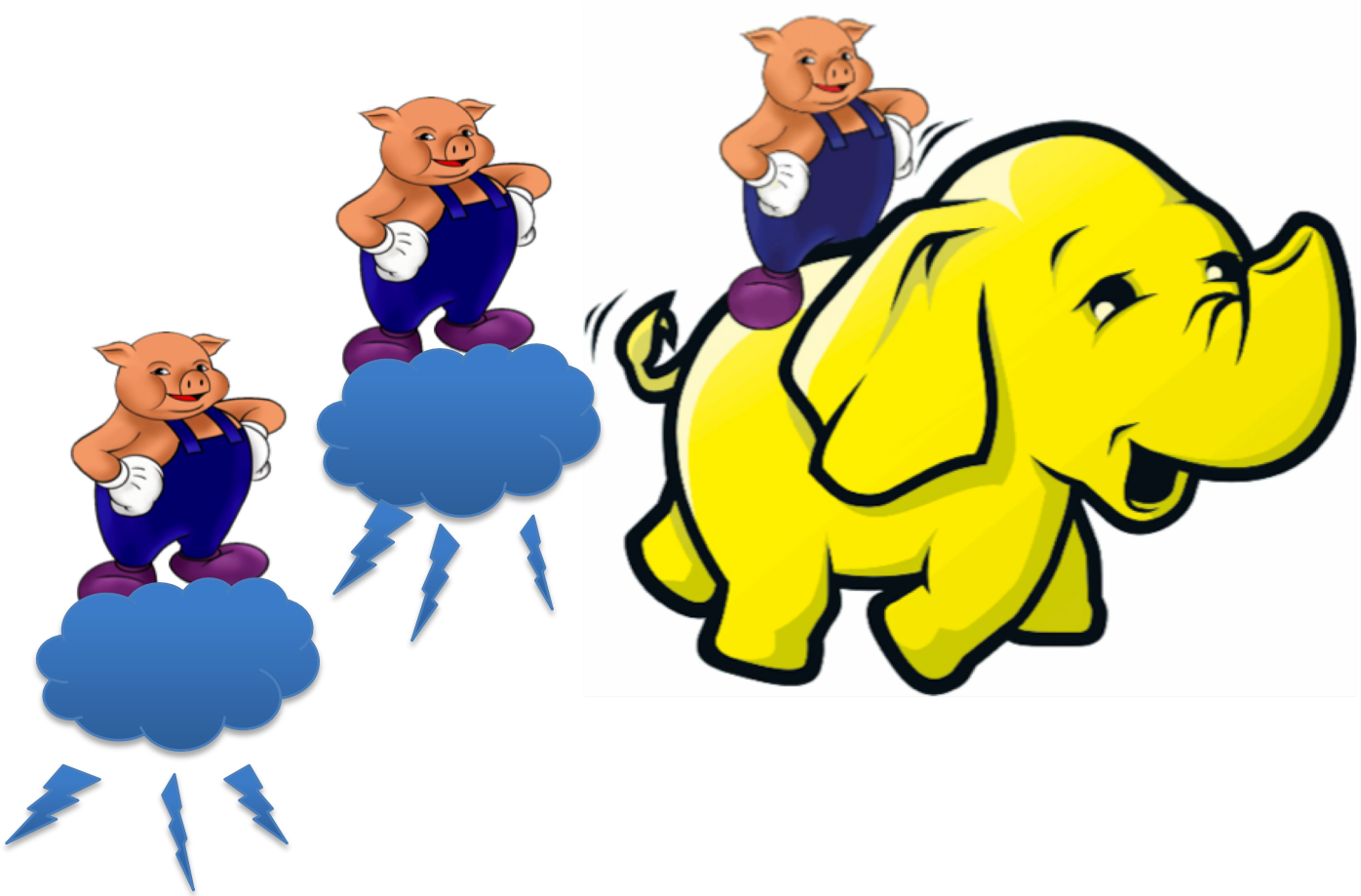
Pig Latin: Advanced Topics



Advanced Relational Operations

- Advanced features of foreach
  - The flatten modifier
  - Nested foreach
- Different Join implementations
- Cogroup
- Union
- Cross

The flatten Modifier

- **Flatten**: Remove the level of the data that are in the types bag or a tuple. For bags, flatten produces a cross product of every item in the bag with all of the other expressions in the generate clause.

```
players = load '/baseball.txt' as
(name:chararray, team:chararray,
position:bag{t:(p:chararray)},
bat:map[]);
pos = foreach players generate name,
flatten(position) as position;
limit10 = limit pos 10;
dump limit10;
```

```
(Jorge Posada,Catcher)
(Jorge Posada,Designated_hitter)
(Landon Powell,Catcher)
(Landon Powell,First_baseman)
(Martín Prado,Second_baseman)
(Martín Prado,Infielder)
(Martín Prado,Left_fielder)
(David Price,Starting_pitcher)
(David Price,Pitcher)
(Jason Pridie,Outfielder)
```

```
Jorge Posada    New York Yankees        {(Catcher),
(Designated_hitter)}
[games#1594,hit_by_pitch#65,on_base_percentage#0.379,grand_slams
#7,home_runs#243,at_bats#5365,sacrifice_flies#43,gdb#163,sacrifi
ce_hits#1,ibbs#71,base_on_balls#838,hits#1488,rbis#964,slugging_
percentage#0.48,batting_average#0.277,triples#9,doubles#342,stri
keouts#1278,runs#817]
Landon Powell   Oakland Athletics      {(Catcher),
(First_baseman)}
[on_base_percentage#0.297,games#46,grand_slams#1,home_runs#7,at_
```

The flatten Modifier (cont'd)

- Group by position after the flatten:

```
players = load '/baseball.txt' as (name:chararray, team:chararray,
position:bag{t:(p:chararray)}, bat:map[]);
pos = foreach players generate name, flatten(position) as position;
pstn_group = group pos by position;
dump pstn_group;
```

```
(Catcher,{(Josh Bard,Catcher),(Ryan Doumit,Catcher),(Bobby Wilson,Catcher),(Brian Schneider,Cat
l Olivo,Catcher),(Ryan Budde,Catcher),(John Buck,Catcher),(Dusty Brown,Catcher),(Jeff Mathis,Ca
(Taylor Teagarden,Catcher),(A.J. Ellis,Catcher),(Brian McCann,Catcher),(Kelly Shoppach,Catcher)
atum,Catcher),(John Baker,Catcher),(José Molina,Catcher),(Brett Hayes,Catcher),(Paul Bako,Catch
 Jaso,Catcher),(George Kottaras,Catcher),(Ryan Hanigan,Catcher),(Miguel Montero,Catcher),(Ronny
atcher),(Jason Jaramillo,Catcher),(Rob Johnson,Catcher),(Chris Coste,Catcher),(Brayan Peña,Catc
Montz,Catcher),(Kurt Suzuki,Catcher),(Jose Lobaton,Catcher),(Humberto Quintero,Catcher),(Rob Bo
(Jorge Posada,Catcher),(Ramón Hernández,Catcher),(José Morales,Catcher),(Jeff Clement,Catcher),
aird,Catcher),(Matt Treanor,Catcher),(Landon Powell,Catcher),(Brandon Inge,Catcher),(Max Ramíre
),(Robinson Cancel,Catcher),(A. J. Pierzynski,Catcher),(J. R. Towles,Catcher),(Yorvit Torrealba
(Wyatt Toregas,Catcher),(Chris Iannetta,Catcher),(Carlos Ruiz,Catcher),(Danny Ardoin,Catcher),(
Catcher),(Dusty Ryan,Catcher),(Mike Rivera,Catcher),(Lou Marson,Catcher),(Ryan Garko,Catcher),(
oli,Catcher),(Jason LaRue,Catcher),(Mike Redmond,Catcher),(Francisco Cervelli,Catcher),(Jason V
cher),(Gregg Zaun,Catcher),(David Ross,Catcher),(Nicky Hundley,Catcher),(Geovany Soto,Catcher),
Jason Kendall,Catcher),(Clint Sammons,Catcher),(Wil Nieves,Catcher),(Brad Ausmus,Catcher),(Mark
Catcher),(Chris Snyder,Catcher),(Dioner Navarro,Catcher),(Wilkin Castillo,Catcher),(Jesús Flore
```

Nested foreach

- A **nested foreach** (or **inner foreach):** applying a set of relational operations to each record in the data pipeline in a foreach statement.

  For example, find the number of unique entries in a group.

```
daily = load '/NYSE_daily.txt' as (exchange, symbol);
grpd = group daily by exchange;
uniqcnt = foreach grpd {
                sym = daily.symbol;
                uniq_sym = distinct sym;
                generate group, COUNT(uniq_sym);
};
dump uniqcnt;
```

Nesting operators inside this foreach.

```
The Job job_1428719433416_0025 has been started successfully.
You can always go back to Query History for results after the run.

(NYSE,237)
```

Nested foreach (cont'd)

- We can modify the code slightly to find the number of entries of each stock symbol.

```
daily = load '/NYSE_daily.txt' as (exchange, symbol);
grpd = group daily by symbol;
symbolcnt = foreach grpd {
        sym = daily.symbol;
        generate group, COUNT(sym);
};
dump symbolcnt;
```

```
The Job job_1428719433416_0031 has been started successfully.
You can always go back to Query History for results after the run.

(CA,257)
(CB,252)
(CE,252)
(CF,254)
(CI,256)
(CL,253)
(CM,252)
(CN,52)
(CO,252)
    ⋮
```

Nested foreach (cont'd)

- Find the top 5 dividends of each stock symbol in a group.

```
divs = load '/NYSE_dividends.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, dividends:float);
grpd = group divs by symbol;
top5 = foreach grpd {
        sorted = order divs by dividends desc;
        tops = limit sorted 5;
        generate group, tops.dividends;
};
dump top5;
```

The Job job_1428719433416_0045 has been started successfully.
You can always go back to Query History for results after the run.

```
(CA,{(0.04),(0.04),(0.04),(0.04)})
(CAE,{(0.029),(0.028),(0.027),(0.023)})
(CAG,{(0.2),(0.19),(0.19),(0.19)})
(CAH,{(9.75),(0.175),(0.175),(0.175),(0.14)})
(CAS,{(0.06)})
(CASC,{(0.05),(0.05),(0.01),(0.01)})
(CAT,{(0.42),(0.42),(0.42),(0.42)})
(CATO,{(0.165),(0.165),(0.165),(0.165)})
(CB,{(0.35),(0.35),(0.35),(0.35)})
(CBC,{(0.05)})
(CBD,{(0.247),(0.148),(0.077)})
```

⋮

More JOIN Implementations

- Joining small to large data
- Joining skewed data
- Joining sorted data

Joining Small to Large Data

- When joining a small file with a big file, it makes sense to send the small file to every machine (node), load it into memory, and then do the join by streaming through the large file and looking up each record in the small file. This is called a fragment-replicate join (because you fragment one file and replicate the other). It supports only inner and left outer joins.

```
daily = load '/NYSE_daily.txt' as (exchange:chararray,
    symbol:chararray, date:chararray, open:float, high:float,
    low:float, close:float, volume:int, adj_close:float);
divs = load '/NYSE_dividends.txt' as (exchange:chararray,
    symbol:chararray, date:chararray, dividends:float);
jnd = join daily by (exchange, symbol), divs by (exchange, symbol)
    using 'replicated';
```

- The *using 'replicated'* keyword tells Pig to use the fragment-replicate algorithm. The 2nd input listed in the join is always the input that is loaded into memory. If Pig cannot fit the replicated input into memory, it will issue an error and fail.

Joining Skewed Data

- Sometimes the data you will be processing with Pig has significant skew in the number of records per key. Pig's default join algorithm is very sensitive to skew, because it collects all of the records for a given key together on a single reducer. In some datasets, there are a few keys that have far more records than other keys. This results in a few reducers that will take much longer than the rest. To deal with this, Pig provides **skew join**.

```
users = load 'users' as (name:chararray, city:chararray);
cinfo = load 'cityinfo' as (city:chararray, population:int);
jnd = join cinfo by city, users by city using 'skewed';
```

- The second input in the join, in this case users, is the one that will be sampled and have its keys with a large number of values split across reducers. The first input will have records with those values replicated across reducers.

Join Sorted Data

- If your inputs are already sorted on the join key, the join can be done efficiently in the map phase by opening both files and walking through them. Pig refers to this as a **merge join.** This can be done without a reduce phase, and so it is more efficient than a default join. For example,

```
daily = load '/NYSE_daily.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, open:float, high:float,
        low:float, close:float, volume:int, adj_close:float);
srtd  = order daily by symbol;
divs  = load '/NYSE_dividends.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, dividends:float);
dsrtd = order divs by symbol;
jnd = join srtd by symbol, dsrtd by symbol using 'merge';
dump jnd;
```

Join Sorted Data (cont'd)

- The result:

The Job job_1428719433416_0076 has been started successfully.
You can always go back to Query History for results after the run.

```
(NYSE,CA,2009-12-09,22.5,22.88,22.36,22.84,7173000,22.8,NYSE,CA,2009-08-06,0.04)
(NYSE,CA,2009-12-08,21.96,22.22,21.78,21.91,3423200,21.87,NYSE,CA,2009-08-06,0.04)
(NYSE,CA,2009-12-07,22.43,22.65,22.23,22.34,3874200,22.3,NYSE,CA,2009-08-06,0.04)
(NYSE,CA,2009-12-04,22.56,22.66,22.25,22.65,4850000,22.61,NYSE,CA,2009-08-06,0.04)
(NYSE,CA,2009-12-03,22.68,22.74,22.24,22.29,3494500,22.25,NYSE,CA,2009-08-06,0.04)
(NYSE,CA,2009-12-02,22.37,22.64,22.16,22.42,2571300,22.38,NYSE,CA,2009-08-06,0.04)
(NYSE,CA,2009-12-01,22.26,22.53,22.21,22.48,3468900,22.44,NYSE,CA,2009-08-06,0.04)
(NYSE,CA,2009-11-30,21.9,22.13,21.73,22.1,4271500,22.06,NYSE,CA,2009-08-06,0.04)
(NYSE,CA,2009-11-27,21.64,22.26,21.56,22.0,1973400,21.96,NYSE,CA,2009-08-06,0.04)
```

Cogroup

- The operation **cogroup** is a generalization of **group**. Instead of collecting records of one input, it collects records of *n* inputs based on a key. The result is a record with a key and one bag for each input. Each bag contains all records from that input that have the given value for the key. All records with a null value in the key will be collected together.

```
daily = load '/NYSE_daily.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, open:float, high:float,
        low:float, close:float, volume:int, adj_close:float);
divs = load '/NYSE_dividends.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, dividends:float);
grpd = cogroup daily by (exchange, symbol), divs by (exchange,
        symbol);
dump grpd;
```

Cogroup (cont'd)

- The result:

```
((NYSE,CA),{(NYSE,CA,2009-11-27,21.64,22.26,21.56,22.0,1973400,21.96),(N
YSE,CA,2009-12-01,22.26,22.53,22.21,22.48,3468900,22.44),(NYSE,CA,2009-1
2-02,22.37,22.64,22.16,22.42,2571300,22.38),(NYSE,CA,2009-12-03,22.68,2
2.74,22.24,22.29,3494500,22.25),(NYSE,CA,2009-12-04,22.56,22.66,22.25,2
2.65,4850000,22.61),(NYSE,CA,2009-12-07,22.43,22.65,22.23,22.34,387420
0,22.3),(NYSE,CA,2009-12-08,21.96,22.22,21.78,21.91,3423200,21.87),(NYS
E,CA,2009-12-09,22.5,22.88,22.36,22.84,7173000,22.8),(NYSE,CA,2009-12-1
```

⋮

```
E,CA,2009-11-13,0.04),(NYSE,CA,2009-02-12,0.04),(NYSE,CA,2009-05-27,0.0
4)})
((NYSE,CB),{(NYSE,CB,2009-05-07,40.25,41.0,39.32,39.52,7247400,38.61),(N
YSE,CB,2009-05-08,40.07,41.01,39.48,40.88,5531200,39.94),(NYSE,CB,2009-0
5-11,39.9,40.77,39.35,39.51,5253400,38.6),(NYSE,CB,2009-05-12,40.15,40.1
6,39.02,39.21,4815900,38.31),(NYSE,CB,2009-05-13,38.71,40.09,38.52,3
8.9,4532700,38.0),(NYSE,CB,2009-05-14,38.94,40.34,38.82,40.18,4601200,3
9.25),(NYSE,CB,2009-05-15,40.1,40.1,38.81,38.94,3882000,38.04),(NYSE,C
B,2009-05-18,39.66,40.5,38.5,40.37,5449600,39.44),(NYSE,CB,2009-05-19,4
```

⋮

Use cogroup with foreach

- When cogroup is used with foreach, where each bag is flattened, it is equivalent to a join—as long as there are no null values in the keys.

```
daily = load '/NYSE_daily.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, open:float, high:float,
        low:float, close:float, volume:int, adj_close:float);
divs = load '/NYSE_dividends.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, dividends:float);
grpd = cogroup daily by (exchange, symbol), divs by (exchange,
        symbol);
sjnd = filter grpd by not IsEmpty(divs);
final = foreach sjnd generate flatten(daily);
dump final;
```

Use cogroup with foreach (cont'd)

- The result:

The Job job_1428719433416_0085 has been started successfully.
You can always go back to Query History for results after the run.

```
(NYSE,CA,2009-11-27,21.64,22.26,21.56,22.0,1973400,21.96)
(NYSE,CA,2009-12-01,22.26,22.53,22.21,22.48,3468900,22.44)
(NYSE,CA,2009-12-02,22.37,22.64,22.16,22.42,2571300,22.38)
(NYSE,CA,2009-12-03,22.68,22.74,22.24,22.29,3494500,22.25)
(NYSE,CA,2009-12-04,22.56,22.66,22.25,22.65,4850000,22.61)
(NYSE,CA,2009-12-07,22.43,22.65,22.23,22.34,3874200,22.3)
(NYSE,CA,2009-12-08,21.96,22.22,21.78,21.91,3423200,21.87)
(NYSE,CA,2009-12-09,22.5,22.88,22.36,22.84,7173000,22.8)
(NYSE,CA,2009-12-10,22.84,23.0,22.64,22.72,3484100,22.68)
(NYSE,CA,2009-12-11,22.8,22.9,22.67,22.83,3633400,22.79)
(NYSE,CA,2009-12-14,23.0,23.25,22.92,22.97,6665000,22.93)
(NYSE,CA,2009-12-15,22.75,22.99,22.51,22.51,13518100,22.47)
(NYSE,CA,2009-12-16,22.62,22.86,22.47,22.49,8536700,22.45)
(NYSE,CA,2009-12-17,22.86,22.86,22.41,22.41,7817700,22.37)
(NYSE,CA,2009-12-18,22.55,22.75,22.43,22.51,7590500,22.47)
(NYSE,CA,2009-12-21,22.79,22.87,22.63,22.65,4099800,22.61)
(NYSE,CA,2009-12-22,22.67,22.79,22.59,22.6,4286800,22.56)
```

⋮

Union

- **Union**: put two data sets together by concatenating them. It does **not** perform a mathematical set union, so duplicate records are not eliminated.
- Unlike union in SQL, Pig does not require that both inputs share the same schema.
  - Both share the same schema → that schema.

```
A = load 'input1' as (x:int, y:float);
B = load 'input2' as (x:int, y:float);
C = union A, B;
describe C;
C: {x: int, y: float}
```

  - One schema can be produced from another by implicit casts → that resulting schema.

```
A = load 'input1' as (x:int, y:float);
B = load 'input2' as (x:int, y:double);
C = union A, B;
describe C;
C: {x: int, y: double}
```

Union (cont'd)

- Otherwise → no schema (i.e., different records have different fields).

```
A = load 'input1' as (x:int, y:float);
B = load 'input2' as (x:int, y:chararray);
C = union A, B;
describe C;
```

Schema for C unknown.

Note: The schema comparison includes names, so different field names will result in the output having no schema.

Cross

- The **cross** operation takes every record in one relation and combines it with every record in another relation and will produce output with n x m records. Pig does implement cross in a parallel fashion.

```
daily = load '/NYSE_daily.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, open:float, high:float,
        low:float, close:float, volume:int, adj_close:float);
divs = load '/NYSE_dividends.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, dividends:float);
crosseddata = cross daily, divs parallel 10;
toshow = limit crosseddata 10;
dump toshow;
```

(Note: If not limited, this code takes a LONG time to run locally due to the n x m records to create and output.)

Cross (cont'd)

- The result:

The Job job_1428719433416_0114 has been started successfully.
You can always go back to Query History for results after the run.

```
(NYSE,CLI,2009-12-24,35.38,35.6,35.19,35.47,230200,35.01,NYSE,CPO,2009-09-28,0.1
4)
(NYSE,CLI,2009-12-24,35.38,35.6,35.19,35.47,230200,35.01,NYSE,CPO,2009-12-30,0.1
4)
(NYSE,CLI,2009-12-28,35.67,36.23,35.49,35.69,565000,35.23,NYSE,CPO,2009-01-06,0.1
4)
(NYSE,CLI,2009-12-28,35.67,36.23,35.49,35.69,565000,35.23,NYSE,CPO,2009-06-26,0.1
4)
(NYSE,CLI,2009-12-28,35.67,36.23,35.49,35.69,565000,35.23,NYSE,CPO,2009-09-28,0.1
4)
```

Parameter Substitution

- Parameter substitution provides a capability with a basic string-replacement functionality in Pig Latin. The following script is saved in a file named dly.pig (in HDFS).

```
daily = load '/NYSE_daily.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, open:float, high:float,
        low:float, close:float, volume:int, adj_close:float);
yesterday = filter daily by date == '$DATE';
grpd = group yesterday all;
minmax = foreach grpd generate MAX(yesterday.high),
        MIN(yesterday.low);
dump minmax;
```

- When you run dly.pig, you need to provide a definition for the parameter DATE as follows:

```
pig -param DATE=2009-12-30 dly.pig
```

Parameter Substitution: The Result

```
Successfully read 57391 records (3194476 bytes) from: "/NYSE_daily.txt"

Output(s):
Successfully stored 1 records (14 bytes) in: "hdfs://sandbox.hortonworks.com:8020/tmp/temp822105052/tmp-1224201560"

Counters:
Total records written : 1
Total bytes written : 14
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1429026430369_0014


2015-04-14 17:54:11,436 [main] INFO  org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service ad
//sandbox.hortonworks.com:8188/ws/v1/timeline/
2015-04-14 17:54:11,438 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at sandbo
s.com/192.168.56.101:8050
2015-04-14 17:54:11,444 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed
cationStatus=SUCCEEDED. Redirecting to job history server
2015-04-14 17:54:11,540 [main] INFO  org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service ad
//sandbox.hortonworks.com:8188/ws/v1/timeline/
2015-04-14 17:54:11,541 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at sandbo
s.com/192.168.56.101:8050
2015-04-14 17:54:11,546 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed
cationStatus=SUCCEEDED. Redirecting to job history server
2015-04-14 17:54:11,673 [main] INFO  org.apache.hadoop.yarn.client.api.impl.TimelineClientImpl - Timeline service ad
//sandbox.hortonworks.com:8188/ws/v1/timeline/
2015-04-14 17:54:11,678 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at sandbo
s.com/192.168.56.101:8050
2015-04-14 17:54:11,682 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed
cationStatus=SUCCEEDED. Redirecting to job history server
2015-04-14 17:54:11,716 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher
2015-04-14 17:54:11,719 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... w
rate code.
2015-04-14 17:54:11,726 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to pr
2015-04-14 17:54:11,727 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input pat
s : 1
(339.74,0.2)
2015-04-14 17:54:11,764 [main] INFO  org.apache.pig.Main - Pig script completed in 47 seconds and 836 milliseconds (
[root@sandbox ~]#
```

Pig Macro

- **Macros** are declared with the **define** statement.
- A macro takes a set of input parameters, which are string values that will be substituted for the parameters when the macro is expanded.
- By convention, input relation names are placed first before other parameters.
- The output relation name is given in a returns statement.
- The parameters, including the output relation name, are referenced inside the macro, and they must be preceded by a $ (dollar sign).
- The macro is then invoked in your Pig Latin by assigning it to a relation.

A Pig Macro Example

Title: macro.pig

Pig script: PIG helper

```
1  define dividend_analysis (daily, year, daily_symbol, daily_date,
2      daily_open, daily_close)
3  returns analyzed {
4      divs = load '/NYSE_dividends.txt' as (exchange:chararray,
5          symbol:chararray, date:chararray, dividends:float);
6      divsthisyear = filter divs by date matches '$year-.*';
7      dailythisyear = filter $daily by date matches '$year-.*';
8      jnd = join divsthisyear by (symbol, date),
9          dailythisyear by ($daily_symbol, $daily_date);
10     $analyzed = foreach jnd generate divsthisyear::symbol,
11         divsthisyear::date, $daily_close - $daily_open;
12 };
13
14 daily = load '/NYSE_daily.txt' as (exchange:chararray,
15     symbol:chararray, date:chararray, open:float, high:float,
16     low:float, close:float, volume:int, adj_close:float);
17 results = dividend_analysis(daily, '2009', 'symbol', 'date',
18     'open', 'close');
19 dump results;
20
```

The Result of a Pig Macro

The Job job_1429026430369_0064 has been started successfully.
You can always go back to Query History for results after the run.

```
(CA,2009-02-12,0.6000004)
(CA,2009-05-27,-0.55999947)
(CA,2009-08-06,-0.40999985)
(CA,2009-11-13,0.34000015)
(CB,2009-03-18,3.0600014)
(CB,2009-06-24,0.15999985)
(CB,2009-09-16,0.66999817)
(CB,2009-12-16,0.5)
(CE,2009-01-13,-0.1800003)
(CE,2009-04-13,0.6099987)
(CE,2009-07-13,-0.20999908)
(CE,2009-10-13,-0.049999237)
(CF,2009-02-12,4.419998)
(CF,2009-05-12,0.9700012)
(CF,2009-08-12,1.3700027)
(CF,2009-11-12,1.4300003)
(CI,2009-03-09,0.21000004)
(CL,2009-01-22,0.020000458)
(CL,2009-04-22,-0.83000183)
```

Macro Modification 1: Interested in a Month of a Year

Title: macro.pig

Pig script: PIG helper

```
1  define dividend_analysis (daily, year_month, daily_symbol, daily_date,
2      daily_open, daily_close)
3  returns analyzed {
4      divs = load '/NYSE_dividends.txt' as (exchange:chararray,
5          symbol:chararray, date:chararray, dividends:float);
6      divsthisyear = filter divs by date matches '$year_month.*';
7      dailythisyear = filter $daily by date matches '$year_month.*';
8      jnd = join divsthisyear by (symbol, date),
9          dailythisyear by ($daily_symbol, $daily_date);
10     $analyzed = foreach jnd generate divsthisyear::symbol,
11         divsthisyear::date, $daily_close - $daily_open;
12 };
13
14 daily = load '/NYSE_daily.txt' as (exchange:chararray,
15     symbol:chararray, date:chararray, open:float, high:float,
16     low:float, close:float, volume:int, adj_close:float);
17 results = dividend_analysis(daily, '2009-02', 'symbol', 'date',
18     'open', 'close');
19 dump results;
20
```

Result for 2009-02

The Job job_1429026430369_0072 has been started successfully.
You can always go back to Query History for results after the run.

```
(CA,2009-02-12,0.6000004)
(CF,2009-02-12,4.419998)
(CR,2009-02-25,-0.3800001)
(CBC,2009-02-11,-0.01999998)
(CBE,2009-02-25,-1.2800007)
(CBT,2009-02-25,-0.09999943)
(CCJ,2009-02-27,-0.119999886)
(CCW,2009-02-25,0.12999916)
(CDR,2009-02-06,0.2999997)
(CFR,2009-02-25,-0.11999893)
(CFT,2009-02-02,0.099998474)
(CGO,2009-02-06,0.05000019)
(CHD,2009-02-05,-1.0099983)
(CHE,2009-02-26,-1.8300018)
(CHI,2009-02-06,0.0)
(CHW,2009-02-06,0.05000019)
(CHY,2009-02-06,0.05000019)
```

Macro Modification 2: Another Month

Title: macro.pig

Pig script: PIG helper

```
define dividend_analysis (daily, year_month, daily_symbol, daily_date,
    daily_open, daily_close)
returns analyzed {
    divs = load '/NYSE_dividends.txt' as (exchange:chararray,
        symbol:chararray, date:chararray, dividends:float);
    divsthisyear = filter divs by date matches '$year_month.*';
    dailythisyear = filter $daily by date matches '$year_month.*';
    jnd = join divsthisyear by (symbol, date),
        dailythisyear by ($daily_symbol, $daily_date);
    $analyzed = foreach jnd generate divsthisyear::symbol,
        divsthisyear::date, $daily_close - $daily_open;
};

daily = load '/NYSE_daily.txt' as (exchange:chararray,
    symbol:chararray, date:chararray, open:float, high:float,
    low:float, close:float, volume:int, adj_close:float);
results = dividend_analysis(daily, '2009-08', 'symbol', 'date',
    'open', 'close');
dump results;

```

Result for 2009-08

The Job job_1429026430369_0074 has been started successfully.
You can always go back to Query History for results after the run.

```
(CA,2009-08-06,-0.40999985)
(CF,2009-08-12,1.3700027)
(CR,2009-08-27,0.5100002)
(CBD,2009-08-12,0.3199997)
(CBE,2009-08-27,-0.30999756)
(CBT,2009-08-26,-0.45000076)
(CCW,2009-08-28,0.12999916)
(CDI,2009-08-11,-0.06000042)
(CEL,2009-08-27,-0.15999985)
(CFR,2009-08-28,-0.87999725)
(CGO,2009-08-06,-0.10000038)
(CHD,2009-08-11,0.08000183)
(CHE,2009-08-13,0.07999802)
(CHI,2009-08-06,-0.09999943)
(CHW,2009-08-06,-0.13000011)
(CHY,2009-08-06,-0.25)
(CIF,2009-08-10,-0.01999998)
```