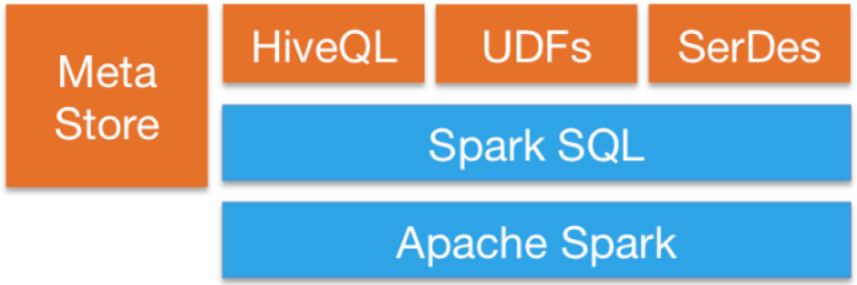
Spark SQL

Spark SQL



- Seamlessly mix SQL queries with Spark programs.
- Connect to any data source the same way.
- Run SQL or HiveQL queries on existing warehouses.
- Connect through JDBC or ODBC.
- https://spark.apache.org/sql/
- Notebook (SparkSQL-02-GettingStarted)

Conclusion

- Semi-Structured Data can be read directly into a “structured” object called a Spark DataFrame
- Unstructured data must be passed through an RDD and parsed piece by piece.

Objectives

- Ingest and run SQL queries on semi-structured data(json) and on unstructured data(text file)
- ***select * from people;*** ***// this is the goal***

```
!cat data/people.json
```

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

```
!cat data/people.txt
```

```
Michael, 29
Andy, 30
Justin, 19
```

Schema

- In the case of json, we can infer the scheme
- With a text file we must explicitly declare it.

```
!cat data/people.json
```

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

```
!cat data/people.txt
```

```
Michael, 29
Andy, 30
Justin, 19
```

Start the Spark Context and Spark Session

```
import pyspark
sc = pyspark.SparkContext('local[*]')
```

```
from pyspark.sql import SparkSession

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

Case of Json

Semi-structured Data

1. Start a sqlContext
2. Read Json into a DataFrame
3. Register the DataFrame as an SQL Table
4. Perform SQL Query
5. Display results

Json

```
# read json file into a DataFrame
from pyspark import SQLContext

#Start an sqlContext
sqlContext = SQLContext(sc)

#Read Json
people = sqlContext.read.json("data/people.json")

#Register the people df as a table
people.registerTempTable("people")

#Perform SQL Query
all_people = spark.sql("SELECT * FROM people ")
all_people.show()
```

Unstructured Data: text file to SQL

1. Import a ***ROW*** type (lots of imports with Spark SQL)
2. Read text into a ***lines*** RDD
3. Parse the ***lines*** RDD into a ***parts*** RDD
4. Move each line of the ***parts*** RDD to a ***ROW*** where the first element ***p[0]*** is a ***name*** and ***p[1]*** is an ***age. This defines the schema and converts the RDD to a DataFrame***
5. Register the table
6. Perform SQL Queries

Text to SQL

```python
from pyspark.sql import Row

# Load a text file and convert each line to a Row.
lines = sc.textFile("data/people.txt")
parts = lines.map(lambda l: l.split(","))
people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))

# Infer the schema, and register the DataFrame as a table.
schemaPeople = spark.createDataFrame(people)
schemaPeople.createOrReplaceTempView("people")

# SQL can be run over DataFrames that have been registered as a table.
teenagers = spark.sql("SELECT name FROM people WHERE age >= 13 AND age <= 19")
teenagers.show()
```