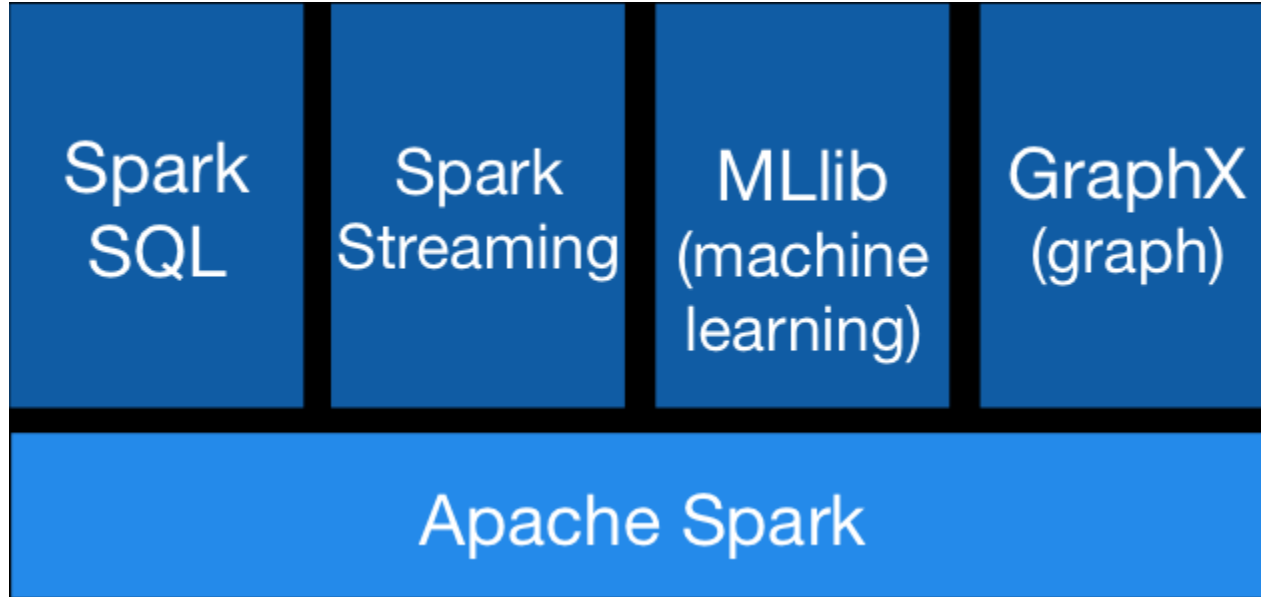


Spark SQL

Spark SQL es el modulo de Apache Spark para trabajar con datos estructurados



Motivos para la creación de SparkSQL

- Permite acceder a datos con SQL directamente
- Permite ejecutar programas Spark más rápido
- Permite a los desarrolladores
 - Escribir menos código
 - Que los programas lean menos datos
 - Usar el optimizador catalyst

DataFrames

- Colección de datos distribuidos organizados en columnas con nombre
- Equivalente a una tabla en modelo relacional
- A diferencia de los RDDs tienen esquema asociado
 - DataFrame = RDD + esquema (descripción de la estructura de los datos)
 - Permite hacer optimizaciones
- Cargan datos de fuentes diversas
 - Ficheros Parquet
 - Ficheros JSON
 - BD externas usando JDBC
 - RDDs

DataFrames (II)

- API disponible en Scala, Python y R
- DSL (domain-specific language)
 - Lenguaje para ejecutar operaciones relacionales sobre DataFrames
- SQLContext
 - Punto de entrada a SparkSQL

Para empezar: crear sesión Spark

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("Python Spark SQL basic example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

Ficheros JSON

- Los datos se representan como pares clave-valor

“name” : “Michael”

- Cuatro tipos de datos

- String “name” : “Michael”

- Number “age” : 27

- Boolean “vivo” : true

- null “profesor” : null

- Los datos se delimitan por comas

- Se usan {} para representar objetos

{“name” : “Michael”, “age” : 27}

- Se usan [] para representar arrays

[{“name” : “Michael”, “age” : 27}, {“name” : “Mary”, “age” : 34}]

Crear un DataFrame

gente.json

```
{"name": "Michael"}  
{"name": "Andy", "age": 30}  
{"name": "Justin", "age": 19}
```

```
df = spark.read.json("gente.json")
```

```
df.show()
```

```
+-----+-----+  
|  age |   name |  
+-----+-----+  
| null | Michael |  
|   30 |    Andy |  
|   19 |   Justin |  
+-----+-----+
```


Operaciones sobre DataFrames

- Lenguaje específico para manejo de datos estructurados

<http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame>

- También disponible una biblioteca de funciones para manipulación de cadenas, fechas, operaciones matemáticas,

...

<http://spark.apache.org/docs/latest/api/python/pyspark.sql.html#module-pyspark.sql.functions>

Operaciones sobre DataFrames: EJEMPLOS

- Imprimir esquema del DF en formato árbol

```
df.printSchema()
```

```
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
```

Operaciones sobre DataFrames: EJEMPLOS

- Seleccionar sólo la columna *name*

```
df.select("name").show()
```

```
+-----+  
|  name |  
+-----+  
|Michael|  
|  Andy |  
| Justin|  
+-----+
```

Operaciones sobre DataFrames: EJEMPLOS

- Seleccionar todos, pero incrementando en 1 la edad

```
df.select(df['name'], df['age'] + 1).show()
```

```
+-----+-----+
|  name | (age + 1) |
+-----+-----+
|Michael|      null |
|  Andy |       31  |
| Justin|       20  |
+-----+-----+
```

Operaciones sobre DataFrames: EJEMPLOS

- Seleccionar los mayores de 21

```
df.filter(df['age'] > 21).show()
```

```
+---+-----+  
|age|name|  
+---+-----+  
| 30|Andy|  
+---+-----+
```

Operaciones sobre DataFrames: EJEMPLOS


- Agrupar por edad y contar

```
df.groupBy("age").count().show()
```

+	-	-	-	+	-	-	-	-	+
		age		count					
+	-	-	-	+	-	-	-	-	+
		19			1				
		null			1				
		30			1				
+	-	-	-	+	-	-	-	-	+

Ejecutar consultas SQL y devolver resultado como DataFrame

Previamente hay
que crear una
vista sobre el
dataframe



```
df.createOrReplaceTempView("people")
```

```
sqlDF = spark.sql("SELECT * FROM people")  
sqlDF.show()
```

```
+-----+-----+  
|  age|   name|  
+-----+-----+  
| null|Michael|  
|   30|   Andy|  
|   19| Justin|  
+-----+-----+
```



Usamos la
función sql

Convertir RDD en DataFrames

- Dos formas

1. Inferir el esquema a partir del RDD

- Código más conciso
- Funciona bien cuando ya conoces el esquema en el momento de escribir la aplicación

2. A través de una interfaz que permite construir el esquema y aplicarlo al RDD

Infiriendo el esquema

1. Crear un RDD de tuplas a partir del RDD original
Cargar un fichero de texto y convertir cada fila a Row.

```
from pyspark.sql import Row
```

```
sc = spark.sparkContext
```

```
lines = sc.textFile("gente.txt")
```

```
parts = lines.map(lambda l: l.split(","))  
people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))
```

Infiriendo el esquema (II)

2. Crear el dataframe

```
dfPeople = spark.createDataFrame(people)
```

3. Registrar el dataframe como tabla

```
dfPeople.createOrReplaceTempView("people")
```

Construyendo el esquema

1. Crear un RDD de tuplas a partir del RDD original

```
# Importar tipos de datos
from pyspark.sql.types import *

sc = spark.sparkContext

# Cargar un fichero de texto y convertir cada fila en una tupla
lines = sc.textFile("gente.txt")
parts = lines.map(lambda l: l.split(","))
# Cada línea se convierte en una tupla
people = parts.map(lambda p: (p[0], int(p[1])))
```

Construyendo el esquema (II)

2. Crear el esquema mediante StructType

```
fields = [StructField("name", StringType(), True), StructField("age", IntegerType(), True)]  
schema = StructType(fields)
```

nullable



3. Aplicar el esquema al RDD mediante el método createDataFrame

```
# Crear el dataframe aplicando el esquema al RDD  
dfPeople = spark.createDataFrame(people, schema)  
  
# Registrar el DataFrame como tabla  
dfPeople.createOrReplaceTempView("people")  
  
# Ejecutar SQL  
results = spark.sql("SELECT name,age FROM people")  
  
results.show()
```

```
+-----+  
|  name |  
+-----+  
|Michael|  
|  Andy |  
|  Justin|  
+-----+
```

Crear dataframe a partir de fichero con esquema

1. Crear el esquema mediante StructType

```
fields = [StructField("name", StringType(), True), StructField("age", IntegerType(), True)]  
schema = StructType(fields)
```

2. Aplicar el esquema al leer el fichero

```
dfPeople = spark.read.format('csv').options(header=True,  
inferSchema=False, delimiter=";").schema(schema).load("gente.csv")
```

Otra opción

```
dfPeople = spark.read.csv(("gente.csv", schema=schema, nullValue="null"))
```