UE Large Scale Processing @ ESIGELEC
2019/2020

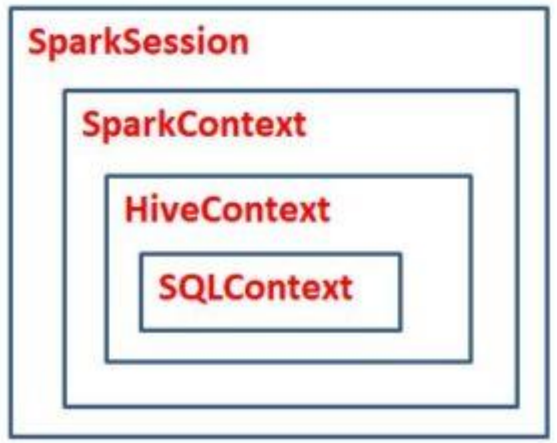
14 - Spark SQL - Spark Application Context

Abdel Dadouche
DJZ Consulting

adadouche@hotmail.com
@adadouche

SparkContext vs SQLContext vs HiveContext vs SparkSession

- Before Spark 2.x , SparkContext is the only entry point of any Spark Application
- Used by the Driver program to connect & communicate with the cluster
- Can be used to provide Job config parameters
- Provided by default in Spark shells (sc)
- Encapsulate HiveContext which encapsulates SQLContext
- Read/write operation only deals with RDD!



SparkContext vs **SQLContext** vs HiveContext vs SparkSession

- SQLContext enables use of SQL in a Spark apps and combine DS/DF “APIs” with SQL
- Provides a basic set of SQL functionality

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import SQLContext

from os.path import expanduser
home = expanduser("~") + "/esigelec-ue-lsp-hdp/spark-3.0.0"
path = "file://" + home + "/examples/src/main/resources"

sparkConf = SparkConf() \
   .setAppName("SparkSQL App") \
   .setMaster("local")

sparkContext = SparkContext(conf=sparkConf)

sqlContext = SQLContext(sparkContext)

# read with the csv method
df = sparkSession.read.option("sep", ";").option("inferSchema", "true") \
  .option("header", "true").csv(path + "/people.csv")

sqlContext.registerDataFrameAsTable(df, "df")

names = sqlContext.sql("SELECT name FROM df where age > 31")

names.show()
```

SparkContext vs SQLContext vs **HiveContext** vs SparkSession

- HiveContext provide a superset of the SQLContext features
- More complete support of HiveQL (parser)
- Can read data from Hive tables
- Define and use Hive UDFs
- No Hive setup needed!
- In Spark 3.0, the HiveContext has been removed and fully incorporated in the SparkSession API with the enableHiveSupport() method

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import HiveContext

from os.path import expanduser
home = expanduser("~") + "/esigelec-ue-lsp-hdp/spark-3.0.0"
path = "file://" + home + "/examples/src/main/resources"

sparkConf = SparkConf() \
   .setAppName("SparkSQL App") \
   .setMaster("local")

sparkContext = SparkContext(conf=sparkConf)

hiveContext = HiveContext(sparkContext)

# read with the csv method
df = sparkSession.read.option("sep", ";").option("inferSchema", "true") \
  .option("header", "true").csv(path + "/people.csv")

hiveContext.registerDataFrameAsTable(df, "df")

names = hiveContext.sql("SELECT name FROM df where age > 31")

names.show()
```

SparkContext vs SQLContext vs HiveContext vs **SparkSession**

- Introduced in Spark 2.0
- Goal: Streamline/wraps access to all contexts
- Provided by default in Spark shells (as spark)
- Provide Hive apabilities (enableHiveSupport)
- Create temp views to query DF/DS with SQL (createOrReplaceTempView)

Note: In Spark 3.0, SparkSession fully replaces HiveContext (removed)

```
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession

from os.path import expanduser
home = expanduser("~") + "/esigelec-ue-lsp-hdp/spark-3.0.0“
path = "file://" + home + "/examples/src/main/resources“

sparkSession = SparkSession.builder
    .master("spark://localhost:7077") \
    .appName("SparkSQL App") \
    .getOrCreate()

# read with the csv method
df = sparkSession.read.option("sep", ";").option("inferSchema", "true") \
  .option("header", "true").csv(path + "/people.csv")

df.createOrReplaceTempView("df")

names = sparkSession.sql("SELECT name FROM df where age > 31")

names.show()
```