### 4.2.3 HBase数据库

由于 org.apache.hadoop.hbase.mapreduce.TableInputFormat 类的实现，Spark 可以通过Hadoop输入格式访问HBase。这个输入格式会返回键值对数据，其中键的类型为org. apache.hadoop.hbase.io.ImmutableBytesWritable，而值的类型为org.apache.hadoop.hbase.client.

Result。

（1）添加依赖

<dependency>

<groupId>org.apache.hbase</groupId>

<artifactId>hbase-server</artifactId>

<version>1.3.1</version>

</dependency>

<dependency>

<groupId>org.apache.hbase</groupId>

<artifactId>hbase-client</artifactId>

<version>1.3.1</version>

</dependency>

（2）从HBase读取数据

package com.atguigu

import org.apache.hadoop.conf.Configuration

import org.apache.hadoop.hbase.HBaseConfiguration

import org.apache.hadoop.hbase.client.Result

import org.apache.hadoop.hbase.io.ImmutableBytesWritable

import org.apache.hadoop.hbase.mapreduce.TableInputFormat

import org.apache.spark.rdd.RDD

import org.apache.spark.{SparkConf, SparkContext}

import org.apache.hadoop.hbase.util.Bytes

object HBaseSpark {

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

//创建spark配置信息

val sparkConf: SparkConf = new SparkConf().setMaster("local[\*]").setAppName("JdbcRDD")

//创建SparkContext

val sc = new SparkContext(sparkConf)

//构建HBase配置信息

val conf: Configuration = HBaseConfiguration.create()

conf.set("hbase.zookeeper.quorum", "hadoop102,hadoop103,hadoop104")

conf.set(TableInputFormat.INPUT\_TABLE, "rddtable")

//从HBase读取数据形成RDD

val hbaseRDD: RDD[(ImmutableBytesWritable, Result)] = sc.**newAPIHadoopRDD**(

conf,

classOf[TableInputFormat],

classOf[ImmutableBytesWritable],

classOf[Result])

val count: Long = hbaseRDD.count()

println(count)

//对hbaseRDD进行处理

hbaseRDD.foreach {

case (\_, result) =>

val key: String = Bytes.toString(result.getRow)

val name: String = Bytes.toString(result.getValue(Bytes.toBytes("info"), Bytes.toBytes("name")))

val color: String = Bytes.toString(result.getValue(Bytes.toBytes("info"), Bytes.toBytes("color")))

println("RowKey:" + key + ",Name:" + name + ",Color:" + color)

}

//关闭连接

sc.stop()

}

}

3）往HBase写入

def main(args: Array[String]) {

//获取Spark配置信息并创建与spark的连接  
 val sparkConf = new SparkConf().setMaster("local[\*]").setAppName("HBaseApp")  
 val sc = new SparkContext(sparkConf)

//创建HBaseConf  
 val conf = HBaseConfiguration.create()  
 val jobConf = new JobConf(conf)  
 jobConf.setOutputFormat(classOf[TableOutputFormat])  
 jobConf.set(TableOutputFormat.OUTPUT\_TABLE, "fruit\_spark")

//构建Hbase表描述器

val fruitTable = TableName.valueOf("fruit\_spark")  
 val tableDescr = new HTableDescriptor(fruitTable)  
 tableDescr.addFamily(new HColumnDescriptor("info".getBytes))

//创建Hbase表  
 val admin = new HBaseAdmin(conf)  
 if (admin.tableExists(fruitTable)) {  
 admin.disableTable(fruitTable)  
 admin.deleteTable(fruitTable)  
 }  
 admin.createTable(tableDescr)

//定义往Hbase插入数据的方法  
 def convert(triple: (Int, String, Int)) = {  
 val put = new Put(Bytes.toBytes(triple.\_1))  
 put.addImmutable(Bytes.toBytes("info"), Bytes.toBytes("name"), Bytes.toBytes(triple.\_2))  
 put.addImmutable(Bytes.toBytes("info"), Bytes.toBytes("price"), Bytes.toBytes(triple.\_3))  
 (new ImmutableBytesWritable, put)  
 }

//创建一个RDD  
 val initialRDD = sc.parallelize(List((1,"apple",11), (2,"banana",12), (3,"pear",13)))

//将RDD内容写到HBase  
 val localData = initialRDD.map(convert)  
  
 localData.saveAsHadoopDataset(jobConf)  
}