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.

Here we extract the first word from a line of text and place it in a ***tuple***:

“001 The quick brown fox “ ---- > ***(001, “001 The quick brown fox”)***

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')]
```

Extracting the Elements of 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

- keys()
- values()
- groupByKey()
- reduceByKey(func)
- mapValues(func)
- sortByKey()
- join

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

Example

```
words = ["one", "two", "two", "three", "three", "three"]

# Create an RDD of tuples (Pair RDD)
wordPairsRDD = sc.parallelize(words).map(lambda word : (word, 1))
```

```
[('one', 1), ('two', 1), ('two', 1), ('three', 1), ('three', 1), ('three', 1)]
```

Pair RDDs – Keys()

rdd = [('one', 1), ('two', 1), ('two', 1), ('three', 1), ('three', 1), ('three', 1)]

rdd.***keys***()

['one', 'two', 'two', 'three', 'three', 'three']

Pair RDDs - Values

rdd = [('one', 1), ('two', 1), ('two', 1), ('three', 1), ('three', 1), ('three', 1)]

rdd.***values***()

[1, 1, 1, 1, 1, 1]

groupByKey() is a *Transformation*

rdd = [('one', 1), ('two', 1), ('two', 1), ('three', 1), ('three', 1), ('three', 1)]

wordCountsWithGroup = rdd.groupByKey()

print (type(wordCountsWithGroup))

<class 'pyspark.rdd.PipelinedRDD'>

print(wordCountsWithGroup.collect())

[('two', <pyspark.resultiterable.ResultIterable object at 0x7fc5cf2c63c8>), ('three', <pyspark.resultiterable.ResultIterable object at 0x7fc5cf2c6400>), ('one', <pyspark.resultiterable.ResultIterable object at 0x7fc5cf2c6438>)]

groupByKey()

```
rdd = [('one', 1), ('two', 1), ('two', 1), ('three', 1), ('three', 1), ('three', 1)]

# GroupByKey then list values (perform an action)

  rdd.groupByKey()    \
             .mapValues(list)


[('two', [1, 1]), ('three', [1, 1, 1]), ('one', [1])]
```

GroupByKey and Sum

rdd = [('one', 1), ('two', 1), ('two', 1), ('three', 1), ('three', 1), ('three', 1)]

rdd.***groupByKey***().mapValues(***sum***)

[('two', 2), ('three', 3), ('one', 1)]

ReduceByKey(function) – A Transformation

rdd = [('one', 1), ('two', 1), ('two', 1), ('three', 1), ('three', 1), ('three', 1)]

rdd.reduceByKey(lambda x, y : x + y)

[('two', 2), ('three', 3), ('one', 1)]

ReduceByKey(function) – A Transformation

```
rdd = [('one', 1), ('two', 1), ('two', 1), ('three', 1), ('three', 1), ('three', 1)]

wordCountsWithReduce = rdd.reduceByKey(lambda x, y : x + y)
print(type(wordCountsWithReduce))
```

```
class 'pyspark.rdd.PipelinedRDD'>
```

ReduceByKey(func) followed by Action

```
wordCountsWithReduce =  rdd.reduceByKey(lambda x, y : x + y)

print(wordCountsWithReduce.collect())

[('two', 2), ('three', 3), ('one', 1)]
```

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)

- Lab – Word Count