* A = LOAD '~/Desktop/student' USING PigStorage(',') AS (name, age, gpa);
* X = FOREACH A GENERATE name,$2;
* DUMP X;
* STORE X INTO '/app/student' USING PigStorage('$');

### DISTINCT

Removes duplicate tuples in a relation.

Suppose we have relation A.

A = LOAD '/home/ciber/Desktop/abc' using PigStorage(',') AS (a1:int,a2:int,a3:int);

DUMP A;

(8,3,4)

(1,2,3)

(4,3,3)

(4,3,3)

(1,2,3)

In this example all duplicate tuples are removed.

X = DISTINCT A;

DUMP X;

(1,2,3)

(4,3,3)

(8,3,4)

### FILTER

Selects tuples from a relation based on some condition.

Suppose we have relation A.

A = LOAD '/home/ciber/Desktop/a' using PigStorage(',') AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)

(4,2,1)

(8,3,4)

(4,3,3)

(7,2,5)

(8,4,3)

Condition states that if the third field equals 3, then include the tuple with relation X.

X = FILTER A BY a3 == 3;

DUMP X;

(1,2,3)

(4,3,3)

(8,4,3)

In this example the condition states that if the first field equals 8 or if the sum of fields f2 and f3 is not greater than first field, then include the tuple relation X.

X = FILTER A BY (a1 == 8) OR (NOT (a2+a3 > a1));

DUMP X;

(4,2,1)

(8,3,4)

(7,2,5)

(8,4,3)

### 

### FOREACH

Generates data transformations based on columns of data.

X = FOREACH A GENERATE \*;

DUMP X;

(1,2,3)

(4,2,1)

(8,3,4)

(4,3,3)

(7,2,5)

(8,4,3)

In this example two fields from relation A are projected to form relation X.

X = FOREACH A GENERATE a1, a2;

DUMP X;

(1,2)

(4,2)

(8,3)

(4,3)

(7,2)

(8,4)

**Group**

Suppose we have relation A.

A = load '/home/ciber/Desktop/student' using PigStorage(',') AS (name:chararray,age:int,gpa:float);

DESCRIBE A;

A: {name: chararray,age: int,gpa: float}

DUMP A;

(John,18,4.0F)

(Mary,19,3.8F)

(Bill,20,3.9F)

(Joe,18,3.8F)

B = GROUP A BY age;

DUMP B;

(18,{(John,18,4.0F),(Joe,18,3.8F)})

(19,{(Mary,19,3.8F)})

(20,{(Bill,20,3.9F)})

**Join**

Suppose we have relations A and B.

A = LOAD 'data1' AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)

(4,2,1)

(8,3,4)

(4,3,3)

(7,2,5)

(8,4,3)

B = LOAD 'data2' AS (b1:int,b2:int);

DUMP B;

(2,4)

(8,9)

(1,3)

(2,7)

(2,9)

(4,6)

(4,9)

In this example relations A and B are joined by their first fields.

X = JOIN A BY a1, B BY b1;

DUMP X;

(1,2,3,1,3)

(4,2,1,4,6)

(4,3,3,4,6)

(4,2,1,4,9)

(4,3,3,4,9)

(8,3,4,8,9)

(8,4,3,8,9)

**Limit**

A = LOAD 'data' AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)

(4,2,1)

(8,3,4)

(4,3,3)

(7,2,5)

(8,4,3)

In this example output is limited to 3 tuples. Note that there is no guarantee which three tuples will be output.

X = LIMIT A 3;

DUMP X;

(1,2,3)

(4,3,3)

(7,2,5)

**Order**

A = load '/home/ciber/Desktop/kumar\_twitter' using PigStorage(',') AS (name:chararray,age:int,gpa:float);

In this example relation A is sorted by the third field, f3 in descending order. Note that the order of the three tuples ending in 3 can vary.

X = ORDER A BY age DESC;

DUMP X;

(gopal,45,3.9)

(gopal,45,3.9)

(gopal,45,3.9)

(ram,39,3.0)

(ram,39,3.0)

(ram,39,3.0)

(hari,34,3.9)

(hari,34,3.9)

(hari,34,3.9)

(rakesh,28,4.0)

(rakesh,28,4.0)

(rakesh,28,4.0)

**Split**

A = LOAD '/home/ciber/Desktop/a' AS (f1:int,f2:int,f3:int);

DUMP A;

(1,2,3)

(4,5,6)

(7,8,9)

SPLIT A INTO X IF f1<7, Y IF f2==5, Z IF (f3<6 OR f3>6);

DUMP X;

(1,2,3)

(4,5,6)

DUMP Y;

(4,5,6)

DUMP Z;

(1,2,3)

(7,8,9)

**Store**

A = LOAD 'data' AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)

(4,2,1)

(8,3,4)

(4,3,3)

(7,2,5)

(8,4,3)

STORE A INTO 'myoutput' USING PigStorage ('\*');

CAT myoutput;

1\*2\*3

4\*2\*1

8\*3\*4

4\*3\*3

7\*2\*5

8\*4\*3

**Union**

A = LOAD 'data' AS (a1:int,a2:int,a3:int);

DUMP A;

(1,2,3)

(4,2,1)

B = LOAD 'data' AS (b1:int,b2:int);

DUMP A;

(2,4)

(8,9)

(1,3)

X = UNION A, B;

DUMP X;

(1,2,3)

(4,2,1)

(2,4)

(8,9)

(1,3)

**Small exercise:**

Download (userid-track.tsv) file from <http://bit.ly/bsVBK3> publicly available for download.

Perform the following exercises for finding number of tracks per Artist

--Load input files:

Songs – load ‘/user/userid-track.tsv’ as (userId, timestamp, mbArtistId, artistName, mbTrackID, trackName);

--Remain only artist and track names

Projected = foreach Songs generate artistName, trackName;

-- Find unique records

Unique = DISTINCT Projected;

-- Group records by artist:

Grouped = GROUP Unique BY artistName;

-- Count the number of songs per artist

Counted = FOREACH Grouped GENERATE group, COUNT(Unique) AS cnt;

DUMP Countd;

**Error Handling**

Pig returns different code upon completion for these scenarios:

1. Return code 0: All jobs succeeded

2. Return code 1: *Used for retrievable errors*

3. Return code 2: All jobs have failed

4. Return code 3: Some jobs have failed

Writing UDF to use in PIG

import java.io.IOException;

import org.apache.pig.EvalFunc;

import org.apache.pig.data.Tuple;

import org.apache.pig.impl.util.WrappedIOException;

public class UPPER extends EvalFunc<String>

{

public String exec(Tuple input) throws IOException {

if (input == null || input.size() == 0)

return null;

try{

String str = (String)input.get(0);

return str.toUpperCase();

}catch(Exception e){

throw WrappedIOException.wrap("Caught exception processing input row ", e);

}

}

}

Create different udfs and export to a jar file: pig\_demo.jar

A small project

-- Register the tutorial JAR file so that the included UDFs can be called in the script.

REGISTER /home/ciber/Desktop/pig\_demo.jar;

-- Use the PigStorage function to load the excite log file into

-- the “raw” bag as an array of records with the fields **user**, **time**, and **query**.

raw = LOAD '/home/ciber/Desktop/excite.log' USING PigStorage('\t') AS (user:chararray, time:chararray, query:chararray);

-- Call the NonURLDetector UDF to remove records if the query field is empty or a URL.

clean1 = FILTER raw BY NonURLDetector(query);

-- Call the ToLower UDF to change the query field to lowercase.

clean2 = FOREACH clean1 GENERATE user, time,ToLower(query) as query;

-- The excite query log timestamp format is YYMMDDHHMMSS. Since we are only interested in the – hour, Call the ExtractHour UDF to extract the hour (HH) from the time field.

houred = FOREACH clean2 GENERATE user,ExtractHour(time) as hour, query;

-- Use the GROUP operator to group records by query and hour.

hour\_frequency1 = GROUP houred BY (query, hour);

-- Use the COUNT function to get the count (occurrences) of each query.

hour\_frequency2 = FOREACH hour\_frequency1 GENERATE flatten($0), COUNT($1)

as count;

-- Use the FOREACH and GENERATE operators to assign names to the fields.

uniq\_frequency = FOREACH hour\_frequency2 GENERATE $1 as hour, $0 as search,$2 as count;

-- Use the FILTER operator to move all records with a score less than or equal to 3.

filtered\_uniq\_frequency = FILTER uniq\_frequency BY count > 3;

-- Use the ORDER operator to sort the remaining records by hour and count.

ordered\_uniq\_frequency = ORDER filtered\_uniq\_frequency BY hour, count;

-- Use the PigStorage function to store the results. The output file contains a list of search

-- with the following fields: **hour**, **ngram**, **score**, **count**, **mean**.

STORE ordered\_uniq\_frequency INTO '/home/ciber/Desktop/excite\_out' USING PigStorage('#');

These commands can be executed by making as script file with the above commands.

bin/pig -X local -F /home/ciber/Desktop/execute.pig

UDFS

import java.io.IOException;

import org.apache.pig.EvalFunc;

import org.apache.pig.data.Tuple;

public class ToLower extends EvalFunc<String> {

public String exec(Tuple input) throws IOException {

if(input == null || input.size() == 0)

return null;

try{

String query = (String)input.get(0);

return query.toLowerCase().trim();

}catch(Exception e){

System.err.println("ToLower: failed to process input; error - " + e.getMessage());

return null;

}

}

}

import java.io.IOException;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import org.apache.pig.FilterFunc;

import org.apache.pig.data.Tuple;

/\*\*

\* This function removes search queries that are URLs (as defined by \_urlPattern).

\* This function also removes empty queries.

\*/

public class NonURLDetector extends FilterFunc {

private Pattern \_urlPattern = Pattern.compile("^[\"]?(http[:|;])|(https[:|;])|(www\\.)");

public Boolean exec(Tuple arg0) throws IOException {

if (arg0 == null || arg0.size() == 0)

return false;

String query;

try{

query = (String)arg0.get(0);

if(query == null)

return false;

query = query.trim();

}catch(Exception e){

System.err.println("NonURLDetector: failed to process input; error - " + e.getMessage());

return false;

}

if (query.equals("")) {

return false;

}

Matcher m = \_urlPattern.matcher(query);

if (m.find()) {

return false;

}

return true;

}

}

import java.io.IOException;

import org.apache.pig.EvalFunc;

import org.apache.pig.data.Tuple;

/\*\*

\* The excite query log timestamp format is YYMMDDHHMMSS

\* This function extracts the hour, HH

\*/

public class ExtractHour extends EvalFunc<String> {

public String exec(Tuple input) throws IOException {

if (input == null || input.size() == 0)

return null;

String timestamp = (String)input.get(0);

return timestamp.substring(6, 8);

}

}