|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Системы управления данными»

**Практическое занятие № 6**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИМБО-02-22, Ким Кирилл Сергеевич* | (подпись) | |
| Преподаватель | *Алексеева Екатерина Сергеевна, преподаватель* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2025 г.

1. Практическая работа №6
   1. Цель работы

В этой лабораторной работе мы будем создавать PairRDD и работать с различными преобразованиями PairRDD

## Создание PairRDD

### Использование keyBy() для создания PairRDD

#### Создайте списка строк

mydata1 = ["Henry, 42, M", "Jessica, 16, F",

"Sharon, 21, F", "Jonathan, 27, M",

"Shaun, 11, M", "Jasmine, 62, F"]

#### Создайте новый RDD из mydata1, используя метод SparkContext.parallelize(<list>)

myrdd = sc.parallelize(mydata1)

#### Используйте take(5), чтобы убедиться, что RDD был создан

#### Разберите каждую строку на отдельные слова, используя функцию String.split(<delimiter>). В приведенных выше данных, разделителем выступает (",")

.map(lambda line: line.split(","))

#### Используйте take(5), чтобы проверить результат преобразования

#### Используйте keyBy(<function to determine key>), чтобы создать PairRDD. keyBy() будет использовать функцию переданную в параметре для создания ключа. Она будет использовать переданные ему данные в качестве значения.

.keyBy(lambda collection: collection[0])

#### Используйте take(5) чтобы проверить результат преобразования. Каков результат выполнения? Он таков каким вы его ожидали?

###### 

### В этот раз используйте преобразование map(), чтобы достичь того же самого. Многие разработчики предпочитают использовать map(), потому что он гораздо более прямолинеен в плане трансформации данных. Некоторые разработчики предпочитают использовать keyBy(), потому что само имя преобразования громко заявляет, что в качестве его вывода ожидается кортеж (Ключ, Значение). Попробуйте повторить это самостоятельно, прежде чем двигаться дальше.

###### 

### Обратите внимание, что при использовании преобразования map() кортеж создается явно, тогда как при преобразовании keyBy() это ожидаемый результат, поскольку имя метода указывает на то, что будет создан парный кортеж. Какое преобразование использовать — это ваш выбор. Между этими двумя методами нет никакой фактической разницы в производительности.

### Создайте более сложный PairRDD. Возьмите источник данных сверху и создайте RDD формы ("name", ("age", "gender")). "name" и "gender" это строки, тогда как "age" это integer.

#### Преобразуйте каждую последовательность строк в список строк, содержащий все элементы. Используйте метод String.split(",") для разбора строк в индивидуальные элементы.

#### Из каждого списка создайте необходимый сложный кортеж, создав таким образом PairRDD.

#### Используйте take(5) чтобы проверить преобразование.

###### 

### Используйте flatMapValues, чтобы создать пары rdd

#### Расширьте mydata, добавив список любимых цветов для каждого пользователя. Каждый человек выбрал два любимых цвета. Два цвета разделены двоеточием (:)

mydata2 = ["Henry,red:blue",

"Jessica,pink:turquoise",

"Sharon,blue:pink",

"Jonathan,blue:green",

"Shaun,sky blue:red",

"Jasmine,yellow:orange"]

#### Создайте RDD из списка, используя SparkContext.parallelize()

#### Из каждой строки проанализируйте каждый из элементов и создайте список

#### Из списка извлеките первый элемент (name) и второй элемент (favorite colors) чтобы создать PairRDD

#### Используйте RDD.flatMapValues(<функция для создания коллекции из значений>) чтобы преобразовать вышестоящий rdd к форме (name, favorite\_color). Преобразование flatMapValues() ожидает, что ввод будет произведен в форме (key, value) парного кортежа. Такой RDD был создан на предыдущем шаге. Поскольку каждому пользователю было разрешено выбрать два любимых цвета, часть значения парного кортежа будет представлять собой строку из двух цветов, разделенную двоеточием (:). Передайте flatMapValues функцию, которая разделит эту строку на два цвета. Подсказка: Используйте функцию String.split(<разделитель>). flatMapValues() применит переданную функцию к части значений ввода (ключ, значение). Ожидается, что функция создаст коллекцию. flatMapValues() затем "сгладит" каждый из цветов и поместит его в отдельную пару (ключ, значение). Он будет дублировать исходный ключ для каждой новой созданной строки.

#### Используйте take(5), чтобы проверить преобразования.

###### 

## Агрегационные преобразования с PairRDD

### Вычислите сумму возрастов для всех мужчин и сумму возрастов для всех женщин.

#### Используйте mydata1 из предыдущих шагов, чтобы создать PairRDD вида (gender, age)

###### 

#### Пол является ключевым, а возраст — это значение пары rdd, созданной на предыдущем шаге. Используйте reduceByKey (с функцией v1 + v2), чтобы добавить все строки возрастов с одним и тем же ключом. Пол в настоящее время является ключевым, поэтому будут добавлены все значения строк с одинаковым полом.

#### Используйте действие collect(), чтобы посмотреть результат

###### 

### Рассчитайте максимальный возраст для каждого пола. Измените функцию, переданную в reduceByKey(), чтобы вычислить максимальное значение. В Python есть функция max().

### Повторите описанное выше, чтобы рассчитать минимум на этот раз. В Python есть функция min()

###### 

### На этот раз вместо reduceByKey() используйте действие countByKey () для получения вывода типа данных словарь. Выведите словарь.

###### 

### Словарь в Python — это структура данных, состоящая из элементов ключ:значение. Результат должен содержать два элемента, по одному для каждого пола и количество для этого пола.

### Выведите 3 самых старых человека в mydata1.

#### Создайте пару rdd вида (age, name) из mydata1

#### Используйте sortByKey для сортировки по убыванию

#### Преобразуйте значения (age, name) строк в (name, age)

#### Используйте take(3), чтобы вывести 3 самых старых человека

###### 

### Используйте mydata2, создайте отчет, который показывает для каждого цвета всех людей, чей это любимый цвет.

#### Следуйте шагам 1.4 указанным выше, чтобы создать парный кортеж rdd с (person, color) информацией для всех записей о человеке и его любимых цветах в mydata2.

#### Преобразуйте кортеж к виду (color, person)

#### Используйте groupByKey для группировки каждого цвета и создания списка людей, сгруппированных по цветам.

#### Используйте вложенный цикл for для вывода каждого цвета и всех людей, чей это любимый цвет. Добавьте табуляцию или некоторый интервал во внутреннем цикле, чтобы он создавал табличный вывод.

for color in colorLikers.collect():

print(color[0])

for person in color[1]:

print(" ", person)

###### 

### groupByKey() - очень дорогостоящая операция, поскольку она требует, чтобы все разделы обменивались друг с другом всеми своими наборами данных. Нет возможности агрегировать данные до того, как произойдет обмен. Менее дорогостоящим методом является использование вместо этого преобразования aggregateByKey(<начальное значение>, <функция агрегации внутри раздела>, <функция агрегации между разделами>). Метод aggregateByKey предоставляет функцию, которая позволяет разделам агрегировать данные, все еще находясь внутри раздела. Это создает локальный набор данных агрегации внутри раздела. Этот агрегированный набор данных обычно намного меньше и может значительно сократить объем данных, которыми необходимо обмениваться между разделами. Наконец, предусмотрена еще одна функция, которая объединяет данные между разделами, создавая глобальный агрегированный набор данных. Повторите преобразования, чтобы создать отчет с предыдущего шага 2.6. Вместо groupByKey на последнем шаге используйте aggregateByKey().

#### Инициализируйте начальное значение агрегации как пустой список. Функции агрегирования добавят в этот пустой список всех пользователей, которым нравится цвет ключевого значения.

zeroValue = []

#### Создайте функцию seqOp(accumulator, element), которая выполняет локальную агрегацию. accumulator изначально будет содержать zeroValue, другими словами, пустой список. Каждый человек, чей ключ равен агрегируемому ключу, будет передаваться как element. Добавьте элемент в список с помощью метода List.append().

def seqOp(accumulator, element):

accumulator.append(element)

return accumulator

#### Создайте функцию combOp(accumulator1, accumulator2),которая произведет глобальную агрегацию. Каждый раздел будет передавать в эту функцию свой локальный накопленный набор данных. Следовательно, эта функция должна объединять каждый список из каждого раздела, который содержит людей, которым нравится объединяемый цвет ключа. В Python оператор плюс (+) объединяет два списка.

def combOp(accumulator1, accumulator2):

return accumulator1 + accumulator2

#### Создайте RDD из mydata2 с двумя (2) разделами. Второй параметр может быть добавлен в метод parallelize() чтобы принудительно задать количество разделов.

sc.parallelize(mydata2, 2)

#### Проверьте результат выполнения с помощью действия collect().

colorLikers2.collect()

###### 

### Соединение PairRDD

#### Создайте PairRDD из mydata1. Он должен быть в следующей форме: (name, [age, gender]). Назовите его data1RDD.

#### Создайте PairRDD из mydata2. Он должен иметь следующую форму: (name, [color1, color2]). Назовите rdd, data2RDD.

#### Объедините data1RDD с data2RDD. Соблюдайте формат вывода.

data1RDD.join(data2RDD)

#### Преобразуйте объединенный RDD в следующую форму: "name, age, gender, color1, color2.", Чтобы получить доступ к каждому элементу во вложенной структуре данных, объедините индексы в цепочку следующим образом: [index][inside index][way inside index]. Например, если данные представлены в следующей форме, доступ к gender и color2 производится с помощью следующего синтаксиса

myTuple = (name,(List[age, gender], List[color1, color2]))

# indexes [0] [1][0] [1][0][1] [1][1] [1][1][1]

myGender = myTuple[1][0][1]

myColor2 = myTuple[1][1][1]

#### Сохраните результирующий RDD в директорию /user/student/persons.

#### 

# Собирая все это воедино

В нашей базе данных MySQL есть записи авторов, которые опубликовали сообщения. Существует XML-файл, который показывает широту и долготу местоположения, из которого было отправлено сообщение. Для каждого автора существует файл json, который показывает тип телефонов, принадлежащих автору.

Соберите всю эту информацию и создайте отчет, который показывает автора (имя и фамилию), отправившего сообщение (первые несколько слов заголовка сообщения), используя телефон (список телефонов, принадлежащих автору) в местоположении (широта, долгота).

## Подготовьте источники данных

### Изучите таблицы авторов в MySQL

#### Используйте следующую команду для доступа к MySQL:

$ mysql -u student -p

Enter password: # введите student когда будет предложено ввести пароль

#### 

#### Из mysql используйте следующие команды для проверки схемы для таблиц авторов

show databases;

use labs;

show tables;

desc authors;

###### 

### Используйте Sqoop для импорта таблицы авторов в директорию HDFS /user/student/authors

#### Используйте подкоманду импорта sqoop

#### Установите для строки --connect значение jdbc:mysql://localhost/<database name>

#### Установите авторизацию: --username и --password значениями "student"

#### Установите --table чтобы импортировать авторов

#### Установите --target-dir в директорию, в которую вы хотите сохранить данные /user/student/authors

#### Сохраните как текстовый файл используя --as-textfile

sqoop import \

--connect jdbc:mysql://localhost/labs \

--username student --password student \

--table authors --target-dir /user/student/authors \

--as-textfile

#### 

#### Используйте командную строку HDFS или Hue для проверки того, что данные были импортированы

### Изучите таблицу posts из MySQL. Используйте команду DESC, чтобы получить вывод схемы таблицы posts

###### 

### Используйте Sqoop, чтобы импортировать таблицу posts в директорию HDFS /user/student/posts

#### Используйте подкоманду импорта sqoop

#### Установите --connect значение в jdbc:mysql://localhost/<database name>

#### Установите авторизацию: --username и --password в значение "student"

#### Установите --table, чтобы импортировать посты

#### Установите --target-dir в значение директории, в которой импортированные данные будут сохранены в /user/student/posts

#### Сохраните как текстовый файл, используя --as-textfile

sqoop import \

--connect jdbc:mysql://localhost/labs \

--username student --password student \

--table posts --target-dir /user/student/posts \

--hive-drop-import-delims \

--as-textfile

#### 

#### 

#### Используйте командную строку HDFS или Hue для проверки того, что данные были импортированы

### Скопируйте файл author\_phone.json в /user/student/author\_phone.json в HDFS

#### Перейдите в Linux директорию /home/student/Data

#### 

#### Проверьте файл author\_phone.json и просмотрите его содержимое и схему данных

#### 

#### Используйте HDFS -put подкоманду, чтобы сделать копию HDFS.

#### 

### Скопируйте post\_records директорию в /user/student/post\_records директорию HDFS.

#### Перейдите в Linux директорию /home/student/Data/post\_records

#### 

#### Изучите любой из XML-файлов и просмотрите его содержимое и схему данных

#### 

#### Переместитесь обратно в /home/student/Data

#### 

#### Используйте подкоманду HDFS -put для создания копии всего каталога post\_records в HDFS

#### 

## Создайте RDD из четырех (4) источников данных в HDFS

### Создайте authorNameRDD, сопоставляя First Name и Last Name с <id> автора

#### Используя SparkContext.textFile(), прочитайте данные из таблицы импортированных авторов

#### Преобразуйте RDD к виду (<id>, (first\_name, last\_name)) Такой RDD показывает имя и фамилию для каждого автора <id>. Обязательно приведите <id> к целому числу из строки. Результирующий rdd должен выглядеть примерно так, как показано ниже.

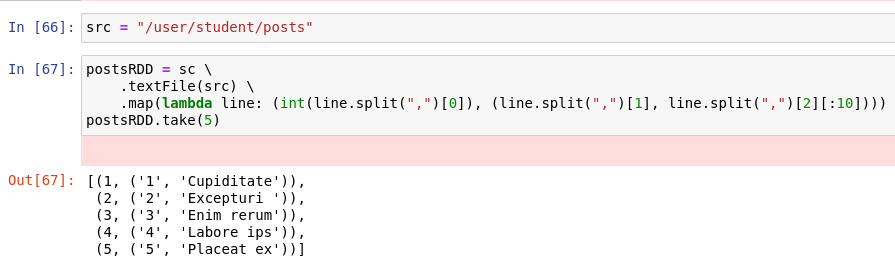
#### 

###### 

### Создайте postsRDD, как отображение post <id> с author\_id из post и первыми несколькими буквами заголовка каждого сообщения

#### Используйте SparkContext.textFile(), считайте данные из таблицы импортированных постов

#### Преобразуйте RDD к виду (<id>, (author\_id, <первые 10 символов title>)) Этот RDD отображает для каждого post <id>, author\_id поста и первые 10 символов заголовка поста. Результирующий rdd должен выглядеть примерно так, как показано ниже:



### Создайте phoneRDD из файла authors\_phone.json. У некоторых авторов есть несколько телефонов, и модель телефона некоторых авторов неизвестна. К сожалению, вместо этого, выводите "Unknown, the data shows an empty string"

#### Импортируйте библиотеку json для разбора записей JSON

#### Файл authors\_phone.json не содержит записей, разделенных новой строкой. Используйте wholeTextFiles чтобы прочитать весь исходный файл

#### Библиотека JSON имеет json.loads(<JSON string>) метод, который создает коллекцию записей JSON. Используйте преобразование flatMap() с функцией json.loads(), чтобы создать новую строку для каждой записи JSON.

#### Каждая запись JSON представлена в виде словаря Python. Словарь — это структура данных ключ:значение. Чтобы получить значение по ключу, используйте метод Dictionary.get(<key name>, None). Этот метод возвращает либо значение для соответствующего ключа, либо None, если ключ не найден. Используйте метод get() и преобразуйте запись JSON в парный кортеж формы (author\_id, phone\_model)

#### Создайте функцию, которая проверяет строку и возвращает "Unkown", если она пустая, заменяет "," на "or" или просто возвращает строку, если и "," отсутствует в строке

def setPhoneName(s):

if s == "": return "Unknown"

elif "," in s: return s.replace(",", " or")

else: return s

#### Используя функцию setPhoneName(), преобразуйте (author\_id, phone\_model) применяя setPhoneName к phone\_model. Результатом по-прежнему должен быть парный кортеж формы (author\_id, <измененная модель телефона после применения setPhoneName>). Результирующий rdd должен выглядеть примерно так, как показано ниже:

###### 텍스트이(가) 표시된 사진 자동 생성된 설명

### Создайте latlongRDD из файла XML в ~/Data/post\_records. Вывод должен быть в форме (post\_id, (latitude, longitude))

#### Файл XML содержит post\_id и поля местоположения. Поле местоположения представляет собой разделенное запятыми поле, показывающее широту и долготу. Используйте wholeTextFiles(), чтобы прочитать файл XML. wholeTextFiles() возвращает кортеж вида (<путь к файлу>, <содержимое XML>). Содержимое XML включает в себя множество XML записей.

#### Определите вспомогательную функцию getPosts(XML строка). Эта функция разберет содержимое XML и вернет коллекцию записей XML.

import xml.etree.ElementTree as ET

def getPosts(s):

posts = ET.fromstring(s)

return posts.iter("record")

#### getPosts(<содержимое XML>) вернет коллекцию XML-записей. <Содержимое XML> находится в части значений кортежа, созданного с помощью wholeTextFiles(). Чтобы создать новую строку для каждой записи XML в коллекции, возвращаемой getPosts(), используйте flatMap() вместо map() для применения преобразования.

#### Определите вспомогательную функцию getPostID(element). Эта функция прочитает XML элементы и вернет <post\_id> поле как строку.

def getPostID(elem):

return elem.find("post\_id").text

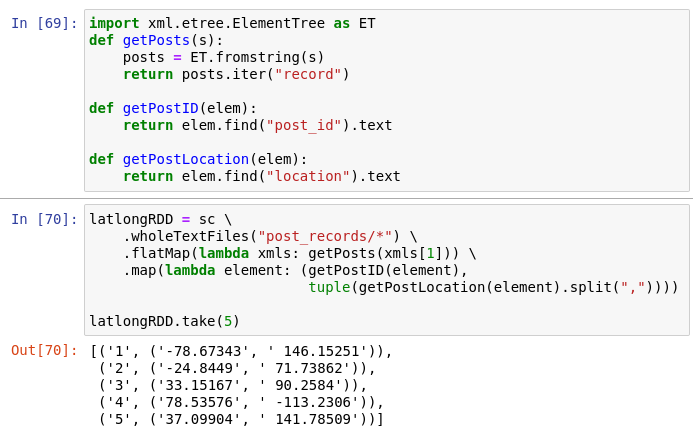
#### Определите вспомогательную функцию getLocation(element). Эта функция считывает XML-элемент и возвращает поле <location> в виде строки.

def getPostLocation(elem):

return elem.find("location").text

#### Используя две только что созданные вспомогательные функции, проанализируйте post\_id и информацию о местоположении из каждой записи XML. Преобразуйте RDD к форме (<post\_id>, <location>) парного rdd.

#### Информация <location> содержит информацию о широте, долготе с разделителем в виде запятой (","). Используйте String.split(<delimiter>) метод чтобы преобразовать PairRDD (<post\_id>, <location>) к (<post\_id>, (<latitude>,<logitude>)). Результирующий rdd должен выглядеть примерно так, как показано ниже:



## Комбинирование и объединение данных для получения более подробной информации

### Объедините authorNameRDD с phoneRDD, чтобы имя автора совпадало с его телефоном (телефонами).

#### Используйте преобразование RDD1.join(RDD2), чтобы объединить authorNameRDD с phoneRDD по author\_id ключу. Назовите новый RDD, authorPhoneRDD. authorPhoneRDD будет выглядеть следующим образом (<author\_id>, ((<first\_name>,<last\_name>), <phone model names>))

#### Воспользуйтесь take(5) чтобы проверить результат. Итоговый результат должен быть похож на приведенный ниже.

#### Используйте count(), чтобы убедиться, что существует 10000 записей

#### 텍스트이(가) 표시된 사진 자동 생성된 설명

#### Преобразуйте формат authorPhoneRDD к (<author\_id>, [<first\_name>,<last\_name>,<phone model names>]). Назовите новый RDD как authorNamePhoneRDD, который должен выглядеть примерно так, как показано ниже:

###### 

### Объедините postsRDD с latlongRDD чтобы информация о каждом посте соответствовала информации о геолокации поста.

#### Используйте преобразования RDD1.join(RDD2), чтобы соединить postsRDD с latlongRDD. Назовите новый RDD как postLocationRDD. Он будет иметь форму (<post\_id>, ((<author\_id>,<title>), (<latitude>,<longitude>))). Результат должен быть похож на приведенный ниже:

#### 텍스트이(가) 표시된 사진 자동 생성된 설명

#### Преобразуйте формат postLocationRDD к (<author\_id>, [<title>,<latitude>,<longitude>]) и назовите его authorPostLocationRDD. Результат должен быть похож на приведенный ниже:

### Объедините authorNamePhoneRDD с authorPostLocationRDD.

#### Назовите результат объединения как nameTitlePhoneLocRDD. Полученный результат будет иметь вид (<author\_id>, ([<first\_name>,<last\_name>,<phone\_model\_names>], [<title>,<latitude>,<longitude>])).

#### 

#### Преобразуйте приведенные выше выходные данные в кортеж из ("string", "string") типов данных. Создайте две строки, объединив различные фрагменты информации. Используйте Python операцию плюс (+), чтобы конкатенировать стоковые литералы со значениями. ("<first\_name> <last\_name> on <phone model names>" , "Posted <title> from lat: <latitude> lon: <longitude>"). Итоговый результат должен выглядеть примерно так, как показано ниже:

###### 테이블이(가) 표시된 사진 자동 생성된 설명

### Сгруппируйте все посты от одного и того же автора.

#### Сначала воспользуйтесь groupByKey() по имени в TitlePhoneLocRDD. Это очень дорогостоящая операция. На самом деле, поскольку она такая дорогая, Spark выполняет это действие лениво. Вместо того, чтобы возвращать фактические результаты, Spark возвращает Resultiterable объекты. Эти объекты являются лениво итерируемыми, и их значения еще не были вычислены.

#### 

###### 텍스트이(가) 표시된 사진 자동 생성된 설명

#### Повторите, используя преобразование aggregateByKey(). Как должна выглядеть функция последовательности? Как насчет комбинированной функции? Конечный результат будет выглядеть примерно так, как показано ниже.

#### 