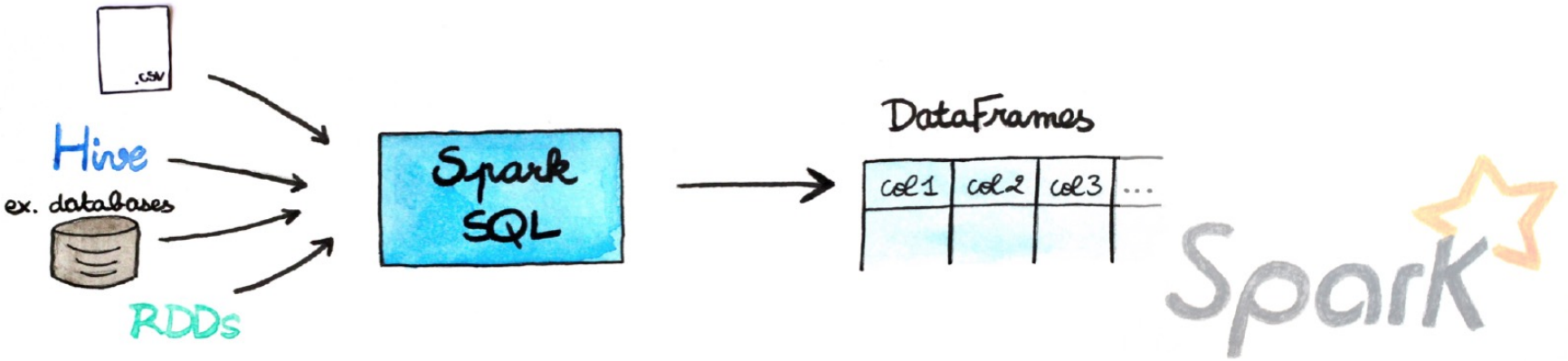
DATAFRAMES

Spark SQL is a component on top of **Spark Core** that facilitates processing of structured and semi-structured data and the integration of several data formats as source (Hive, Parquet, JSON).

https://spark.apache.org/docs/latest/sql-programming-guide.html

DataFrames

- DataFrame is an immutable distributed collection of data.
- Unlike an RDD, **data is organized into named columns**, like a table in a relational database or a dataframe in R/Python
- Distributed collection of data grouped into named columns:
  - DataFrames = RDD + Schema
- Designed to make large data sets processing even easier.
- Allows developers to impose a structure onto a distributed collection of data, allowing higher-level abstraction;

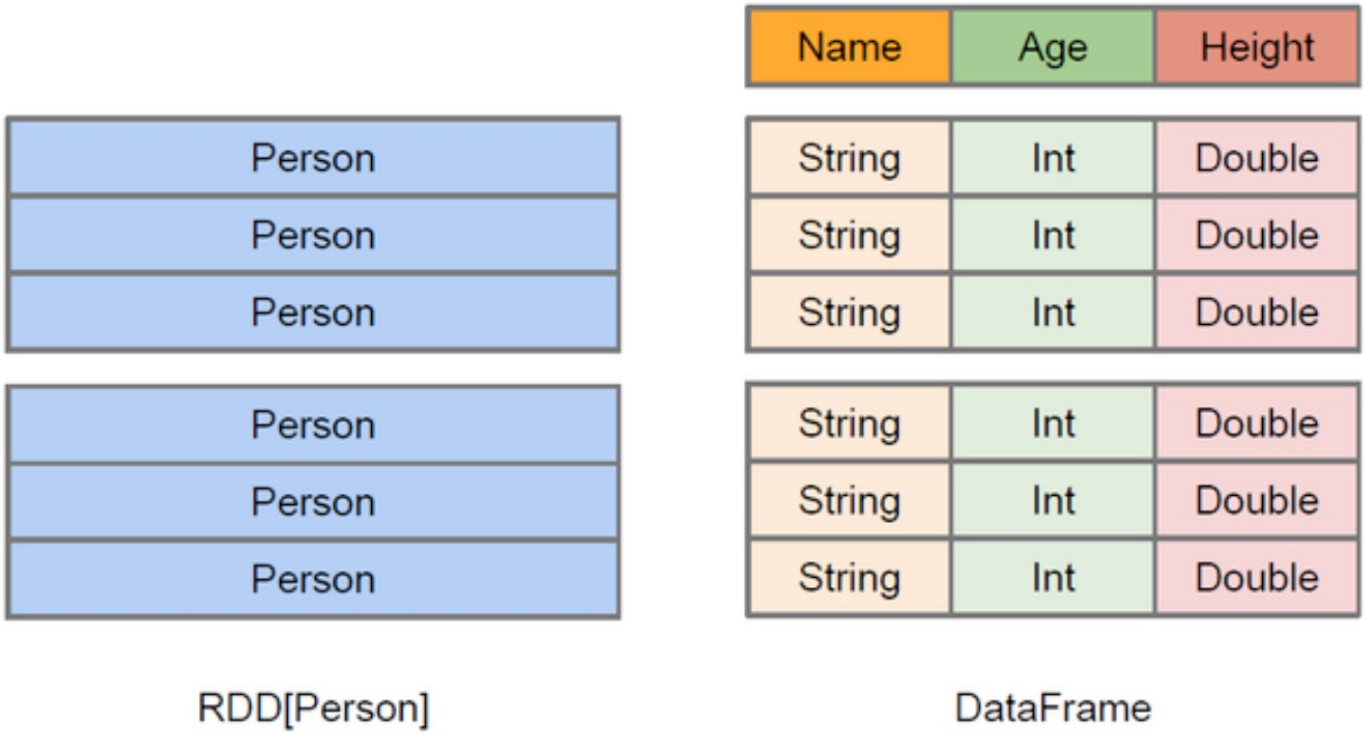


The structured spectrum

Structured	Semi-structured	Unstructured
• Relational Databases • Parquet • Formatted Messages	• HTML • XML • JSON	• Plain text • Generic media

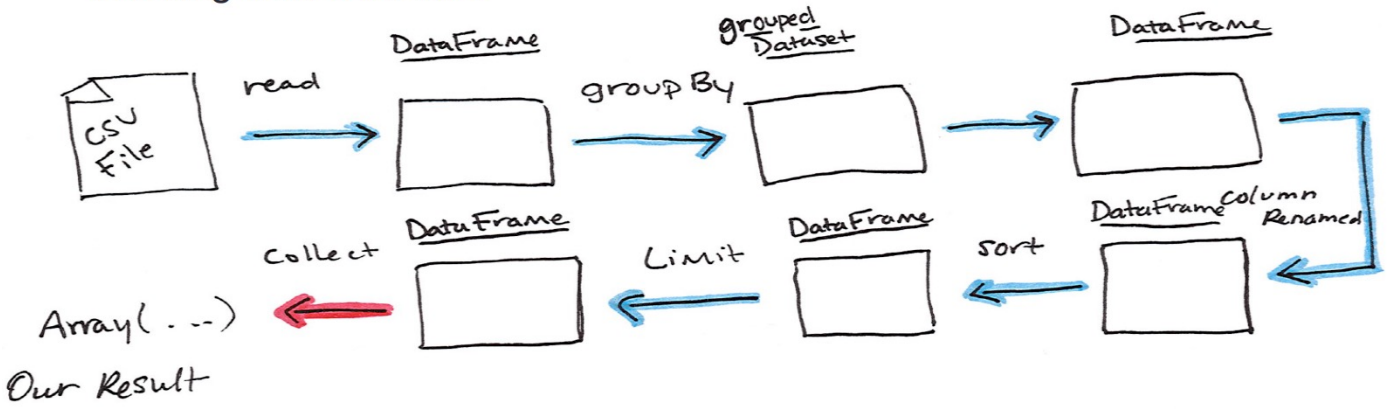
RDD vs DataFrames

DataFrames are composed of Row objects,
along with a schema that describes the data types of each column in the row.

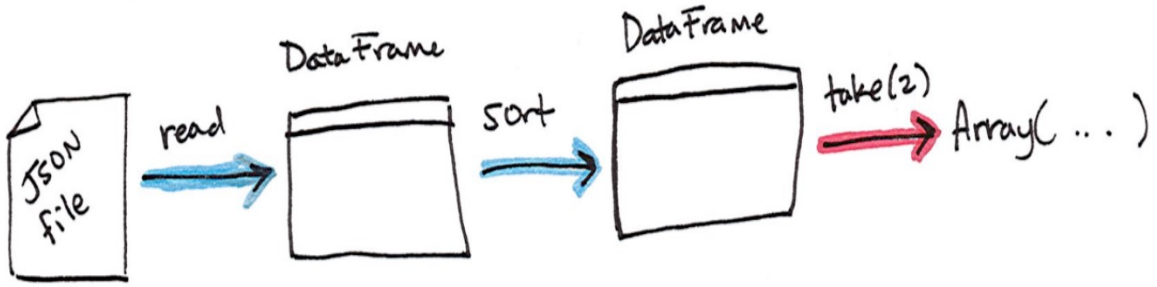


DataFrames and CVS/JSON Files

Working with CSV files



Working with Json files



https://spark.apache.org/docs/latest/sql-data-sources.html

DataFrame – read/write formats

- sqlContext.read.[format]

```
>> sqlContext.read.parquet(path)
>> sqlContext.read.json(path, [schema])
>> sqlContext.read.jdbc(url, table)
>> sqlContext.read.load(path, [format])
```

- sqlContext.write.[format]

```
>> sqlContext.write.parquet(path)
>> sqlContext.write.json(path, [mode])
>> sqlContext.write.jdbc(url, table, [mode])
>> sqlContext.write.save(path, [format], [mode])
```

DataFrames and RDDs

- Create from RDD of tuples

```
>> rdd = sc.parallelize([(”a”, 1), (”b”, 2), (”c”, 3)])
>> df = sqlContext.createDataFrame(rdd,[”name”, ”id”])
>> df.show()
```

```
+----+--+
|name|id|
+----+--+
|   a| 1|
|   b| 2|
|   c| 3|
+----+--+
```

Examples of SparkSQL (1)

- Read a JSON file

```
players = sqlContext.read.json('players.json’)
players.printSchema()
players.select("FullName").show(4)
+-----------------------+
| FullName              |
+-----------------------+
| Ãngel Bossio          |
| Juan Botasso          |
| Roberto Cherro        |
| Alberto Chividini.    |
+-----------------------+
```

Examples of SparkSQL (1)

Create a view of our DataFrame. The lifetime of this temporary table is tied to the SparkSession that was used to create this DataFrame.

```
players.registerTempTable("players")

sqlc.sql("select distinct Team from players").show(5)
```

```
+-----------+
| Team      |
+-----------+
| Scotland  |
| Paraguay  |
| Poland    |
| Spain     |
| Italy     |
+-----------+
```

Examples of SparkSQL (2)

```
(py39) rf208@MCL7WK6WC9M0 walkthrough_ex
{"age":'',"name":"Michael"},
{"age":30, "name":"Andy"},
{"age":"19", "name":"Justin"}

Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.2
      /_/

Using Python version 3.9.7 (default, Sep
Spark context Web UI available at http://
Spark context available as 'sc' (master =
SparkSession available as 'spark'.
>>> df = spark.read.json('people.json')
>>> df.show()
+---+-------+
|age|   name|
+---+-------+
|   |Michael|
| 30|   Andy|
| 19| Justin|
+---+-------+
```

```
# Select everybody, but increment the age by 1
df.select(df['name'], df['age'] + 1).show()
# +-------+---------+
# |   name|(age + 1)|
# +-------+---------+
# |Michael|     null|
# |   Andy|       31|
# | Justin|       20|
# +-------+---------+

# Select people older than 21
df.filter(df['age'] > 21).show()
# +---+----+
# |age|name|
# +---+----+
# | 30|Andy|
# +---+----+

# Count people by age
df.groupBy("age").count().show()
# +---+-----+
# |age|count|
# +---+-----+
# | 30|    1|
# | 19|    1|
# |   |    1|
# +---+-----+
```

Pandas DataFrame

- When in PySpark, there is also an easy option to convert Spark DataFrame to Pandas dataframe.

```
# Convert Spark DataFrame to Pandas
pandas_df = spark_df.toPandas()

# Create a Spark DataFrame from Pandas
spark_df = context.createDataFrame(pandas_df)
```

- One powerful and easy way to visualize data is

```
dataframe.toPandas().plot()
```