Spark SQL实验小结（2022.4.10）

原创

shdkdh 已于 2022-04-10 21:22:46 修改 2077 收藏

文章标签： spark

于 2022-04-10 21:21:52 首次发布

本文链接：https://blog.csdn.net/weixin_45823693/article/details/124086170

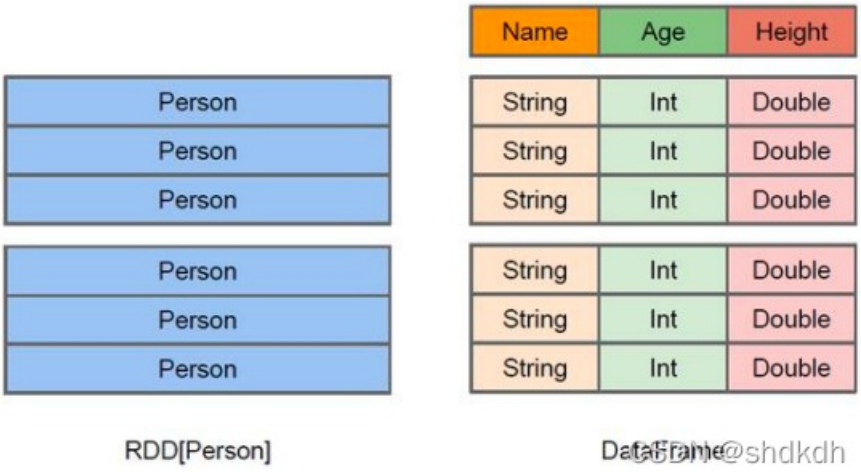
Spark SQL实验小结（2022.4.10）

说实话，这作业写的也是真够久的，给我累到了~但是，我不能让我辛辛苦苦学的东西过几天就忘了，所以呢，所以呢，学一下大佬的学习方法，写个博客记录一下（希望真的真的能从今天开始养成习惯，虽然很久以前也是这样说的，绝了，希望记住吧！）

文章目录

一、RDD与DataFrame

RDD 是整个 Spark 平台的存储、计算以及任务调度的逻辑基础，更具有通用性，适用于各类数据源，而 DataFrame 是只针对结构化数据源的高层数据抽象，其中在 DataFrame 对象的创建过程中必须指定数据集的结构信息(Schema)，所以 DataFrame 生来便是具有专用性的数据抽象，只能读取具有鲜明结构的数据集。
如下图所示：



上图直观地体现了 DataFrame 和 RDD 的区别。
左侧的 RDD[Person]虽然以 Person 为类型参数，但 Spark 框架本身不了解 Person 类的内部结构。而右侧的 DataFrame 却提供了详细的结构信息，使得 Spark SQL 可以清楚地知道该数据集中包含哪些列，每列的名称和类型各是什么。
DataFrame 是为数据提供了 Schema 的视图。可以把它当做数据库中的一张表来对待DataFrame 也是懒执行的，但性能上比 RDD 要高，主要原因：优化的执行计划，即查询计划通过 Spark catalyst optimiser 进行优化。
（参考尚硅谷的大数据技术之SparkSql）

二、创建DataFrame对象

1.创建方式

1> 结构化数据文件创建DataFrame
2> 外部数据库创建DataFrame
3> RDD创建DataFrame方式
4> Hive中的表创建DataFrame（由于之前偷懒，没配置好hive环境，所以这个就不详细介绍了，因为自己没有亲自实践，所以写出的东西肯定bug百出，下次弄了的话再补齐吧~）

2.结构化数据文件创建DataFrame

本实验数据样式：

```
1::F::1::10::48067
2::M::56::16::70072
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
6::F::50::9::55117
7::M::35::1::06810
```

先上代码：

```scala
package SparkSQL
// 导包
import org.apache.spark.rdd.RDD
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql.{DataFrame, Dataset, Row, SQLContext, SparkSession}

object dataFrame_test {
  def main(args: Array[String]): Unit = {
    // 0.准备环境(常见操作)
    // SparkSession 可以删除，因为Spark有类型推断性质，后面同理
    val spark: SparkSession = SparkSession.builder().appName("dataFrame_test").master("local[*]").getOrCreate()
    val sc : SparkContext = spark.sparkContext
    // 读取外部文件，函数：spark.read，类型为DataFrame(后面的参数可根据自己的情况进行修改)
    val people1_DF : DataFrame = spark.read.format("csv").option("header", "false").option("inferSchema", "false
")
                    .option("delimiter","::").load("E:\\IntelliJ IDEA\\IdeaProjects\\dzh\\SparkSQL_data\\people.
dat")
    // printSchema 函数查看数据模式，打印出列的名称和类型
    people1_DF.printSchema()
    // show 函数，查看数据，可以指定查看的行数，默认显示前20行数据
    people1_DF.show()
    spark.close()
  }
}
```

这一部分没什么好详细阐述的，就是直接套代码就好了（不过spark sql 依赖肯定是要配好的）

3.外部数据库创建DataFrame（Mysql）

这里弄了我好久啊，主要原因是重装了一下Mysql，然后图形界面也没了，最后是在命令行界面弄的，不过感觉大家都是在命令行下操作，不知道是不是我的错觉。

1）mysql 下载好

2）下载 mysql 的依赖包，我下的是

> mysql-connector-java-8.0.11

版本，版本要根据自己的mysql版本来下载（我的mysql版本是8.0.11的，所以我下的是上面的版本）。

3）添加依赖（版本根据自己的版本进行修改）

```xml
<dependency>
 <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.11</version>
</dependency>
```

如果显示红色，不妨先试一下rebuild（这里我一开始是红色（报错），然后在网上搜了好久，还一直是红色的，然后在那折腾了好久，最后rebuild一下，就好了，不过也可能是前面被我改对了，如果不成的话，再去网上搜一下吧）

4）代码1（印象中，上面弄好了以后，就可以直接读取了，此处默认mysql中以及建立好了相应的数据库和表）

```scala
package SparkSQL
// 导包
import java.util.Properties
import org.apache.spark.rdd.RDD
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql.{DataFrame, Dataset, Row, SQLContext, SparkSession}

object dataFrame_test {
  def main(args: Array[String]): Unit = {
    // 0.准备环境
    val spark: SparkSession = SparkSession.builder().appName("dataFrame_test").master("local[*]").getOrCreate()
    val sc : SparkContext = spark.sparkContext

    // 1.连接 spark SQL 和 mysql
    /**
     * jdbc(url: String, table: String, properties: Properties)
     * 解析： 作用是连接mysql，读取mysql里的表数据，返回DataFrame
     * url： 连接mysql的路径：     8.0版本：jdbc:mysql://ip:port/dbname?serverTimezone=UTC
     * 低版本：jdbc:mysql://ip:port/dbname
     * table： 数据库里的表名
     * properties: 用于指定连接mysql的用户名，密码等一个Properties对象
     */
    // 此处， test1 是mysql中数据库的名称，people 是mysql中数据表的名称，user 和 password 是需要连接的mysql的用户和密码
    val url = "jdbc:mysql://localhost:3306/test1"
    val table = "people"
    val prop = new Properties()
    prop.setProperty("user", "root")
    prop.put("password", "123456")
    val people2_DF: DataFrame = spark.read.jdbc(url, table, prop)
    people2_DF.printSchema()
    people2_DF.show(5)
    people2_DF.head(3)
    people2_DF.collect()
    spark.close()
  }
}
```

5）代码2（第二种方式）

```scala
package SparkSQL
// 导包
import java.util.Properties
import org.apache.spark.rdd.RDD
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql.{DataFrame, Dataset, Row, SQLContext, SparkSession}

object dataFrame_test {
  def main(args: Array[String]): Unit = {
    // 0.准备环境
    val spark: SparkSession = SparkSession.builder().appName("dataFrame_test").master("local[*]").getOrCreate()
    val sc : SparkContext = spark.sparkContext

    val people2_DF: DataFrame = read(spark)
    people2_DF.printSchema()
    people2_DF.show(5)
    people2_DF.head(3)
    people2_DF.collect()
    spark.close()
  }
  // 也可以使用下面的方式读取数据
  def read(spark: SparkSession): DataFrame = {
    val jdbcDF = spark.read
      .format("jdbc")
      .option("url", "jdbc:mysql://localhost:3306/test1")
      .option("dbtable", "people")
      .option("user", "root")
      .option("password", "123456")
      .option("driver", "oracle.jdbc.driver.OracleDriver")
      .load()
    jdbcDF
  }
}
```

4.RDD创建DataFrame方式

1）关键点：先定义一个 case class，然后借助这个将RDD转换成DataFrame （定义方式有多种，这里只展示一种）
2）代码

```scala
package SparkSQL
// 导包
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.rdd.RDD
import org.apache.spark.sql.types.{IntegerType, StringType, StructField, StructType}
import org.apache.spark.sql.{DataFrame, Dataset, Row, SparkSession}
// 定义一个 case class
case class Person(name:String, sex:String, age:Int, x1:String, x2:String)

object dataFrame_test {

  def main(args: Array[String]): Unit = {

    //创建上下文环境配置对象
    val sparkConf = new SparkConf().setMaster("local[*]").setAppName("sparkSql")
    //创建 SparkSession 对象
    val spark = SparkSession.builder().config(sparkConf).getOrCreate()
    //获取sparkContext对象
    val sc: SparkContext = spark.sparkContext
    // 读取文件数据
    // RDD
    val data = sc.textFile("E:\\IntelliJ IDEA\\IdeaProjects\\dzh\\SparkSQL_data\\people.dat").map(x=>x.split("::
"))
    //定义一个样例类
    //将rdd与样例类进行关联
    val peopleRDD = data.map(x=>Person(x(0), x(1), x(2).toInt, x(3), x(4)))

    //将RDD手动转换成dataframe
    //需要手动导入隐式转换，“mySpark”需要与上面构建的sparkSession对象保持一致
    import spark.implicits._
    //    val moviesDF: DataFrame = moviesRDD.toDF()
    val people3_DF = peopleRDD.toDF()
    people3_DF.printSchema()
    people3_DF.show(5)
    people3_DF.head(3)
    people3_DF.collect()
    spark.stop()
  }
}
```

5.Hive中的表创建DataFrame

下次实践过再补齐~

END