|  |  |
| --- | --- |
| [favorite](http://stackoverflow.com/questions/28338694/apache-spark-joins-example-with-java) | **When I use Left Outer Join -- leftOuterJoin() or Right Outer Join -- rightOuterJoin(), both two methods are returning a JavaPairRDD which contains a special type Google Options. But I do not know how to extract the original values from Optional type.**  **Anyways I would like to know can I use same join methods which return the data in my own format. I did not find any way to do that. Meaning is when I am using Apache Spark, I am not able to customize the code in my own style since they already have given all pre-defined things.** |

my 2 sample input datasets

customers\_data.txt:

4000001,Kristina,Chung,55,Pilot

4000002,Paige,Chen,74,Teacher

4000003,Sherri,Melton,34,Firefighter

and

trasaction\_data.txt

00000551,12-30-2011,4000001,092.88,Games,Dice & Dice Sets,Buffalo,New York,credit

00004811,11-10-2011,4000001,180.35,Outdoor Play Equipment,Water Tables,Brownsville,Texas,credit

00034388,09-11-2011,4000002,020.55,Team Sports,Beach Volleyball,Orange,California,cash

00008996,11-21-2011,4000003,121.04,Outdoor Recreation,Fishing,Colorado Springs,Colorado,credit

00009167,05-24-2011,4000003,194.94,Exercise & Fitness,Foam Rollers,El Paso,Texas,credit

public class SparkJoins {

@SuppressWarnings("serial")

public static void main(String[] args) throws FileNotFoundException {

JavaSparkContext sc = new JavaSparkContext(new SparkConf().setAppName("Spark Count").setMaster("local"));

JavaRDD<String> customerInputFile = sc.textFile("C:/path/customers\_data.txt");

JavaPairRDD<String, String> customerPairs = customerInputFile.mapToPair(new PairFunction<String, String, String>() {

public Tuple2<String, String> call(String s) {

String[] customerSplit = s.split(",");

return new Tuple2<String, String>(customerSplit[0], customerSplit[1]);

}

}).distinct();

JavaRDD<String> transactionInputFile = sc.textFile("C:/path/transactions\_data.txt");

JavaPairRDD<String, String> transactionPairs = transactionInputFile.mapToPair(new PairFunction<String, String, String>() {

public Tuple2<String, String> call(String s) {

String[] transactionSplit = s.split(",");

return new Tuple2<String, String>(transactionSplit[2], transactionSplit[3]+","+transactionSplit[1]);

}

});

//Default Join operation (Inner join)

JavaPairRDD<String, Tuple2<String, String>> joinsOutput = customerPairs.join(transactionPairs);

System.out.println("Joins function Output: "+joinsOutput.collect());

//Left Outer join operation

JavaPairRDD<String, Iterable<Tuple2<String, Optional<String>>>> leftJoinOutput = customerPairs.leftOuterJoin(transactionPairs).groupByKey().sortByKey();

System.out.println("LeftOuterJoins function Output: "+leftJoinOutput.collect());

//Right Outer join operation

JavaPairRDD<String, Iterable<Tuple2<Optional<String>, String>>> rightJoinOutput = customerPairs.rightOuterJoin(transactionPairs).groupByKey().sortByKey();

System.out.println("LeftOuterJoins function Output: "+rightJoinOutput.collect());

sc.close();

}

}

Output:

Joins function Output: [(4000001,(Kristina,092.88,12-30-2011)), (4000001,(Kristina,180.35,11-10-2011)), (4000003,(Sherri,121.04,11-21-2011)), (4000003,(Sherri,194.94,05-24-2011)), (4000002,(Paige,020.55,09-11-2011))]

LeftOuterJoins function Output: [(4000001,[(Kristina,Optional.of(092.88,12-30-2011)), (Kristina,Optional.of(180.35,11-10-2011))]), (4000002,[(Paige,Optional.of(020.55,09-11-2011))]), (4000003,[(Sherri,Optional.of(121.04,11-21-2011)), (Sherri,Optional.of(194.94,05-24-2011))])]

LeftOuterJoins function Output: [(4000001,[(Optional.of(Kristina),092.88,12-30-2011), (Optional.of(Kristina),180.35,11-10-2011)]), (4000002,[(Optional.of(Paige),020.55,09-11-2011)]), (4000003,[(Optional.of(Sherri),121.04,11-21-2011), (Optional.of(Sherri),194.94,05-24-2011)])]

When you do left outer join and right outer join, you might have null values. right!

So spark returns Optional object. after getting that result, you can map that result to your own format.

your can use isPresent() method of Optional to map your data.

Here is the example :

JavaPairRDD<String,String> firstRDD = ....

JavaPairRDD<String,String> secondRDD =....

// join both rdd using left outerjoin

JavaPairRDD<String, Tuple2<String, Optional<Boolean>>> rddWithJoin = firstRDD.leftOuterJoin(secondRDD);

// mapping of join result

JavaPairRDD<String, String> mappedRDD = rddWithJoin

.mapToPair(tuple -> {

if (tuple.\_2().\_2().isPresent()) {

//do your operation and return

return new Tuple2<String, String>(tuple.\_1(), tuple.\_2().\_1());

} else {

return new Tuple2<String, String>(tuple.\_1(), "not present");

}

});