Key-Value Pairs (Pair RDDs)

Key-Value Pairs (Pair RDDs)

Spark provides special operations for key-value pairs. These are widely used on the distributed platforms.

- In Python key-value pairs are ***dictionaries***
- In Java and Scala they are ***maps***
- Spark will make use of tuples to create a ***pair RDD***.

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Key-Value Pairs (Pair RDDs)

We often extract fields from an RDD and treat it as a key for the purpose of creating aggregations.

Creating a pair RDD using the first word as the key in Python

```
pairs = lines.map(lambda x: (x.split(" ")[0], x))
```

Key-Value Pairs (Pair RDDs)

Create a key-value pair

In [3]:
```
lines = ["Mary has a cat named Kitty",
         "Jim has a dog named Spot",
         "Sue has a bird name Tweety"]
linesRDD = sc.parallelize(lines)
pairsRDD = linesRDD.map(lambda x: (x.split(" ")[0], x))
print(pairsRDD.collect())
```

```
<class 'pyspark.rdd.PipelinedRDD'>
<class 'list'>
[('Mary', 'Mary has a cat named Kitty'), ('Jim', 'Jim has a dog named Spot'), ('Sue', 'Sue ha
s a bird name Tweety')]
```

Key-Value Pairs (Pair RDDs)

- Examine the tuple created

```
<class 'list'>
[('Mary', 'Mary has a cat named Kitty'), ('Jim', 'Jim has a dog named Spot'), ('Sue', 'Sue ha
s a bird name Tweety')]
```

In [6]:
```
firstElement = pairsRDD.map(lambda x : x[0])
print(firstElement.collect())
```

```
['Mary', 'Jim', 'Sue']
```

In [7]:
```
secondElement = pairsRDD.map(lambda x : x[1])
print(secondElement.collect())
```

```
['Mary has a cat named Kitty', 'Jim has a dog named Spot', 'Sue has a bird name Tweety']
```

Key-Value Pairs (Pair RDDs)

Operations on pair RDDs

- reduceByKey
- groupByKey
- keys
- values
- join

Reference : https://spark.apache.org/docs/latest/rdd-programming-guide.html - working-with-key-value-pairs

Key-Value Pairs (Pair RDDs)

- GroupByKey

  rdd.groupByKey()

  mylist = [(1,3), (1,5), (2,4), (3,4), (2,8)]
  rdd = groupByKey(mylist)

- This returns a tuple like this:
  [(1, [3, 5]),
  (2, [4, 8]),
  (3, [4])

A list of tuples (with a list) – makes printing interesting ……

Key-Value Pairs (Pair RDDs)

```
mylist = [(1,3), (1,5), (2,4), (3,4), (2,8)]
tupleRDD = sc.parallelize(mylist)
groupRDD = tupleRDD.groupByKey()
print(type(groupRDD.collect()[0]))
```

```
<class 'tuple'>
```

```
for tuple in groupRDD.collect():
    print(tuple[0], [v for v in tuple[1]])
```

```
2 [4, 8]
1 [3, 5]
3 [4]
```

Key-Value Pairs (Pair RDDs)

- ReduceByKey – similar to groupByKey except it aggregates/reduces on the worker before shuffle. ReduceByKey is preferred.
- Performs a reduction in the ***lambda*** function.

rdd.reduceByKey(***lambda function***)

The shape returned is a list of tuples:

[(key1, value1),
  (key2, value2),
  ……
  (key3, value3)]

Key-Value Pairs (Pair RDDs)

```
words = ["dog", "dog", "cat", "dog", "cat", "bird"]
wordRDD = sc.parallelize(words)
wordRDDTuple = wordRDD.map(lambda word : (word,1))
reduceByKeyRDD = wordRDDTuple.reduceByKey(lambda x, y : x + y)
#print(type(reduceByKeyRDD.collect()))
#print(type(reduceByKeyRDD.collect()[0]))
```

```
for tuple in reduceByKeyRDD.collect():
    print(tuple[0], tuple[1])
```

```
cat 2
bird 1
dog 3
```

Key-Value Pairs (Pair RDDs)

- Lab – Word Count