http://pig.apache.org/docs/r0.14.0/basic.html

“使用 Apache Pig 处理数据”:
https://www.ibm.com/developerworks/cn/linux/l-apachepigdataquery/

Pig的几种模式

pig -x local
Tez Local
Mapreduce
Tez

复杂数据类型

Tuple
A tuple is an ordered set of fields.
(field [, field …])

Bag
A bag is a collection of tuples.
{ tuple [, tuple …] }

Map
A map is a set of key/value pairs.
[key#value <, key#value …>]
key必须是唯一的chararray

表达式

*号
表示一个tuple的所有字段

Project-range表达式
.. $x：第0列到第x列
$x ..：第x列到最后一列
$x .. $y：第x列到最后一列

算数运算符

Arithmetic Operators：+ - * / % ?: case
Boolean Operators：and or in not
Cast Operators：(type)field
Comparison Operators：== != < > <= >= matches
Type Construction Operators：() {} []
Dereference Operators：tuple.id bag.id map#'key'
Disambiguate Operator：::
Flatten Operator：flatten
Null Operators：is null / is not null
Sign Operators：+ -

关系运算符

LOAD

语法：LOAD 'data' [USING function] [AS schema];
function：http://pig.apache.org/docs/r0.14.0/func.html#load-store-functions

如果LOAD时候没有指定数据类型，字段会默认被设为bytearray类型，使用时候根据具体类型进行隐式转换。

```
A = LOAD 'data' AS (f1,f2,f3);
B = FOREACH A GENERATE f1 + 5;
```

举例：
load以空格分隔的文件：A = LOAD 'FilePath' using PigStorage(' ') as (col1: int, col2: int);
Pig默认是tab分隔。

复杂数据类型：

```
cat data;
(3,8,9) (4,5,6)
(1,4,7) (3,7,5)

A = LOAD 'data' AS (t1:tuple(t1a:int, t1b:int,t1c:int),t2:tuple(t2a:int,t2b:int,t2c:int));

DUMP A;
((3,8,9),(4,5,6))
((1,4,7),(3,7,5))

X = FOREACH A GENERATE t1.t1a,t2.$0;

DUMP X;
(3,4)
(1,3)
```

ASSERT

语法：ASSERT alias BY expression [, message];

CROSS

Computes the cross product of two or more relations.
语法：alias = CROSS alias, alias [, alias …] [PARTITION BY partitioner] [PARALLEL n];

CUBE

对key的所有可能的组合进行group。
语法：alias = CUBE alias BY { CUBE expression | ROLLUP expression }, [CUBE expression | ROLLUP expression] [PARALLEL n];

举例：
CUBE operation:

```
salesinp = LOAD '/pig/data/salesdata' USING PigStorage(',') AS
        (product:chararray, year:int, region:chararray, state:chararray, city:chararray,
sales:long);
cubedinp = CUBE salesinp BY CUBE(product,year);
result = FOREACH cubedinp GENERATE FLATTEN(group), SUM(cube.sales) AS totalsales;
```

input: (car, 2012, midwest, ohio, columbus, 4000)
output:
(car,2012,4000)
(car,,4000)
(,2012,4000)
(,,4000)

ROLLUP operation:

```
salesinp = LOAD '/pig/data/salesdata' USING PigStorage(',') AS
        (product:chararray, year:int, region:chararray, state:chararray, city:chararray,
sales:long);
rolledup = CUBE salesinp BY ROLLUP(region,state,city);
result = FOREACH rolledup GENERATE FLATTEN(group), SUM(cube.sales) AS totalsales;
```

input: (car, 2012, midwest, ohio, columbus, 4000)
output:
(midwest,ohio,columbus,4000)
(midwest,ohio,,4000)
(midwest,,,4000)
(,,,4000)

DISTINCT
语法：alias = DISTINCT alias [PARTITION BY partitioner] [PARALLEL n];

FILTER
语法：alias = FILTER alias BY expression;

FOREACH
语法：alias = FOREACH { block | nested_block };
Use the FOREACH…GENERATE operation to work with columns of data (if you want to work with tuples or rows of data, use the FILTER operation).

GROUP
语法：alias = GROUP alias { ALL | BY expression} [, alias ALL | BY expression …] [USING 'collected' | 'merge'] [PARTITION BY partitioner] [PARALLEL n];

The GROUP operator groups together tuples that have the same group key (key field). The key field will be a tuple if the group key has more than one field, otherwise it will be the same type as that of the group key. The result of a GROUP operation is a relation that includes one tuple per group. This tuple contains two fields:

- The first field is named "group" (do not confuse this with the GROUP operator) and is the same type as the group key.
- The second field takes the name of the original relation and is type bag.

举例：
cat data:
1 a
1 a
1 c
2 a
2 b
3 c

A = LOAD 'data' AS (id: int, name: chararray);
B = GROUP A by id;
DESCRIBE B;
B: {group: int, A: {id: int, name: chararray}}

DUMP B;
(1, {(1, a), (1, a), (1, c)})
(2, {(2, a), (2, b)})
(3, {(3, c)})

C = FOREACH B GENERATE group,

COGROUP
例子：
X = COGROUP A BY owner, B BY friend2;

DESCRIBE X;
X: {group: chararray,A: {owner: chararray,pet: chararray},B: {friend1: chararray,friend2: chararray}}

JOIN (inner)
语法：alias = JOIN alias BY {expression|'('expression [, expression …]')'} (, alias BY {expression|'('expression [, expression …]')'} …) [USING 'replicated' | 'skewed' | 'merge' | 'merge-sparse'] [PARTITION BY partitioner] [PARALLEL n];

例子：
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)

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)

JOIN (outer)
语法：alias = JOIN left-alias BY left-alias-column [LEFT|RIGHT|FULL] [OUTER], right-alias BY right-alias-column [USING 'replicated' | 'skewed' | 'merge'] [PARTITION BY partitioner] [PARALLEL n];

LIMIT
语法：alias = LIMIT alias n;

MAPREDUCE
Executes native MapReduce jobs inside a Pig script.
语法：alias1 = MAPREDUCE 'mr.jar' STORE alias2 INTO 'inputLocation' USING storeFunc LOAD 'outputLocation' USING loadFunc AS schema [`params, ... `];

ORDER BY
语法：alias = ORDER alias BY { * [ASC|DESC] | field_alias [ASC|DESC] [, field_alias [ASC|DESC] …] } [PARALLEL n];

RANK
语法：alias = RANK alias [BY { * [ASC|DESC] | field_alias [ASC|DESC] [, field_alias [ASC|DESC] …] } [DENSE]];

例子：
A = load 'data' AS (f1:chararray,f2:int,f3:chararray);

```
DUMP A;
(David,1,N)
(Tete,2,N)
(Ranjit,3,M)
(Ranjit,3,P)
(David,4,Q)
(David,4,Q)
(Jillian,8,Q)
(JaePak,7,Q)
(Michael,8,T)
(Jillian,8,Q)
(Jose,10,V)

B = rank A;
dump B;
(1,David,1,N)
(2,Tete,2,N)
(3,Ranjit,3,M)
(4,Ranjit,3,P)
(5,David,4,Q)
(6,David,4,Q)
(7,Jillian,8,Q)
(8,JaePak,7,Q)
(9,Michael,8,T)
(10,Jillian,8,Q)
(11,Jose,10,V)

C = rank A by f1 DESC, f2 ASC;
dump C;
(1,Tete,2,N)
(2,Ranjit,3,M)
(2,Ranjit,3,P)
(4,Michael,8,T)
(5,Jose,10,V)
(6,Jillian,8,Q)
(6,Jillian,8,Q)
(8,JaePak,7,Q)
(9,David,1,N)
(10,David,4,Q)
(10,David,4,Q)

C = rank A by f1 DESC, f2 ASC DENSE;
dump C;
(1,Tete,2,N)
(2,Ranjit,3,M)
(2,Ranjit,3,P)
```

(3,Michael,8,T)
(4,Jose,10,V)
(5,Jillian,8,Q)
(5,Jillian,8,Q)
(6,JaePak,7,Q)
(7,David,1,N)
(8,David,4,Q)
(8,David,4,Q)

SAMPLE
语法：SAMPLE alias size;
size是百分比。

SPLIT
语法：SPLIT alias INTO alias IF expression, alias IF expression [, alias IF expression …] [, alias OTHERWISE];

STORE
语法：STORE alias INTO 'directory' [USING function];

STREAM
Sends data to an external script or program.
语法：alias = STREAM alias [, alias …] THROUGH {`command` | cmd_alias } [AS schema] ;

UNION
语法：alias = UNION [ONSCHEMA] alias, alias [, alias …];

UDF Statements

DEFINE
Assigns an alias to a UDF or streaming command.
语法：DEFINE alias {function | [`command` [input] [output] [ship] [cache] [stderr]] };

REGISTER
Registers a JAR file so that the UDFs in the file can be used.
语法：REGISTER path;

Pig Macros

Define
定义宏：DEFINE alias {function | [`command` [input] [output] [ship] [cache] [stderr]] };
展开宏：alias [, alias ...] = macro_name (param [, param ...]);

Import
Import macros defined in a separate file.
语法：IMPORT 'file-with-macro';

Parameter Substitution

Substitute values for parameters at run time.

命令行中语法：pig {-param param_name = param_value | -param_file file_name} [-debug | -dryrun] script
脚本中语法：{%declare | %default} param_name param_value

举例：
myscript.pig内容如下：
A = LOAD '$data' USING PigStorage() AS (f1:int, f2:int, f3:int);
DUMP A;
$ pig -param data=mydata myscript.pig -- mydata是存放数据的文件名

myparams文件存放变量：
# my parameters
data1 = mydata1
cmd = `generate_name`
$ pig -param_file myparams script2.pig

使用declare声明执行的脚本：
%declare CMD `generate_date`;
A = LOAD '/data/mydata/$CMD';

注释

单行注释：--
多行注释：/* */

Table 10-1. Pig-Python type translations

Pig type	Python type
int	number
long	number
float	number
double	number
chararray	string
bytearray	string
map	dictionary
tuple	tuple
bag	list of tuples

by chenxiaoyu@mobvoi
E-mail: jschenxiaoyu@gmail.com
2016.02.19