flatMap Transformation

In [1]:

```
from pyspark.sql import SparkSession

if __name__ == '__main__':
#Create the spark session
     spark = SparkSession.builder.appName("flatMap transformation").getOrCreate()


#Dataset
data = ["list of strings", "to test", "flatMap", "Transformation", "and compare it", "with map Transformation"]

#Show the dataset
print('Dataset =', data)

rdd = spark.sparkContext.parallelize(data)
print('rdd =', rdd.collect())
```

```
22/10/10 17:20:20 WARN Utils: Your hostname, Vaishalis-MacBook-Pro.local resolves to a loopback address: 127.0.0.1; using 192.168.0.105 instead (on interface en0)
22/10/10 17:20:20 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/10/10 17:20:21 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
22/10/10 17:20:22 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Dataset = ['list of strings', 'to test', 'flatMap', 'Transformation', 'and compare it', 'with map Transformation']
[Stage 0:>                                                          (0 + 10) / 10]
rdd = ['list of strings', 'to test', 'flatMap', 'Transformation', 'and compare it', 'with map Transformation']
```

In [2]:

```
#Printing all the elements in the RDD
for element in rdd.collect():
    print(element)
```

```
list of strings
to test
flatMap
Transformation
and compare it
with map Transformation
```

In [3]:

```
#Map - map() is the transformation takes a function and applies the function to each element of the input RDD.
#The result in the function will become the value of each element in the resultant RDD.

#split() method splits a string into a list. Here, the separator is whitespace.

rdd2 = rdd.map(lambda x: x.split(" "))
for element in rdd2.collect():
    print(element)
print (rdd2.collect())
```

```
['list', 'of', 'strings']
['to', 'test']
['flatMap']
['Transformation']
['and', 'compare', 'it']
['with', 'map', 'Transformation']
[['list', 'of', 'strings'], ['to', 'test'], ['flatMap'], ['Transformation'], ['and', 'compare', 'it'], ['with', 'map', 'Transformation']]
```

In [4]:

```
#Flatmap - flatMap() is the transformation that takes a function and applies the function to each elements of the RDD as in the map() function.
#The DIFFERENCE is that flatMap will return multiple values for each element in the source RDD.

rdd3=rdd.flatMap(lambda x: x.split(" "))
for element in rdd3.collect():
    print(element)
print(rdd3.collect())
```

```
list
of
strings
to
test
flatMap
Transformation
and
compare
it
with
map
Transformation
['list', 'of', 'strings', 'to', 'test', 'flatMap', 'Transformation', 'and', 'compare', 'it', 'with', 'map', 'Transformation']
```

In [5]:

```
#Using a function to split the words of each element and return multiple values for each element if their word length is more than 2 letters
def tokenize(x):
    tokens = x.split()
    newlist = []
    for words in tokens:
        if(len(words) > 2):
            newlist.append(words)
    return newlist

rdd4 = rdd.flatMap(lambda x: tokenize(x))
print("rdd4 = ", rdd4)
print("rdd4.collect() = ", rdd4.collect())
```

```
rdd4 =  PythonRDD[3] at RDD at PythonRDD.scala:53
rdd4.collect() =  ['list', 'strings', 'test', 'flatMap', 'Transformation', 'and', 'compare', 'with', 'map', 'Transformation']
```