachir	+33223235779	307	R1	PR	true	THEO-ERG
NA						
Bekka Karim	+33223236180	615	R1	MC	false	G&S
NA						
Belgacem Maher	+33223236670	NA	EXT	DOC	false	ANANUM
NA						
Bellis Alexandre	+33223236696	634	R1	DOC	false	GAN
NA						
Belmiloudi Aziz	+33223238646	NA	INSA	MC	true	ANANUM
NA						
Ben Elouefi Rim	+33223236670	NA	EXT	DOC	false	STAT
NA						
Benasseni Jacques	+33299141822	NA	R2	PR	true	STAT
NA						
Bennani-Dosse Moh...	+33299141796	NA	R2	MC	false	STAT
NA						
Bernier Joachim	+33223237558	214	R1	DOC	false	ANANUM
NA						
Berthelot Pierre	+33223236043	601	R1	PE	true	GA
NA						

only showing top 20 rows

1.6.1 Schema

```
[7]: df.printSchema()
```

```
root
 |-- name: string (nullable = true)
 |-- phone: string (nullable = true)
 |-- office: string (nullable = true)
 |-- organization: string (nullable = true)
 |-- position: string (nullable = true)
 |-- hdr: boolean (nullable = true)
 |-- team1: string (nullable = true)
 |-- team2: string (nullable = true)
```

1.6.2 display

```
[8]: display(df)
```

```
DataFrame[name: string, phone: string, office: string, organization: string, position: string,
```

1.6.3 select

```
[9]: df.select(df["name"], df["position"], df["organization"])
```

```
[9]: DataFrame[name: string, position: string, organization: string]
```

```
[10]: df.select(df["name"], df["position"], df["organization"]).show()
```

```
+-----+-----+-----+
|          name|position|organization|
+-----+-----+-----+
|    Alphonse Paul|      DOC|          R1|
|    Ammari Zied|      MC|          R1|
|    André Simon|      DOC|          R1|
|    Angst Jurgen|      MC|          R1|
| Bailleul Ismaël|      MC|          R1|
|    Baker Mark|      PR|          R1|
|    Balac Stephane|      MC|          R1|
|    Bauer Max|      MC|          R1|
|    Bavard Juliette|      CR|        CNRS|
| Beauchard Karine|      PR|          R1|
|    Bekka Bachir|      PR|          R1|
|    Bekka Karim|      MC|          R1|
|    Belgacem Maher|      DOC|        EXT|
```

	Bellis Alexandre	DOC	R1
	Belmiloudi Aziz	MC	INSA
	Ben Elouefi Rim	DOC	EXT
	Benasseni Jacques	PR	R2
	Bennani-Dosse Moh...	MC	R2
	Bernier Joachim	DOC	R1
	Berthelot Pierre	PE	R1

+-----+-----+-----+-----+

only showing top 20 rows

1.6.4 filter

```
[11]: df.filter(df["organization"] == "R2").show()
```

team1 team2	name	phone	office	organization	position	hdr	
-------------	------	-------	--------	--------------	----------	-----	--

+-----+-----+-----+-----+-----+-----+-----+-----+

	Benasseni Jacques	+33299141822	NA	R2	PR	true	STAT
	Bennani-Dosse Moh...	+33299141796	NA	R2	MC	false	STAT
	Cornillon Pierre-...	+33299141819	NA	R2	MC	false	STAT
	Fromont Magalie	+33299053264	NA	R2	PR	true	STAT
	Giacofci Joyce Ma...	+33299141800	NA	R2	MC	false	STAT
	Klutchnikoff Nicolas	+33299141819	NA	R2	MC	false	STAT
	Le Guevel Ronan	+33299141800	NA	R2	MC	false	PROC-STOC
	Mom Alain	+33299141808	NA	R2	MC	false	STAT
	Morvan Marie	+33223236670	NA	R2	DOC	false	STAT
	Pelletier Bruno	+33299141807	NA	R2	PR	true	STAT
	Rouviere Laurent	+33299141804	NA	R2	MC	false	STAT

+-----+-----+-----+-----+-----+-----+-----+-----+

1.6.5 filter + select

```
[12]: df2 = df.filter(df["organization"] == "R2").select(df['name'],df['team1'])
```

```
[13]: df2.show()
```

```
+-----+-----+
|          name|   team1|
+-----+-----+
| Benasseni Jacques|   STAT|
|Bennani-Dosse Moh...|   STAT|
|Cornillon Pierre-...|   STAT|
|   Fromont Magalie|   STAT|
|Giacofci Joyce Ma...|   STAT|
|Klutchnikoff Nicolas|   STAT|
|   Le Guevel Ronan|PROC-STOC|
|       Mom Alain|   STAT|
|   Morvan Marie|   STAT|
|   Pelletier Bruno|   STAT|
|   Rouviere Laurent|   STAT|
+-----+-----+
```

1.6.6 orderBy

```
[14]: (df.filter(df["organization"] == "R2")
      .select(df["name"],df["position"])
      .orderBy("position")).show()
```

```
+-----+-----+
|          name|position|
+-----+-----+
|   Morvan Marie|   DOC|
|   Mom Alain|   MC|
|   Rouviere Laurent|   MC|
|   Le Guevel Ronan|   MC|
|Klutchnikoff Nicolas|   MC|
|Giacofci Joyce Ma...|   MC|
|Cornillon Pierre-...|   MC|
|Bennani-Dosse Moh...|   MC|
|   Pelletier Bruno|   PR|
|   Fromont Magalie|   PR|
|   Benasseni Jacques|   PR|
+-----+-----+
```

1.6.7 groupBy

```
[15]: df.groupby(df["hdr"])
```

```
[15]: <pyspark.sql.group.GroupedData at 0x7f301c6279a0>
```

```
[16]: df.groupby(df["hdr"]).count().show()
```

```
+-----+-----+
|  hdr|count|
+-----+-----+
| true|   103|
|false|   141|
+-----+-----+
```

WARNING: Don't confuse GroupedData.count() with DataFrame.count(). GroupedData.count() is not an action. DataFrame.count() is an action.

```
[17]: df.filter(df["hdr"]).count()
```

```
[17]: 103
```

```
[18]: df.filter(df['hdr']).select("name").show()
```

```
+-----+
|          name|
+-----+
|    Ammari Zied|
|  Bailleul Ismaël|
|    Baker Mark|
|  Beauchard Karine|
|    Bekka Bachir|
|  Belmiloudi Aziz|
|  Benasseni Jacques|
|  Berthelot Pierre|
|    Bourqui David|
|Breton Jean-Chris...|
|    Briane Marc|
|    Cadre Benoît|
|    Caloz Gabriel|
|    Cantat Serge|
|    Caruso Xavier|
|  Castella Francois|
|    Causeur David|
|  Cerveau Dominique|
|  Chartier Philippe|
|  Chauvet Guillaume|
```

```
+-----+
only showing top 20 rows
```

```
[19]: df.groupby(df["organization"]).count().show()
```

```
+-----+-----+
|organization|count|
+-----+-----+
|      ENS   |    3|
|      CNRS  |   19|
|      INSA  |   19|
|       R2   |   11|
|     INRIA  |    9|
|      AGRO  |    5|
|      EXT   |    2|
|       R1   |  176|
+-----+-----+
```

1.6.8 Exercises

- How many teachers from INSA (PR+MC) ?
- How many MC in STATS team ?
- How many MC+CR with HDR ?
- What is the ratio of student supervision (DOC / HDR) ?
- List number of people for every organization ?
- List number of HDR people for every team ?
- Which team contains most HDR ?
- List number of DOC students for every organization ?
- Which team contains most DOC ?
- List people from CNRS that are neither CR nor DR ?

```
[20]: df.select("organization").filter(df["organization"]=="INSA").count()
```

```
[20]: 19
```

```
[21]: (df.select(["position", "team1", "team2"])
      .filter((df["team1"]=="STAT") | (df["team2"]=="STAT"))
      .filter(df["position"] == "MC").count())
```

```
[21]: 15
```

```
[22]: (df.select(["position", "hdr"])
      .filter((df["position"]=="MC") | (df["position"]=="CR"))
      .filter(df["hdr"]).count())
```

[22]: 28

```
[23]: (df.select("position").filter(df["position"]=="DOC").count() /  
      df.select(df["hdr"]).filter(df["hdr"]).count())
```

[23]: 0.6019417475728155

```
[24]: (df.select(["hdr", "team1", "team2"]))  
      .filter("hdr")  
      .rdd.flatMap(lambda row: (row.team1, row.team2))  
      .filter(lambda v : v != 'NA')  
      .map(lambda row : (row,1))  
      .reduceByKey(lambda a, b:a+b)  
      .sortBy(lambda v: -v[1])  
      .collect()  
)
```

```
[24]: [('ANANUM', 21),  
      ('THEO-ERG', 14),  
      ('STAT', 14),  
      ('EDP', 11),  
      ('G&S', 9),  
      ('GAN', 9),  
      ('GA', 8),  
      ('GAE', 8),  
      ('PROC-STOC', 7),  
      ('MECA', 6),  
      ('IREM', 2),  
      ('ADM', 1)]
```

```
[25]: (df.select(["position", "team1", "team2"]))  
      .filter(df.position=="DOC")  
      .rdd.flatMap(lambda row: [row.team1, row.team2])  
      .filter(lambda v : v != 'NA')  
      .map(lambda row : (row,1))  
      .reduceByKey(lambda a, b:a+b)  
      .sortBy(lambda v: -v[1])  
      .collect()  
)
```

```
[25]: [('ANANUM', 14),  
      ('STAT', 9),  
      ('THEO-ERG', 8),  
      ('GAN', 8),  
      ('PROC-STOC', 8),  
      ('EDP', 7),  
      ('MECA', 4),
```

```
('GAE', 4),
('GA', 4),
('G&S', 2)]
```

```
[26]: import pyspark.sql.functions as f

df1 = (df.select(["position", "team1", "hdr"])
       .filter(df.hdr)
       .groupBy("team1")
       .agg(f.count("position").alias("count1")))
)
```

```
[27]: df2 = (df.select(["position", "team2", "hdr"])
          .filter(df.hdr)
          .filter(df.team2 != "NA")
          .groupBy("team2")
          .agg(f.count("team2").alias("count2")))
)
```

```
[28]: df3 = (df1.join(df2, df1.team1 == df2.team2, how="left")
            .na.fill(0)
            .drop("team2"))
```

```
[29]: df3.withColumn("total", df3.count1+df3.count2).orderBy("total",
↪ascending=False).show()
```

```
+-----+-----+-----+-----+
|   team1|count1|count2|total|
+-----+-----+-----+-----+
|  ANANUM|    21|     0|   21|
| THEO-ERG|    11|     3|   14|
|    STAT|    14|     0|   14|
|    EDP|    10|     1|   11|
|    GAN|     9|     0|    9|
|    G&S|     8|     1|    9|
|    GA|     7|     1|    8|
|    GAE|     8|     0|    8|
|PROC-STOC|     6|     1|    7|
|    MECA|     6|     0|    6|
|    IREM|     2|     0|    2|
|    ADM|     1|     0|    1|
+-----+-----+-----+-----+
```

```
[30]: (df.filter((df.position=="DOC") & (df.team1 == "ANANUM"))
      .select("name")
      .show())
```

```
)
```

```
+-----+
|          name|
+-----+
|   Belgacem Maher|
|   Bernier Joachim|
|   Calvez Adrien|
|   Corre Samuel|
|   Dao Manh Khang|
|   Doli Valentin|
|   Fontaine Marine|
|   Horsin Romain|
| Joannopoulos Emilie|
|   Le Balc'h Kevin|
|   Moitier Zoïs|
| Nguyen Thi-Hoai-T...|
|   Rosello Angelo|
|   Tusseau Maxime|
+-----+
```

```
[31]: (df.select("organization")
       .groupby("organization").count().show())
```

```
+-----+-----+
|organization|count|
+-----+-----+
|      ENS|    3|
|     CNRS|   19|
|     INSA|   19|
|       R2|   11|
|    INRIA|    9|
|     AGRO|    5|
|      EXT|    2|
|       R1|  176|
+-----+-----+
```

```
[32]: (df.select(["name", "organization", "position"])
       .filter((df.position == "DR") | (df.position == "CR"))
       .show())
```

```
+-----+-----+-----+
|          name|organization|position|
+-----+-----+-----+
|   Bavard Juliette|      CNRS|    CR|
| Bonthonneau Yannick|      CNRS|    CR|
```

	Cantat Serge	CNRS	DR
	Caruso Xavier	CNRS	CR
	Cérou Frédéric	INRIA	CR
	Chartier Philippe	INRIA	DR
	Coulon Rémi	CNRS	CR
	Crouseilles Nicolas	INRIA	CR
	Dauge Monique	CNRS	DR
	Duchene Vincent	CNRS	CR
	Erhel Jocelyne	INRIA	DR
	Faou Erwan	INRIA	DR
	Gros Michel	CNRS	CR
	Héas Patrick	CNRS	CR
	Herzet Cedric	CNRS	CR
	Kleptsyn Victor	CNRS	CR
	Le Gland François	INRIA	DR
	Lemou Mohammed	CNRS	DR
	Loray Frank	CNRS	DR
	Memin Etienne	INRIA	DR

+-----+-----+-----+

only showing top 20 rows

```
[33]: (df.select(["name","organization","position"])
      .filter(df.organization == "CNRS")
      .filter((df.position != "DR") & (df.position != "CR")))
      .groupBy("position").count().show())
```

+-----+-----+
position count
+-----+-----+
TC 2
IR 2
AI 1
IE 1
+-----+-----+

```
[34]: sc.stop()
```