You can see this official guide.

Overview

There are several ways to interact with Spark SQL including SQL, the DataFrames API and the Datasets API.

SQLContext

Create SQlContext:
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
In spark-shell mode, sqlContext has been created.

import sqlContext.implicits._ // this is used to implicitly convert an RDD to a DataFrame.

Creating DataFrames

With a SQLContext, applications can create DataFrames from an existing RDD, from a Hive table, or from data sources.
val df = sqlContext.read.json("examples/src/main/resources/people.json")

DataFrame Operations

df.show() // Displays the content of the DataFrame to stdout
df.printSchema() // Print the schema in a tree format
df.select("name").show() // Select only the "name" column
df.select(df("name"), df("age") + 1).show() // Select everybody, but increment the age by 1
df.filter(df("age") > 21).show() // Select people older than 21
df.groupBy("age").count().show() // Count people by age

Running SQL Queries Programmatically

The sql function on a SQLContext enables applications to run SQL queries programmatically and returns the result as a DataFrame.
val df = sqlContext.sql("SELECT * FROM table")

Creating Datasets

// Encoders for most common types are automatically provided by importing
sqlContext.implicits._
val ds = Seq(1, 2, 3).toDS()
ds.map(_ + 1).collect() // Returns: Array(2, 3, 4)

```
// Encoders are also created for case classes.
case class Person(name: String, age: Long)
val ds = Seq(Person("Andy", 32)).toDS()

// DataFrames can be converted to a Dataset by providing a class. Mapping will be done by name.
val path = "examples/src/main/resources/people.json"
val people = sqlContext.read.json(path).as[Person]  // return a Dataset
```

Interoperating with RDDs

Inferring the Schema Using Reflection

The Scala interface for Spark SQL supports automatically converting an RDD containing case classes to a DataFrame. This RDD can be implicitly converted to a DataFrame and then be registered as a table.

```
val people = sc.textFile("examples/src/main/resources/people.txt").map(_.split(",")).map(p => Person(p(0), p(1).trim.toInt)).toDF()
people.registerTempTable("people")
```

Programmatically Specifying the Schema

When case classes cannot be defined ahead of time (for example, the structure of records is encoded in a string, or a text dataset will be parsed and fields will be projected differently for different users), a DataFrame can be created programmatically with three steps.

1. Create an RDD of Rows from the original RDD;
2. Create the schema represented by a StructType matching the structure of Rows in the RDD created in Step 1.
3. Apply the schema to the RDD of Rows via createDataFrame method provided by SQLContext.

```
val people = sc.textFile("examples/src/main/resources/people.txt")
val schemaString = "name age"
val schema =
  StructType(
    schemaString.split(" ").map(fieldName => StructField(fieldName, StringType, true)))
val rowRDD = people.map(_.split(",")).map(p => Row(p(0), p(1).trim))
val peopleDataFrame = sqlContext.createDataFrame(rowRDD, schema)
peopleDataFrame.registerTempTable("people")
```

连接MySQL

如果遇到 java.sql.SQLException: No suitable driver found for jdbc:mysql 问题，增加以下代码Class.forName("com.mysql.jdbc.Driver")，参考链接：link.

运行上一句代码时会出现 spark java.lang.ClassNotFoundException: com.mysql.jdbc.Driver 错误，需要设置jar包的路径指向mysql-connector-java-5.1.38-bin.jar：
./spark-submit --jars <path to jar>
如果在spark-shell或者standalone模式中运行，同样用--jars参数指定 mysql-connector-java-5.1.38-bin.jar的路径。

如果不起作用的话，同时加入--driver-class-path <path to jar>设置。
如果再不行，再加上--conf spark.executor.extraClassPath=<path to jar>以及--conf spark.driver.extraClassPath=<path to jar>设置。

读MySQL

```
val sqlContext = new SQLContext(data.sparkContext)
val reader = sqlContext.read
val properties = new Properties()
properties.setProperty("user", username)
properties.setProperty("password", password)
val df = reader.jdbc(url, table, Array("id = 1"), properties)
```

DataFrame的几种写入模式

append
不改变表中已经存在记录，将新纪录写入。

overwrite
原来的记录全部删除，只有增加的记录。
举个例子：如果原来的表中有三个字段a, b, c，Overwrite的数据中只有两个字段a, b，那么原来c字段就没有了，慎用。

error
如果已经存在记录，抛出异常。

ignore
如果有已经存在的记录，DataFrame中的数据全部不会写入。

spark 2.0.0中新增了SparkSession：

```
import org.apache.spark.sql.SparkSession
case class Table(field1: type, field2: type)
val spark = SparkSession.builder.appName("sql").getOrCreate()
import spark.implicits._
val df = sc.textFile(path).map(_.split("\001"))
.map { x =>
```

```
  Table(x(0), x(1))
}.toDF()
```

Spark 2.0 通过 jdbc 连接数据库：
http://dblab.xmu.edu.cn/blog/1190/

by chenxiaoyu@mobvoi
E-mail: jschenxiaoyu@gmail.com
2016.1.14