

Министерство науки и высшего образования Российской Федерации

Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

ЯЗЫК PIŁ LATIN

Методические указания по выполнению домашней работы №2
по курсу «Технологии обработки больших данных»

Калуга – 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	3
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ.....	4
КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ	5
УСТАНОВКА RIG	15
РАБОТА С RIG	18
ПРИМЕРЫ RIG-СКРИПТОВ	20
ЗАДАНИЕ НА ДОМАШНЮЮ РАБОТУ	28
ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ.....	28
ВАРИАНТЫ ЗАДАНИЙ.....	28
КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ	31
ФОРМА ОТЧЕТА ПО ДОМАШНЕЙ РАБОТЕ.....	31
ОСНОВНАЯ ЛИТЕРАТУРА.....	32
ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА	32

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения домашних работ по курсу «Технологии обработки больших данных» на кафедре «Программное обеспечение ЭВМ, информационные технологии и прикладная математика» факультета фундаментальных наук Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 4-го курса направления подготовки 09.03.04 «Программная инженерия», содержат краткое описание синтаксиса языка Pig Latin, а также примеры решения задач и задание на выполнение домашней работы.

Методические указания составлены для ознакомления студентов с языком обработки данных Pig Latin. Для выполнения домашней работы студенту необходимы минимальные знания по программированию на высокоуровневом языке программирования (Java, Python или др.), базовые знания SQL

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения домашней работы является формирование практических навыков реализации pig-скриптов для обработки больших данных.

Основными задачами выполнения домашней работы являются:

1. Получить навыки обработки больших данных, используя Pig Latin.
2. Изучить принцип работы Pig Latin.
3. Изучить синтаксис Pig Latin.
4. Уметь писать запросы, комбинируя несколько источников данных.

Результатами работы являются:

- Входные файлы с данными
- Pig-скрипт работы с данными
- Выходные файлы с результатами вычислений
- Подготовленный отчет

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Apache Pig – высокоуровневая платформа для создания SQL подобных программ, запускаемых на Hadoop кластере. Apache Pig состоит непосредственно из среды выполнения программы и SQL подобного языка под названием Pig.

Концепция MapReduce довольно трудозатратна для исполнения программистом сама по себе, особенно когда нужно связать несколько MapReduce задач между собой в одну длинную цепочку. Apache Pig меняет такое положение вещей, создавая более простую абстракцию процедурного языка над платформой MapReduce. Вместо того, чтобы создавать отдельное приложение MapReduce, вы можете написать небольшой сценарий на Pig, который автоматически распараллеливается и распределяется между узлами кластера.

Сценарий Pig включает серию операций (преобразований, трансформаций), которые необходимо применить к входным данным, чтобы получить выходные данные. Эти операции описывают поток данных, который затем преобразуется (компилируется) исполнительной средой Pig Latin в исполняемое представление в виде серии MapReduce задач и запускается для выполнения.

Основной объект в Pig Latin – это «отношение». Именно с отношениями работают все операторы языка. В форме отношений представляются входные и выходные данные.

Каждое отношение представляет собой набор однотипных объектов — «кортежей» (tuples). Аналоги в реляционных БД: кортеж — это строка, отношение — это таблица. Кортежи соответственно состоят из нумерованных или именованных объектов — «полей», произвольных базовых типов (число, строка и т.д.). В таблице 1 приведены типы данных, с которыми умеет работать Apache Pig. Все приведённые типы могут быть вам знакомы из высокоуровневых языков программирования и реляционных БД.

Таблица 1

Типы данных Pig Latin

Простые типы данных	Описание	Пример
int	32-битное целое со знаком	10
long	64-битное целое со знаком	10L
float	32-битное число с плавающей точкой	10.5f
double	64-битное число с плавающей точкой	10.5
Массивы		
chararray	Массив символов в формате UTF-8	hello world
bytearray	Массив байт (blob)	
Составные типы данных		
tuple	Кортеж – упорядоченное множество полей	(19,2)
bag	Коллекция кортежей	{(19,2), (18,1)}
map	Множество пар ключ-значение	['name'#'Alex', 'age'#40]

Обратите внимание, что в составном типе map, который определяет множество пар ключ-значение, используется разделить «#», а не привычный программистам «:».

Обычно программу Pig Latin можно разделить на 3 части:

1. Загрузка данных в программу;
2. Трансформация данных;
3. Вывод данных на экран, либо в файл.

Загрузка данных

Для загрузки данных из локальной файловой системы или распределённой HDFS используется оператор LOAD.

Синтаксис выражения:

```
variable = LOAD 'path' USING function AS schema;
```

Где:

- variable – Наименование отношения (переменная, к которой впоследствии можно обратиться по её имени)
- path – Путь к файлу, либо к директории, которая содержит файлы исходных данных. Если указана директория, то будут загружены все файлы, содержащиеся в ней. Для загрузки из HDFS используются пути типа «hdfs:///directories/file», а для загрузки из локальной файловой системы «file:///directories/file». По умолчанию будет выбрана файловая система, доступная в данном режиме запуска скрипта Pig (см. далее). Указание файловой системы не обязательно применять всегда, это необходимо только для уточнения нахождения файлов.
- function – Функция для загрузки данных. Доступны 4 встроенных варианта: BinStorage, JsonLoader, PigStorage, TextLoader (о них будет сказано далее). Также можно использовать пользовательские функции. По умолчанию используется PigStorage, в параметрах которого необходимо указать разделитель данных (например, «;» для csv файлов).
- schema – Описание формата входных данных в контексте типов данных, доступных в Pig. Например:
(id: int, name: chararray, weight: float, ...);

Пример загрузки данных:

```
all_students = LOAD 'student.txt'  
USING PigStorage(',')  
AS (name:chararray, age:int, gpa:float);
```

Программа на Pig Latin состоит из выражений, каждое из которых завершается символом «;», не забывайте ставить его в конце каждой строки.

Функции загрузки и сохранения

Описание всех функций для загрузки и сохранения данных приведены в таблице 2.

Таблица 2

Функции загрузки и сохранения Pig Latin

Функция	Описание
PigStorage([delimiter])	Используется для загрузки и сохранения данных в структурированные текстовые файлы. В качестве разделителя для полей используется delimiter (по умолчанию '\t')
BinStorage()	Используется для загрузки и сохранения данных в бинарные файлы.
JsonLoader(['schema'])	Используется для загрузки данных из JSON файлов. Параметр 'scheme' – задаёт формат хранимых данных.
JsonStorage()	Используется для сохранения данных в JSON файлы. Схема определяется Pig автоматически.
PigDump()	Используется для сохранения данных в кодировке UTF-8.
TextLoader()	Используется для загрузки неструктурированных данных (тексты) в кодировке UTF-8.
HBaseStorage('columns', ['options'])	Используется для загрузки и сохранения данных в HBase таблицу.
AvroStorage(['schema record name'], ['options'])	Используется для загрузки и сохранения данных в Avro файлы.
AccumuloStorage(['columns', 'options'])	Используется для загрузки и сохранения данных в Accumulo таблицу.

Операторы отношений

Основной способ трансформации данных в Pig Latin – операторы отношений. В Pig Latin результатом любого оператора является отношение, представляющее собой набор кортежей. Основные операторы отношений представлены в таблице 3.

Таблица 3

Операторы отношений

Оператор	Пример	Описание
FILTER	out = FILTER in BY field;	Отбирает кортежи из отношения in, основываясь на значении поля field
ORDER BY	out = ORDER in BY field [ASC DESC];	Сортирует отношение in по одному или нескольким полям
GROUP BY	out = GROUP in BY field;	В рамках одного отношения группирует кортежи с одинаковым ключом группировки.
UNION	out = UNION in1, in2 [, in3 ...];	Создаёт объединение двух и более отношений
DISTINCT	out = DISTINCT in;	Удаляет повторяющиеся кортежи
LIMIT	out = LIMIT in limit_number	Берёт первые limit_number кортежей из отношения
FOREACH	out = FOREACH in GENERATE expression [AS alias];	Проходит по всем кортежам отношения in и генерирует новые кортежи с именем alias
SAMPLE	out = SAMPLE in factor;	Создает случайную выборку из кортежей размером factor
INNER JOIN	out = JOIN in1 BY field1, in2 BY field2;	Выполняет операцию внутреннего

		объединения (как в реляционных БД)
LEFT OUTER JOIN	out = JOIN in1 BY field1 LEFT, in2 BY field2 LEFT	Выполняет операцию левого внешнего объединения
RIGHT OUTER JOIN	out = JOIN in BY field1 RIGHT, in2 BY field2 RIGHT	Выполняет операцию правого внешнего объединения
FULL OUTER JOIN	out = JOIN in1 BY field1 FULL, in2 BY field2 FULL	Выполняет операцию полного внешнего объединения
FLATTEN	FLATTEN(in)	Используется для приведения сложных типов данных к «плоскому» виду, аналог оператора распаковки «*» в python

Встроенные функции

Pig содержит множество встроенных функций. Список основных функций приведён в таблице 4.

Таблица 4

Основные функции Pig Latin

Синтаксис	Описание
AVG(expression)	Возвращает среднее значение массива
BagToString(vals:bag, [delimiter:chararray])	Конвертирует коллекцию кортежей bag в строку, используя разделитель delimiter
BagToTuple(expression)	Конвертирует коллекцию кортежей в один кортеж
CONCAT (expression, expression, [...expression])	Выполняет конкатенацию выражений
COUNT(expression)	Возвращает количество элементов в коллекции кортежей

DIFF (expression, expression)	Выполняет сравнение двух выражений
IsEmpty(expression)	Проверка элемента на пустоту
MAX(expression)	Вычисляет максимальное значение в массиве
MIN(expression)	Вычисляет минимальное значение в массиве
SIZE(expression)	Возвращает количество элементов в коллекции
SUBTRACT(bag1, bag2)	Возвращает элементы, присутствующие в коллекции bag1, но отсутствующие в коллекции bag2
SUM(expression)	Вычисляет сумму элементов в массиве
TOKENIZE(expression [, 'field_delimiter'])	Разделяет строку и возвращает коллекцию слов

Операторы сравнения и математические операторы

В Pig также как и в высокоуровневых языках программирования поддерживаются арифметические, логические и операторы сравнения, перечисленные в таблице 5.

Таблица 5

Арифметические, логические и операторы сравнения в Pig

Оператор	Символ в Pig	Пример
Сложение	+	FOREACH in GENERATE f1 + f2
Вычитание	-	FOREACH in GENERATE f1 - f2
Умножение	*	FOREACH in GENERATE f1 * f2
Деление	/	FOREACH in GENERATE f1 / f2
Модуль от деления	%	FOREACH in GENERATE f1 % f2
Логическое И	and	FILTER in BY (f1 == 'NY') and

		(f2 == 'MI')
Логическое ИЛИ	or	FILTER in BY (f1 == 8) or (f1 == 10)
Логическое НЕ	not	FILTER in BY NOT (f1 > 5)
Содержание	in	FILTER in BY f1 IN ('Kaluga', 'Moscow', 'Tula')
Проверка на null	is null	FILTER in BY field is not null
Равенство	==	field == 'Name'
Не равенство	!=	field != 'Oksana'
Меньше	<	age < 18
Меньше или равно	<=	age <= 18
Больше	>	age > 18
Больше или равно	>=	age >= 18

Больше операторов вы можете узнать на официальном сайте проекта: <https://pig.apache.org/>

Смотрите документацию, раздел Pig Latin Basics для вашей версии: <https://pig.apache.org/docs/r0.17.0/basic.html>

Доступ к полям отношений может осуществляться не только по имени поля, но и по индексу поля в кортеже, что иногда может быть полезно, особенно когда считываешь данные от другого пользователя. Пример обращений приведёт в таблице 6.

Таблица 6

Свойства полей отношения

	Первое поле	Второе поле	Третье поле
Тип данных	chararray	int	float
Обращение по индексу	\$0	\$1	\$2

Обращение по имени	name	age	gpa
--------------------	------	-----	-----

Сохранение данных

Для сохранения данных в локальной файловой системе или HDFS используется оператор STORE.

Синтаксис выражения:

```
STORE variable INTO 'path' USING function;
```

Все переменные идут по аналогии со считыванием данных. В качестве функции может использоваться: BinStorage, JsonStorage, PigStorage, PigDump.

Пример:

```
STORE students INTO '$$DT/all_students.txt'
USING PigStorage(',');
```

Предпочтительно сохранять результаты каждого выполнения скрипта в отдельную директорию, имя которой включает дату и время исполнения. Для этого можно воспользоваться функцией вызова произвольной шелл-команды прямо из Pig-скрипта (ее нужно написать в обратных кавычках).

В примере выше использовалась переменная DT, которая определена следующим образом:

```
%declare DT `date +D%y%m%dT%H%M`
```

Здесь:

%declare – объявление внешней переменной;

DT – имя переменной;

date +D%y%m%dT%H%M – shell-команда, которая выдаёт текущую дату и время в представленном формате.

Для последующего получения значения, хранящегося в переменной DT используется запить \$\$DT.

Операторы диагностики

Pig поддерживает несколько операторов диагностики, которые можно использовать для отладки сценариев Pig. Оператор DUMP отображает данные на экране и оказывается чрезвычайно полезным, если вы хотите увидеть не только данные, но и их схему хранения. Также можно использовать оператор DESCRIBE для получения подробного описания схемы отношения (поле и тип) или оператор ILLUSTRATE для красивого вывода таблицы с описанием схемы отношения и хранящихся данных.

Оператор EXPLAIN чуть более сложен, но очень полезен. Можно использовать этот оператор для указанного массива, чтобы увидеть, как физические операторы сгруппированы в задания map и reduce (т. е. как были получены данные).

Таблица 7 содержит список операторов диагностики Pig Latin и их описания.

Таблица 7

Операторы диагностики Pig Latin

Оператор	Описание
DUMP	Выводит содержимое массива на экран
DESCRIBE	Возвращает схему отношения
ILLUSTRATE	Выводит содержимое отношения и схему его хранения на экран
EXPLAIN	Показывает планы исполнения MapReduce.

Определяемые пользователями функции

Pig является мощным и полезным языком, его возможности можно расширять с помощью определяемых пользователями функций (User-Defined Functions, UDF). В сценариях Pig можно использовать ваши собственные функции, разработанные для определенных задач, например, для анализа входных данных или форматирования результирующих данных и даже операторов. Пользовательские функции часто пишутся на языке Java и позволяют Pig поддерживать специализированную обработку, а кроме того, являются способом расширения возможностей Pig для выполнения конкретных задач.

Также поддерживаются пользовательские функции, написанные на языке Python и Scala. Далее в примере будет продемонстрировано подключение Python функции в выполняемый скрипт.

Для того, чтобы функцию можно было использовать, её необходимо предварительно зарегистрировать и определить псевдоним. Ниже приведён пример для функции `check_content`, содержащейся в файле `checker.py`:

```
REGISTER 'path_to_script/checker.py'  
using jython as checker;
```

Для вызова функции используется подобные конструкции:

```
is_ok = FOREACH data GENERATE  
checker.check_content(text) AS scheme;
```

Обязательно необходимо указать схему данных (`scheme`), в которые поступают данные из python-скрипта. При указании схемы используются ключевые слова, определяющие типы данных. Пример схемы возвращаемых данных:

```
bag_name:bag{tuple_name:tuple(character:chararray,  
count:int)};
```

УСТАНОВКА PIG

Необходимое ПО

Для установки Pig необходима операционная система Linux. Далее будет рассмотрена установка и настройка Pig для Ubuntu (возможно использование виртуальной машины).

Для работы Pig требуется следующее программное обеспечение:

Java 8 JDK (и выше)

Сконфигурированный кластер Hadoop (необходим только для запуска Pig в режиме MapReduce)

Установка и настройка Hadoop подробно описана в методических указаниях к домашней работе №1.

Проверить наличие установленной версии Java можно командной:

```
echo $JAVA_HOME
```

Если в результате выполнения данной команды не будет ничего выведено, то необходимо установить Java командной:

```
sudo apt-get install openjdk-8-jdk
```

или

```
sudo apt-get install default-jdk
```

Затем добавить в файл `~/.bashrc` следующую строку:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Для того, чтобы изменения вступили в силу, необходимо выполнить:

```
source ~/.bashrc
```

Скачивание Pig

Скачать последнюю версию Pig можно с сайта: <https://pig.apache.org/releases.html>

Необходимо скачать архив интересующей вас версии Pig вручную с сайта или сделать это командой:

```
sudo wget https://dlcdn.apache.org/pig/pig-{version}/
```

После скачивания архива, его необходимо распаковать и переместить файлы в удобный каталог (например, `/usr/local/pig`), для разархивации используется команда:

```
sudo tar xzvf pig-0.17.0.tar.gz
```


Настройка переменных окружения

В файл `~/.bashrc` необходимо добавить строку для возможности запуска Pig в консоли из любого места:

```
export PATH=/usr/local/pig/bin:$PATH
```

Для того, чтобы изменения вступили в силу, необходимо выполнить:

```
source ~/.bashrc
```

Проверка установки

Проверить установку Pig, можно с помощью следующей команды:

```
pig -version
```

Если установка прошла успешно, будет выведена версия Pig, как представлено ниже:

```
Apache Pig version 0.17.0 (r1797386)  
compiled Jun 02 2017, 15:41:58
```

РАБОТА С PIG

Режимы выполнения

В Pig есть 6 видов режимов выполнения:

- Local Mode – режим работы с локальной файловой системой. Для запуска скриптов в этом режиме используется следующая команда:

```
pig -x local script_name
```

- Tez Local Mode – этот режим похож на локальный за исключением того, что внутри Pig будет вызывать Tez runtime движок (см. официальную документацию).

```
pig -x tez_local script_name
```

- Spark Local Mode – этот режим похож на локальный за исключением того, что внутри Pig будет вызывать Spark runtime движок (см. официальную документацию).

```
pig -x spark_local
```

- Mapreduce Mode – для запуска Pig в этом режиме необходимо иметь запущенный кластер Hadoop и установленные переменные (\$HADOOP_HOME и другие) в файле ~/.bashrc. Этот режим является режимом по умолчанию.

```
pig -x mapreduce
```

- Tez Mode - для запуска Pig в этом режиме необходимо иметь доступ к кластеру Hadoop со сконфигурированным фреймворком Apache Tez. Для перехода в этот режим используется следующая команда:

```
pig -x tez
```

- Spark Mode - для запуска Pig в этом режиме необходимо иметь доступ к кластерам Spark, YARN или Mesos и к кластеру Hadoop. Для перехода в этот режим используется следующая команда:

```
pig -x spark
```

Примечание: Spark Local Mode является экспериментальным. Некоторые запросы выдают ошибку на большом объеме данных в локальном режиме.

Интерактивный режим

Запустить Pig в интерактивном режиме можно с помощью оболочки Grunt. Вызвать оболочку Grunt можно командой `pig` в терминале, при этом по умолчанию будет использоваться MapReduce режим (укажите явно необходимый режим):

```
pig -x local
```

В интерактивном режиме команды вводятся по одной за раз (как в интерактивном режиме Python):

```
pig -x local
grunt> A = load 'passwd' using PigStorage(':');
grunt> B = foreach A generate $0 as id;
grunt> dump B;
```

Пакетный режим (Batch mode)

В данном режиме все команды прописываются в отдельном файле. Создадим файл `id.pig`, в который поместим следующие команды:

```
A = load 'passwd' using PigStorage(':');
B = foreach A generate $0 as id;
store B into 'id.out';
```

Чтобы выполнить скрипт Pig из файла необходимо выполнить команду:

```
pig -x local id.pig
```

Результаты работы данного скрипта будут помещены в файл `id.out`

ПРИМЕРЫ

Пример 1

База данных твитов состоит из двух файлов:

Файл `tweets.csv` имеет формат: `tweet_id, tweet, login`

Файл `users.csv` имеет формат: `login, user_name, state`

В каждом штате найти пользователя с наибольшим числом твитов.

Загрузка данных

```
users = LOAD 'users.csv' USING PigStorage(',')
AS (login:chararray,
    user_name:chararray,
    state:chararray);

tweets = LOAD 'tweets.csv' USING PigStorage(',')
AS (tweet_id:long,
    tweet:chararray,
    login:chararray);
```

LOAD 'users.csv' USING PigStorage(',') – выполняет операцию загрузки данных из файла 'users.csv'. Оператор **LOAD** используется для чтения данных из внешних источников. **PigStorage(',')** указывает, что данные в файле разделены запятыми (,).

AS (login:chararray, user_name:chararray, state:chararray) – определяет схему данных для переменной `users`. Каждая запись данных из файла 'users.csv' будет содержать три поля, которые будут названы соответственно: `login`, `user_name` и `state`. Тип данных для каждого поля `chararray`.

Подсчитаем количество твитов для каждого пользователя

```
tweets_count = FOREACH (GROUP tweets BY login)
GENERATE group AS login, COUNT(tweets) AS t_count;
```

GROUP tweets BY login – использует оператор **GROUP**, все записи с одинаковым значением `login` будут сгруппированы вместе.

FOREACH (...) GENERATE group AS login, COUNT(tweets) AS t_count – после группировки используется оператор **FOREACH**, чтобы пройти по каждой группе данных. Внутри **FOREACH** используется **GENERATE** для создания новой структуры данных.

Выполним объединение по полю login

```
tweets_count = JOIN tweets_count BY login, users BY login;
```

Сгруппируем полученное отношение по штатам

```
g = GROUP tweets_count BY state;
```

Найдём в каждом штате пользователя с наибольшим числом твитов

```
max_t_count = FOREACH g {  
  counts=ORDER tweets_count BY tweets_count::t_count  
  DESC;  
  top_counts = LIMIT counts 1;  
  GENERATE FLATTEN(top_counts);}
```

FOREACH g {...} – использует **FOREACH** для обработки каждой группы данных.

counts = ORDER tweets_count BY tweets_count::t_count DESC – используется оператор **ORDER**, чтобы отсортировать записи в каждой группе по убыванию значения `t_count`.

top_counts = LIMIT counts 1 – после сортировки используется оператор **LIMIT**, чтобы выбрать только первую запись, что

соответствует записи с максимальным значением `t_count` в каждой группе.

GENERATE FLATTEN(top_counts) – **FLATTEN** используется для “разглаживания” вложенных структур данных. Если в наборе данных есть поля, содержащие кортежи или множества, то оператор **FLATTEN** позволяет преобразовать эти вложенные структуры в отдельные записи.

Выберем необходимые поля для вывода результата (штат, логин и количество твитов)

```
max_t_count = FOREACH max_t_count GENERATE
users::state,tweets_count::login,tweets_count::t_count;
```

Сохранение результатов в папку `max_t_count`

```
STORE max_t_count INTO 'max_t_count';
```

Листинг примера целиком

```
users = LOAD 'users.csv' USING PigStorage(',')
AS (login:chararray,
    user_name:chararray,
    state:chararray);

tweets = LOAD 'tweets.csv' USING PigStorage(',')
AS (tweet_id:long,
    tweet:chararray,
    login:chararray);

tweets_count = FOREACH (GROUP tweets BY login)
GENERATE group AS login, COUNT(tweets) AS t_count;

tweets_count = JOIN tweets_count BY login, users BY
login;
```

```

g = GROUP tweets_count BY state;

max_t_count = FOREACH g {
  counts = ORDER tweets_count BY tweets_count::t_count
DESC;
  top_counts = LIMIT counts 1;
  GENERATE FLATTEN(top_counts);
}

max_t_count      =      FOREACH      max_t_count      GENERATE
users::state,      tweets_count::login,
tweets_count::t_count;

STORE max_t_count INTO 'max_t_count';

```

Пример 2 – Использование PigLatin для решения задач MapReduce

Подсчитать количество появлений каждого буквенного символа в файле. Подсчет должен быть регистро-зависимым (т.е. буквы «а» и «А» считаются разными). Результат сохранить в файле.

Для наглядности показана часть вывода результата с помощью команды *dump* при построчном выполнении кода.

Загрузка данных из файла 'book.txt':

```
lines = load 'book.txt' as (line);
```

```

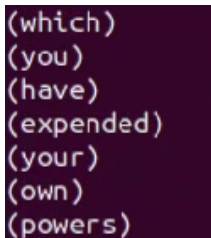
(I hadn't told him who owned the car. His hand was on a revolver in
his)
(pocket every minute he was in the house—" He broke off defiantly.)
("What if I did tell him? That fellow had it coming to him. He thre
w)
(dust into your eyes just like he did in Daisy's, but he was a toug
h)
(one. He ran over Myrtle like you'd run over a dog and never even)
(stopped his car.")
()
(There was nothing I could say, except the one unutterable fact tha
t it)

```

Обработка считанных данных:

```
words = foreach lines generate flatten(TOKENIZE(line))  
as word;
```

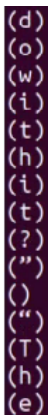
FOREACH для обработки каждой строки данных из lines. TOKENIZE(line) разбивает каждую строку на слова.



```
(which)  
(you)  
(have)  
(expended)  
(your)  
(own)  
(powers)
```

```
characters      =      foreach      words      generate  
flatten(TOKENIZE(REPLACE(word,  ' ',  '|'),  '|'))  as  
character;
```

Каждое слово разбивается на символы с использованием TOKENIZE(REPLACE(word, " ", "|"), "|").



```
(d)  
(o)  
(w)  
(i)  
(t)  
(h)  
(i)  
(t)  
(?)  
(")  
(')  
(")  
(T)  
(h)  
(e)
```

```
filter_characters = filter characters by (character  
matches '.*[a-zA-Z].*');
```

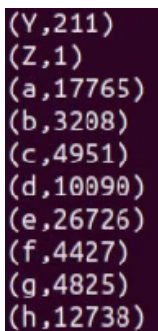
Фильтрация символов с использованием FILTER, чтобы оставить только буквенные символы.


```
new_characters = foreach filter_characters generate
character;
```

Создает новый набор данных, содержащий только буквенные символы.

```
group_characters = foreach (group new_characters by
character) generate group as character,
COUNT(new_characters) as count;
```

Используется оператор GROUP для группировки данных по символам. Затем с помощью FOREACH и GENERATE для каждой группы подсчитывается количество символов с использованием COUNT.



```
(Y,211)
(Z,1)
(a,17765)
(b,3208)
(c,4951)
(d,10090)
(e,26726)
(f,4427)
(g,4825)
(h,12738)
```

```
store group_characters into 'result1';
```

Сохраняет результаты в файл 'result1'.

Листинг примера целиком

```
lines = load 'book.txt' as (line);
words = foreach lines generate flatten(TOKENIZE(line))
as word;
characters = foreach words generate
flatten(TOKENIZE(REPLACE(word, ' ', '|'), '|')) as
character;
filter_characters = filter characters by (character
matches '.*[a-zA-Z].*');
new_characters = foreach filter_characters generate
character;
```

```

group_characters = foreach (group new_characters by
character)      generate      group      as      character,
COUNT(new_characters) as count;
store group_characters into 'result1';

```

В некоторых случаях можно использовать Python для преобразования данных в удобный формат, а затем выполнять их обработку с помощью Pig.

Тот же пример с использованием Python

Python-скрипт:

```
#!/usr/bin/env python3
```

```

@outputSchema("{(character:chararray, count:int)}")
def divide_text(input):
    input = str(input)
    input = input.replace(' ', '').replace(',', '',
    '').replace('.', '').replace('?', '').replace(';',
    '').replace('-', '')
    dictionary = dict()
    for symbol in input:
        if symbol in dictionary:
            dictionary[symbol] += 1
        else:
            dictionary[symbol] = 1
    return dictionary.items()

```

Pig-скрипт:

```

REGISTER 'divider.py' using jython as div;

data = LOAD 'data1.txt' USING PigStorage() AS
(text:chararray);

```

```
characters = FOREACH data GENERATE div.divide_text(text)
AS bag_0:bag{tup:tuple(character:chararray,count:int)};

flat = FOREACH characters GENERATE FLATTEN(bag_0);

sorted_items = ORDER flat BY count DESC;

%declare DT `date +D%y%m%dT%H%M`

STORE sorted_items INTO '1/$$DT/out.txt';
```

ЗАДАНИЕ НА ДОМАШНЮЮ РАБОТУ

Задание 1

Для всех вариантов:

Выполнить задание из лабораторной работы №2, используя язык Pig Latin.

Задание 2

База данных твитов состоит из двух файлов:

Файл tweets.csv имеет формат: tweet_id, tweet, login

Файл users.csv имеет формат: login, user_name, state

Необходимо создать эти файлы и выполнить задание по варианту, используя Pig Latin.

ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ

Скрипт должен выводить данные в папку с номером задания и текущей датой в названии. Для демонстрации возможно использовать режим работы с локальной файловой системой.

ВАРИАНТЫ ЗАДАНИЙ

1. Найти все твиты, содержащие слово “favorite” и подсчитать их количество. Найти все твиты пользователей из штата NY, сгруппировав их по имени пользователя.
2. Найти все твиты пользователей из штата NY, сгруппировав их по имени пользователя. Отсортировать результат по активности пользователя (пользователи с наибольшим числом твитов должны быть вверху списка). Сохранить два файла:
Первый в формате:
user_name, number of tweets
Второй в формате:
user_name, (tweet, tweet ...)
3. Вывести имена пользователей, опубликовавших хотя бы 2 твита. Отсортировать результат по активности пользователя

- (пользователи с наибольшим числом твитов должны быть вверху списка).
4. Вывести имена пользователей, неопубликовавших ни одного твита. Подсчитать в каком штате наибольшее число таких пользователей.
 5. В каждом штате найти пользователя с наибольшим числом твитов. Подсчитать среднее количество твитов для этих пользователей
 6. Выбрать все твиты пользователей из штата NY. Вывести список 20 самых часто используемых слов в их твитах.
 7. Найти все твиты о дожде (должны содержать слова *rain*, *rainy*). Вывести список штатов и количество твитов о дожде пользователей из каждого штата.
 8. Выбрать все твиты пользователей из штата NY и штата MI. Найти список 20 самых часто используемых слов в твитах для обоих штатов. В один файл вывести те слова из списков, которые совпали для обоих штатов, во второй файл вывести несовпавшие слова.
 9. Найти всех пользователей, написавших менее 3 твитов. Подсчитать общее количество твитов, написанных этими пользователями. Вычислить долю, которую составляют эти твиты от общего количества твитов в базе.
 10. Найти все твиты, содержащие слово “*birthday*” и подсчитать их количество. Найти все твиты пользователей из штата MI, сгруппировав их по логину пользователя.
 11. Вывести имена и штаты пользователей, опубликовавших хотя бы 2 твита. Отсортировать результат по убыванию активности пользователя (пользователи с наименьшим числом твитов должны быть вверху списка).
 12. Вывести имена пользователей, опубликовавших только 1 твит. Подсчитать в каком штате наибольшее число таких пользователей.
 13. Выбрать все твиты пользователей из штата AL. Вывести список 15 самых часто используемых слов в их твитах, отсортировав их по убыванию частоты использования.

14. Найти все твиты, содержащие хотя бы одно слово длиной не менее 10 символов. Вывести список штатов и количество таких твитов пользователей из каждого штата.
15. Выбрать все твиты пользователей из штатов NY, MI и FL. Найти список 30 самых часто используемых слов в твитах для данных штатов. Вывести в файл те слова из списков, которые совпали для данных трёх штатов.
16. Найти всех пользователей, написавших ровно 2 твита. Подсчитать общее количество твитов, написанных этими пользователями. Вычислить долю, которую составляют эти твиты от общего количества твитов в базе.
17. В каждом штате найти пользователя, написавшего самый длинный твит. Подсчитать среднюю длину твитов для этих пользователей
18. В каждом штате найти пользователя, написавшего твит с наименьшим числом слов. Подсчитать среднее число слов в твите для этих пользователей
19. Выбрать все твиты пользователей из штата WA. Вывести список 10 самых часто используемых слов в их твитах, отсортировав их по алфавиту.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

- 1) Перечислите основные части, из которых состоит программа, написанная на Pig Latin.
- 2) Дайте определение отношению.
- 3) Дайте определение кортежу.
- 4) Приведите команду для загрузки данные.
- 5) Приведите команду для обращения к полям отношений.
- 6) Перечислите основные типы данных Pig Latin.
- 7) Перечислите основные операции отношений.
- 8) Приведите команду для сохранения полученных данных в файл.
- 9) Приведите команду для вывода на экран полученных данных.
- 10) Приведите метод использования в скрипте функций, написанных на другом языке.

ФОРМА ОТЧЕТА ПО ДОМАШНЕЙ РАБОТЕ

На выполнение домашней работы отводится 2 занятия (4 академических часа: 3 часа на выполнение и сдачу домашней работы и 1 час на подготовку отчета).

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), этапы выполнения работы (со скриншотами), результаты выполнения работы. выводы.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Федин Ф.О. Анализ данных. Часть 1. Подготовка данных к анализу [Электронный ресурс] : учебное пособие / Ф.О. Федин, Ф.Ф. Федин. — Электрон. текстовые данные. — М. : Московский городской педагогический университет, 2012. — 204 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/26444.html>
2. Федин Ф.О. Анализ данных. Часть 2. Инструменты Data Mining [Электронный ресурс] : учебное пособие / Ф.О. Федин, Ф.Ф. Федин. — Электрон. текстовые данные. — М. : Московский городской педагогический университет, 2012. — 308 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/26445.html>
3. Чубукова, И.А. Data Mining [Электронный ресурс] : учеб. пособие — Электрон. дан. — Москва : , 2016. — 470 с. — Режим доступа: <https://e.lanbook.com/book/100582>. — Загл. с экрана.
4. Воронова Л.И. Big Data. Методы и средства анализа [Электронный ресурс] : учебное пособие / Л.И. Воронова, В.И. Воронов. — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2016. — 33 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/61463.html>
5. Юре, Л. Анализ больших наборов данных [Электронный ресурс] / Л. Юре, Р. Ананд, Д.У. Джеффри. — Электрон. дан. — Москва : ДМК Пресс, 2016. — 498 с. — Режим доступа: <https://e.lanbook.com/book/93571>. — Загл. с экрана.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

6. Волкова Т.В. Разработка систем распределенной обработки данных [Электронный ресурс] : учебно-методическое пособие / Т.В. Волкова, Л.Ф. Насейкина. — Электрон. текстовые данные. — Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2012. — 330 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/30127.html>

7. Кухаренко Б.Г. Интеллектуальные системы и технологии [Электронный ресурс] : учебное пособие / Б.Г. Кухаренко. — Электрон. текстовые данные. — М. : Московская государственная академия водного транспорта, 2015. — 116 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/47933.html>
8. Воронова Л.И. Интеллектуальные базы данных [Электронный ресурс] : учебное пособие / Л.И. Воронова. — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2013. — 35 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/63324.html>
9. Николаев Е.И. Базы данных в высокопроизводительных информационных системах [Электронный ресурс] : учебное пособие / Е.И. Николаев. — Электрон. текстовые данные. — Ставрополь: Северо-Кавказский федеральный университет, 2016. — 163 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/69375.html>

Электронные ресурсы:

10. <https://pig.apache.org/> (англ.)
11. https://www.tutorialspoint.com/apache_pig/index.htm (англ.)
12. <http://hadoop.apache.org/> (англ.)