Trabajando con Spark SQL y dataframes

Spark SQL es el módulo que permite trabajar con datos estructurados de manera uniforme independientemente del origen de datos.

Dataframes

Un data frame es una colección distribuida de datos organizada por columnas,las cuales tienen asociado un nombre.Un dataframe es equivalente al concepto de tabla de una base de datos relacional,aunque el dataframe está optimizado para el rendimiento dentro de un cluster con spark.

El punto de entrada de Spark SQL es la clase SQLContext del módulo pyspark.sql.

```
from pyspark.sql import SQLContext

sqlContext = SQLContext(sc)
```

Para crear un dataframe se puede usar la función **createDataFrame(rdd)**

Consultas SQL

Spark permite la consulta de datos mediante un subconjunto del lenguaje SQL a través del método sql(consulta) de la clase SQLContext

```
sqlContext = SQLContext(sc)
data=sqlContext.read.json("data.json")
data.registerTempTable("Nombre_tabla")
consulta=sqlContext.sql(""" SELECT * from Nombre_tabla""")
```

Creación de DataFrames

Con un objeto SQLContext, las aplicaciones pueden crear DataFrames de un RDD existente, de una tabla o de fuentes de datos.

Como ejemplo, lo siguiente crea un DataFrame basado en el contenido de un archivo JSON:

Al realizar sqlContext.read.json lo que se obtiene es un dataframe que contiene un esquema de la estrucura de tablas con la información que lee a partir del fichero json

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)

df = sqlContext.read.json("examples/src/main/resources/people.json")

# Displays the content of the DataFrame to stdout
df.show()
```

Operaciones sobre un DataFrame

Los dataFrames proporcionan un conjunto de funciones para la manipulación de datos estructurados. Aquí incluimos algunos ejemplos básicos de procesamiento de datos estructurados usando DataFrames:

En Python es posible acceder a las columnas de un DataFrame ya sea por atributo (df.age) o por indexación (df ['age']). Mientras que el primero es conveniente para la exploración interactiva de datos, se recomienda a los usuarios utilizar la segunda forma.

```
# Print the schema in a tree format
df.printSchema()
## root
## |-- age: long (nullable = true)
## |-- name: string (nullable = true)

# Select only the "name" column
df.select("name").show()
## name
## Michael
## Andy
## 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
```

Ejecución de consultas SQL de forma programática

La función sql en un SQLContext permite a las aplicaciones ejecutar consultas SQL de forma programática y devuelve el resultado como un DataFrame.

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
df = sqlContext.sql("SELECT * FROM table")
```

Interoperando con RDDs

Spark SQL soporta dos métodos diferentes para convertir RDDs existentes en DataFrames. El primer método utiliza la reflexión para inferir el esquema de un RDD que contiene tipos específicos de objetos. Este enfoque está basado en la reflexión y funciona bien cuando ya conoce el esquema mientras escribe su aplicación Spark.

Spark SQL puede convertir un RDD de objetos Row en un DataFrame, deduciendo los tipos de datos. Las filas se construyen al pasar una lista de pares clave / valor. Las claves de esta lista definen los nombres de columna de la tabla y los tipos se infieren mirando la primera fila.

```
# sc is an existing SparkContext.
from pyspark.sql import SQLContext, Row
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a Row.
lines = sc.textFile("examples/src/main/resources/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 = sqlContext.createDataFrame(people)
schemaPeople.registerTempTable("people")

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

# The results of SQL queries are RDDs and support all the normal RDD operations.
teenNames = teenagers.map(lambda p: "Name: " + p.name)
for teenName in teenNames.collect():
  print(teenName)
```

El segundo método para crear DataFrames es a través de una interfaz programática que le permite construir un esquema y luego aplicarlo a un RDD existente. Si bien este método es más detallado, le permite crear DataFrames cuando las columnas y sus tipos no se conocen hasta el tiempo de ejecución.

Los pasos a seguir:

- **Crear un RDD de tuplas o listas del RDD original;**
- **Cree el esquema representado por un StructType que coincida con la estructura de tuplas o listas en el RDD creado en el paso 1.**
- **Aplicar el esquema al RDD a través del método createDataFrame proporcionado por SQLContext.**

```
# Import SQLContext and data types
from pyspark.sql import SQLContext
from pyspark.sql.types import *

# sc is an existing SparkContext.
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a tuple.
lines = sc.textFile("examples/src/main/resources/people.txt")
parts = lines.map(lambda l: l.split(","))
people = parts.map(lambda p: (p[0], p[1].strip()))

# The schema is encoded in a string.
schemaString = "name age"

fields = [StructField(field_name, StringType(), True) for field_name in schemaString.split()]
schema = StructType(fields)

# Apply the schema to the RDD.
schemaPeople = sqlContext.createDataFrame(people, schema)

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

# SQL can be run over DataFrames that have been registered as a table.
```