# Lesson 1 - Pig Basics

## List the ways to invoke Pig

Pig or Pig Latin as the language is called, is a high-level language designed to remove the complexities of coding MapReduce applications.

Pig converts its operators into MapReduce code. So instead of needing Java programming skills and an understanding of the MapReduce coding infrastructure, people with little programming skills, can simply invoke SORT or FILTER operators without having to code a MapReduce application to accomplish those tasks.

One of the strong features of Pig is that the developer / analyst does not have to depend solely on the Pig-supplied operators. It is possible to extend the language by coding one’s own user-defined functions.

### Interactive mode

To invoke Pig in interactive mode, just execute pig and you are then placed into the Grunt shell. By default you will be running in MapReduce mode.

To execute Pig interactively in local mode, execute

pig –x local

### Script mode

To invoke a Pig script, just include the name of the script when you invoke pig. For example to run a pig script called myscript.pig in local mode, execute

pig –x local myscript.pig

To run the same script in MapReduce mode, execute

pig myscript.pig

### Options

Both of interactive and script ways have two options.

* Local mode
  + gives you access to data in your local file system. It runs in a single Java virtual machine. This mode is normally used for testing against a limited amount of data.
* MapReduce mode
  + runs on a Hadoop cluster and converts the Pig statements into MapReduce code.

It is also possible to invoke pig using a Java command as well.

## Describe the data structures used in Pig

First look at Pig data As you might expect, the Pig language has a number of scalar data types used to define individual fields. Some examples are

* int
  + 32 bit
* long
  + 64 bit
* Float
  + 32 big
* Double
  + 64 bit
* Chararray
  + string data
* bytearray
  + a blob

### Tuples

Fields are grouped together into tuples. You might think of a tuple as a row in a relational data table or a record in a file. Pig has greater flexibility over a relational table in that all tuples do not have to contain the same number of fields.

### Bags

Tuples are grouped into a bag. Now this is where it can get a little complex. I just said that a bag is a group of tuples, but a “field” in a tuple can be a bag that contains other tuples. In this case, the bag is an inner bag.

The bag, that contains the tuples that correlate to a row in a relational table, is known as an outer bag. It is also known as a relation.

### Map

Map data is also supported in Pig. A map is a set of key / value pairs. Accessing a map with a specify key will return the value associated with that key.

## Common operations

The STORE operator takes a relation as input but writes the data in the relation to a file

the DUMP operator, which like the STORE operator, takes a relation as input but writes the data values to the console.

All Pig statements, both in a script and when running in a Grunt shell, are terminated by a semicolon.

### Comments

All statements between a /\* and an \*/ are commented.

Double dashes -- will comment the rest of that line.

## Explain how to use the LOAD operator to read data

Most Pig operators take a relation (an outer bag) as input and in turn emits a relation. The exceptions are the LOAD operator which only emits a relation (after reading data from some source).

The LOAD operator is used to read data from a source and place that data into a relation.

Since there are a variety of ways that data can be stored, there are going to be a variety of functions that can be used with this operator. Pig supplies four such load functions and you can supply your own as well.

### PigStorage

The default load function is PigStorage.

This reads data that is in a delimited format with the default delimiter being the tab character. If some other delimiting character is to be used, then that character is supplied in single quotes.

### TextLoader

The TextLoader reads in a line of text and this line of text is placed into a single tuple.

### BinStorage

The BinStorage function is used to read data in a machine-readable format. This is used by Pig internally to store temporary data.

### JsonLoader

There is also a JsonLoader. This loads data that is in a standard JSON format.

### Example.

A = load ‘/datadir/datafile’ using PigStorage(‘\t’);

will read data from a file, datafile located in the /datadir. Since the PigStorage function is being used, the data is expected to be in a delimited format with the tab character as being the delimiter. The results are placed into a relation call A.

As I said, you can code your own load function if your data is in some proprietary format not supported by the supplied load functions. There is a LOAD option that we should discuss. This option provides a schema for you input data. To specify a schema, you code as (your schema info) after the load function. So, continuing with a similar example,

A = load ‘/datadir/datafile’ using PigStorage(‘,’) as (f1:int, f2:chararray, f3:float);

would read a comma delimited file and interpret the first field as an integer and associate it with a name of f1. The second field would be a chararray and the third a float. If you do not supply a schema for the PigStorage function, then the fields are not named and all fields default to bytearray for their data type. The JsonLoader function requires a schema.

Accessing data You have seen examples where the LOAD operator was reading data from a specific file. But it can go beyond that. If only a directory is specified, then the LOAD operator will read all files within that directory.

### Case sensitivity

* Are
  + Relations
  + Fields
  + Function names
* Not
  + Keywords
  + operator names

### Field references

Fields are assigned names through the use of a schema and fields may be referenced by their names. But a field can also be referenced by position as well.

This is accomplished by specifying a $ and a number. Field numbers start with zero. So with a schema of (name: chararray, age:int, salary:float), the age field could be reference by position as $1.

If a schema is not specified when data is loaded, then the fields can only be referenced by position.

### Operators

Pig Latin allows for the normal arithmetic operators found in programming languages - addition, subtraction, multiplication, and division. Also included in modulo.

Binary condition is essentially a shorthand notation for an if then else.

X == 5 ? 1 : 2;

If X equals 5,

then 1 is returned,

otherwise, 2 is returned.

Conditional processing is handled by Boolean operators, and, or, not.

Field values can be compared using the comparison operators.

Equal to is written as == and not equal is written as !=.

is null will check if a field is NULL.

## Describe how to substitute parameters in a Pig script

Parameter substitution Parameters can be passed to a Pig script.

This allows a single generic script to be used for a specific situation. Values that are not common across all executions of the script can be changed for each invocation of the script.

Within the script, parameters are referenced by preceding the parameter name with a $.

There are two ways to pass parameter values to a script.

1. Parameters can be set to values on the command line when the script is invoked
2. the parameters, along with their values, can be defined in a parameter file. Then that parameter file is referenced when the script is invoked.

Assume that within your Pig script, myscript.pig, you have the statement

a = load ‘$dir/myfile.txt’;

Your script is assuming that a parameter by the name of dir is going to be passed. Now let’s look at the two ways that the value for dir can be specified. Executing

pig -param dir=’/labfiles’ myscript.pig

passes dir to your script with a value of

/labfiles.

So the load statement in your script expands to

a = load ‘/labfiles/myfile.txt’;

A second technique is to have a file that has the parameter values.

Assume that the name of this file is myparams.txt. Within this file, you would code a statement

dir = ‘/labfiles’.

Then when you invoke your script, you would execute

pig -param\_file myparams.txt myscript.pig.

## Explain how to use the OUTPUT operator to write data

Output The STORE operator saves results to a file.

It uses built-in functions, similar to the LOAD operator, to determine how the output data is to be formatted.

* PigStorage is once again the default and the tab character is the default delimiter.
* BinStorage is used by Pig for temporary files.
* PigDump stores the data as tuples in a human-readable UTF-8 format.
* JsonStorage writes data in a JSON format.

As with the LOAD operator, you can code and specify your own function. The format of the STORE operator is

STORE alias name into ‘<directory>’ [using function()];

The specified directory will be created. If it already exits, then the operation will fail.

Files written into the directory will be of the format

part-nnnnn. --nnnnn is a numeric value.

As stated before, the DUMP operator writes the results to the console.

Basically, it is designed for debugging and testing.

continue with lesson 2. Trademarks The following are trademarks of IBM and other companies

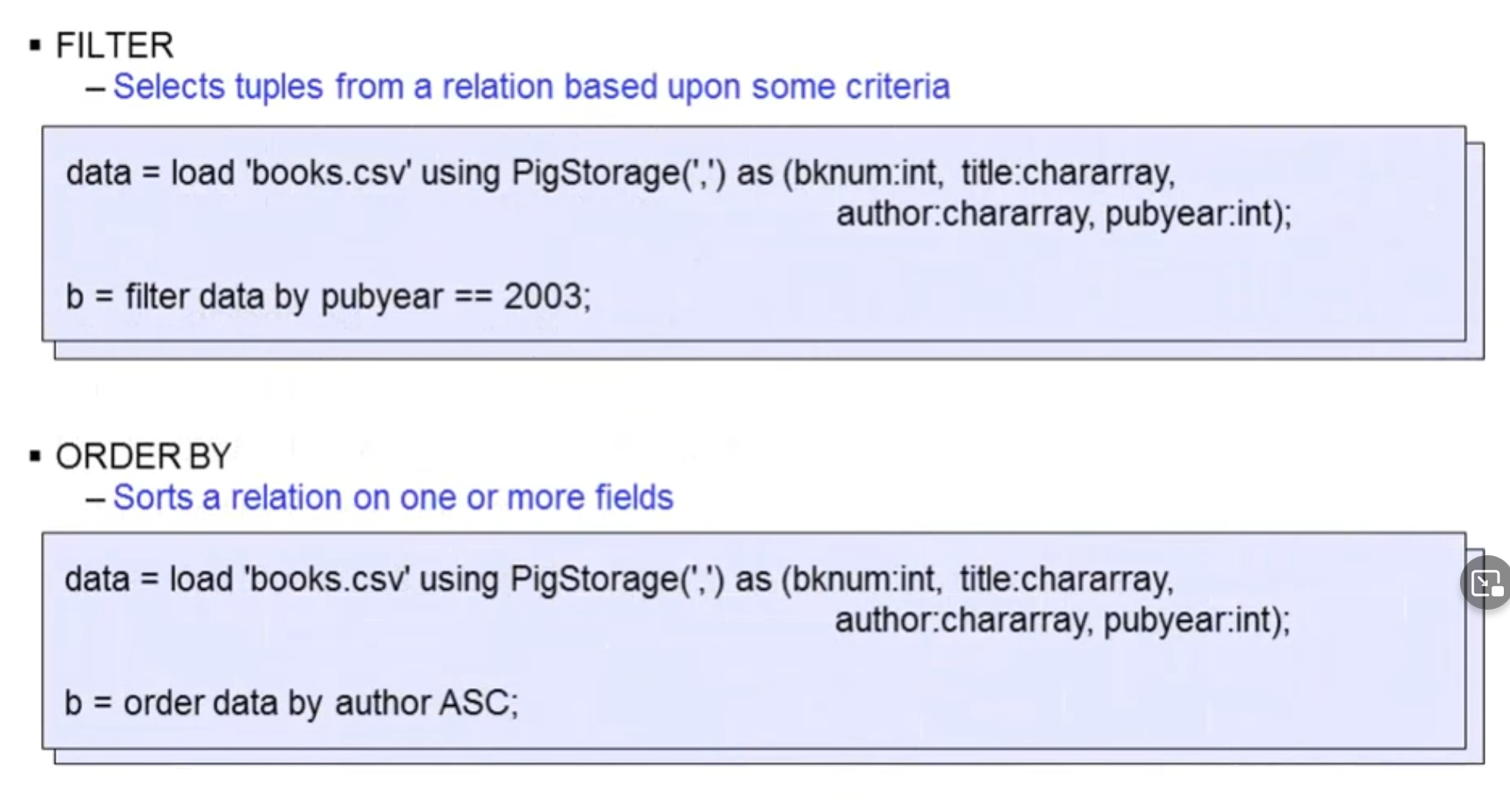
# Lesson 2 - Pig Relational Operators

## List the relational operators used in Pig

### Filter

### Order by

Sorts a relation on one or more fie



To sort in descending sequence, you would specify DESC.

Also, multiple comma separated fields may be specified, each specifying their own sort sequence.

### Group

Q: For the tuples (3,5,2) (5,2,1) (3,7,3) (3,6,1), using the GROUP operator on the third field produces the following: (2,{(3,5,2)}), (1,{(5,2,1),(3,6,1)}), (3,{(3,7,3)}). True or false? Disregard order when answering.

Q UNION, GROUP, and COGROUP can be used interchangeably without creating different outputs. True or false?

Groups together tuples that have the same group key

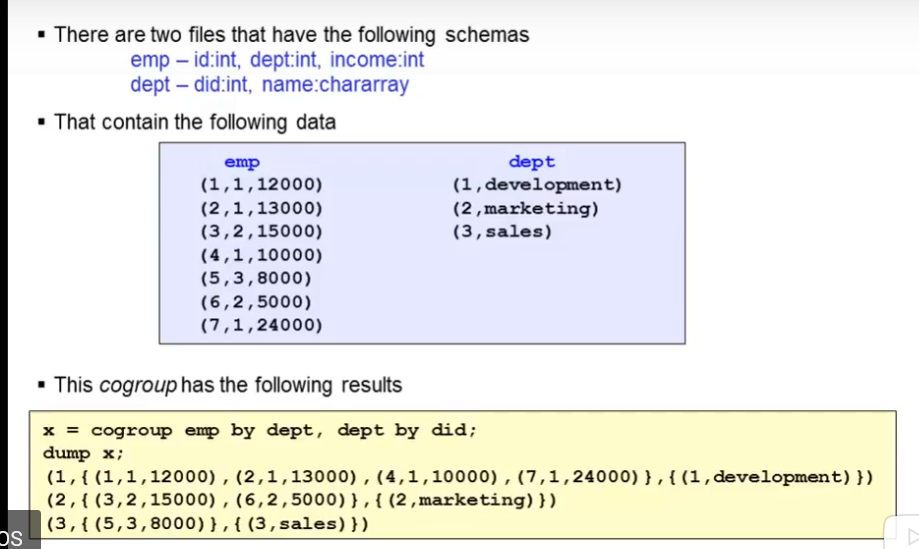
* + The group key can be a single field or multiple fields
    - If multi enclose in parentheses
* Result is a relation that contains one tuple per group
  + This tuple contains two fields
    - Group
      * (Do not get this confused with the name of the operator.)
      * contains value of group key
    - Name of the original relation that was grouped
* Format
  + Alias = GROUP alias by <all | expression> [PARTITION BY partitioner][PARALLEL n];
* Example

Data = load ‘data.csv’ using PigStorag(‘,’) as (f1:int,f2:int,f3:int);  
Dump data;  
(1,2,3)  
(4,5,6)  
(7,8,9)  
(4,3,2)

Grp = group data by f1  
Dump grp  
(1,{(1,2,3)})  
(4,{(4,5,6),(4,3,2)})  
(7,{(7,8,9)})

### Cogroup

GROUP and COGROUP operators are the same operator but, by convention, the COGROUP operator is used when grouping multiple relations at the same time.

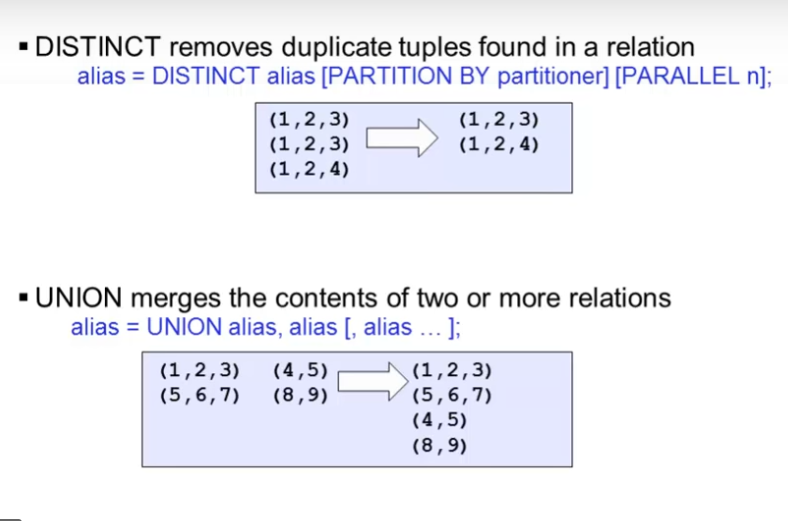


### FOREACH

Q: Which operators can be used within a nested FOREACH block?  
A: LIMIT, DISTINCT, ORDER BY, FILTER

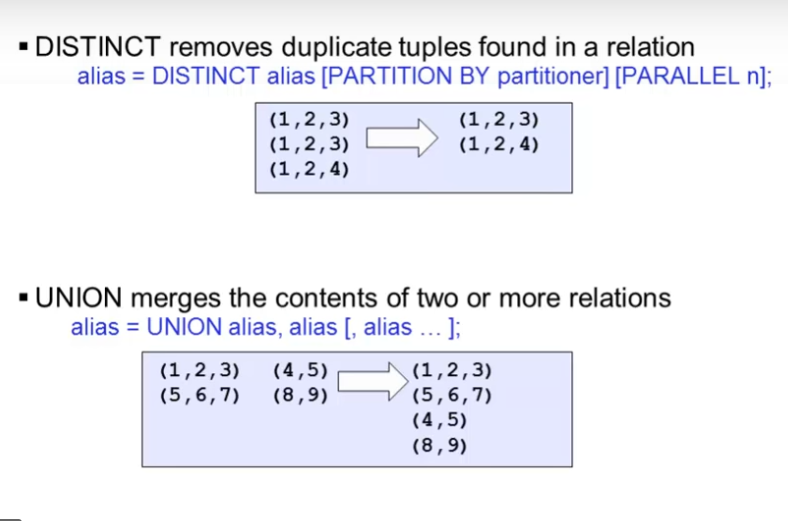
* allows you to create a new relation through projection.
* This new relation may have fewer fields or more fields than the original relation.
* There are two formats for this operator.
  + One that works with a block
  + one that works with a nested block.
* Alias = FOREACH{block | nested\_block};

### DISTINCT



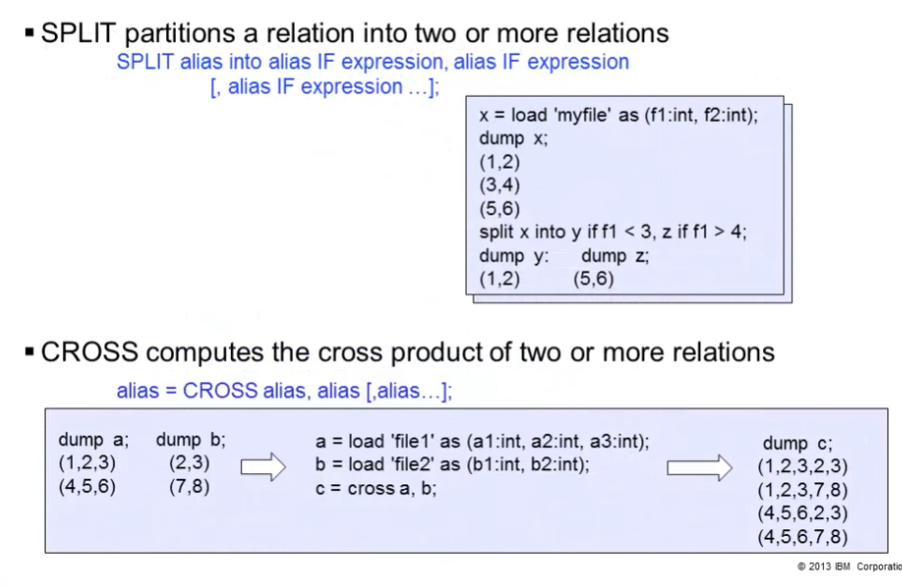
### UNION

merges the contents of two or more relations.



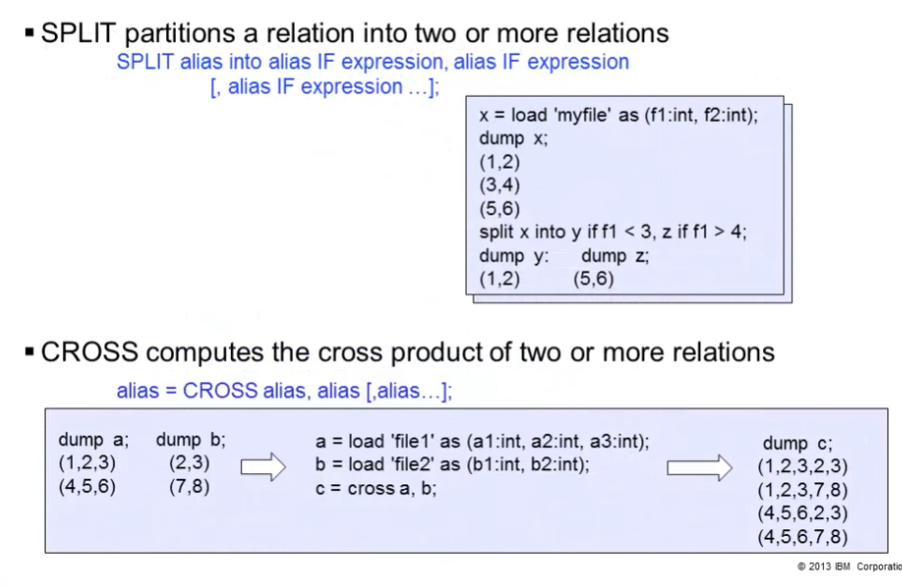
### SPLIT

SPLIT partitions a relation into two or more relations based upon some conditional processing. For example split x into y if f1 < 3; z if f1 > 4;



### CROSS

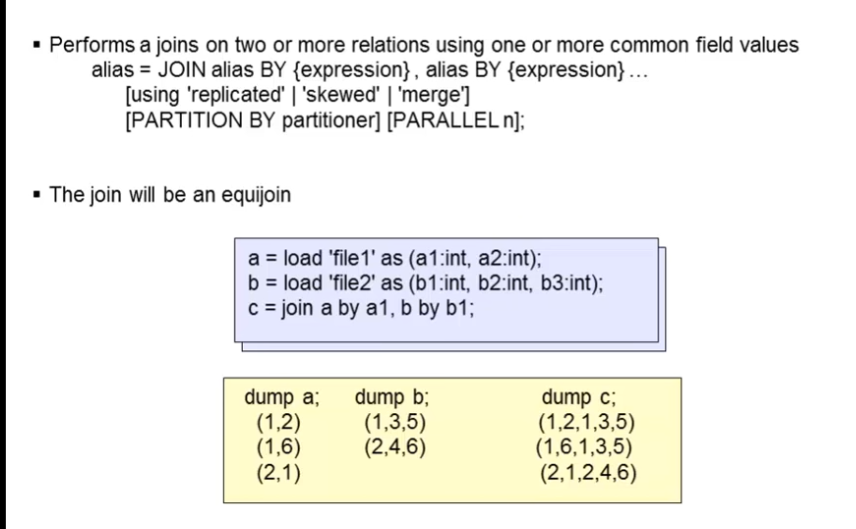
CROSS computes the cross product of two or more relations.



### JOIN – inner

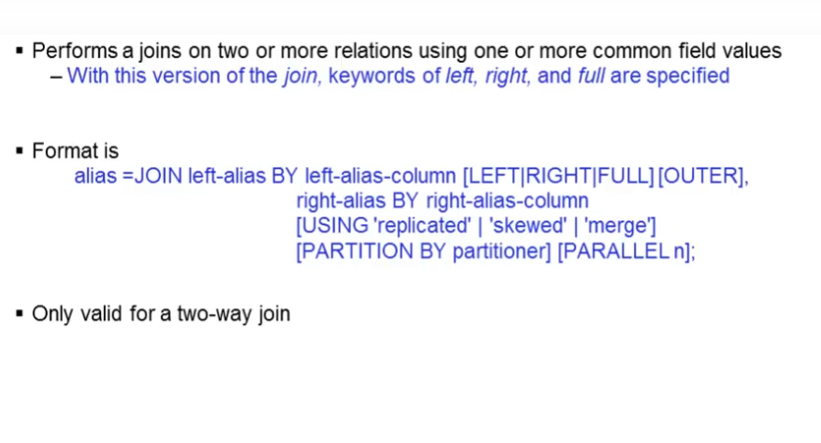
This performs a join on two or more relations using one or more common fields.

The join will be an equijoin. So c = join a by a1, b by b1; will join the tuples in relations a and b when the a1 field in relation a is equal to the b1 field in relation b.



### JOIN – outer

* With this version of the join, keywords left, right and full are specified. It is only valid for a two-way join.
* Left, right, and full are specified
* Only valid for two way join



### JOIN ‘using’

This is designed to give you some control on how the joins are to be processed

#### Replicated

* Replicated joins work well if one or more of the relations can fit in memory.
* This allows the Hadoop work to be done on the map side.

#### Skewed

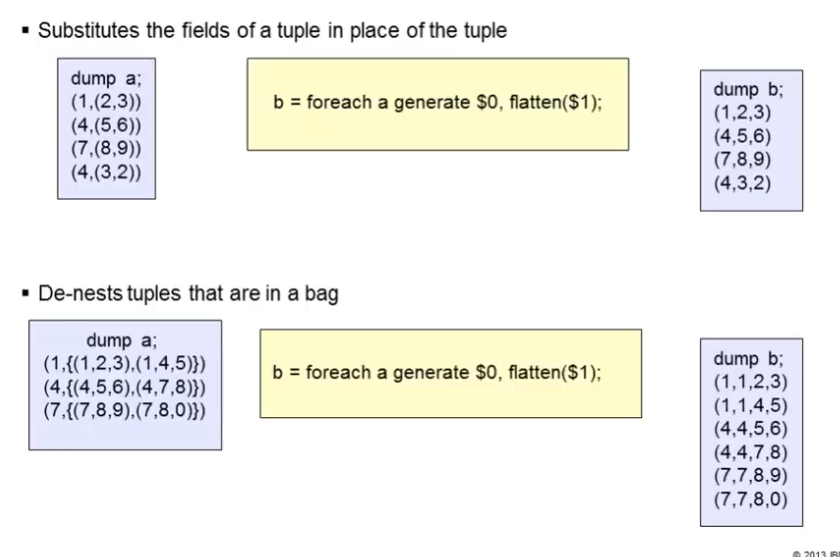
* Skewed joins should be used where the data is skewed.
* It computes a histogram of the key space.
  + Uses this data to allocate reducers for a given key

#### Merged

* Merged joins where both inputs to the join are already sorted on the join key.
* This allows for the data to be joined in the map phase of the MapReduce job.

## Explain how to flatten your data

The FLATTEN operator, when working with tuples, substitutes the field of a tuple for that tuple. If you have (1, (2,3)), then flattening it would result in (1,2,3).



## Describe dereference operators

* Sometimes it is necessary to reference a field in a tuple or bag that is outside the scope of the current operator.
  + This is especially true with the FOREACH operator.
  + data = load 'myfile' as (f1:int, f2:int, f3:int);  
    y = group data by f1;  
    z = foreach y generate group, data.f2, data.f3;
  + Fields f2 and f3 are not defined in relation y.
* The fields had to be qualified with the tuple or bag name
  + Since f2 and f3 were defined in the data relation
  + Data.f2 data.f3
* Can also be done by field position
  + Data.$1 data.$2

### Name

In order to reference them, they have to be qualified with either a tuple or bag name where they were defined.

### Position

It is also possible to dereference by position as well, for instance, data.$1 or data.$2.

# Evaluation Functions and Other Commands

## List the eval functions used in Pig

### IsEmpty

* Checks if bag or map is empty

### DIFF

* Compares two fields in a tuple

### COUNT\_STAR

* Computers the number of elements in a bag

### COUNT

* Counts the number of elements in a bag
  + Requires a preceding GROUP ALL statement for global counts
  + Requires a preceding GROUP BY statement for group counts

### CONCAT

* Concatenates two columns

### AVG

* Computes the average of the number of values in a single column bag

### MAX

* Computers the maximum value in a single column bag
  + Requires a preceding GROUP ALL statement for global counts
  + Requires a preceding GROUP BY statement for group counts

### MIN

* Computers the minimum value in a single column bag
  + Requires a preceding GROUP ALL statement for global counts
  + Requires a preceding GROUP BY statement for group counts

### SIZE

### Computes the number of elements based on any Pig Data type

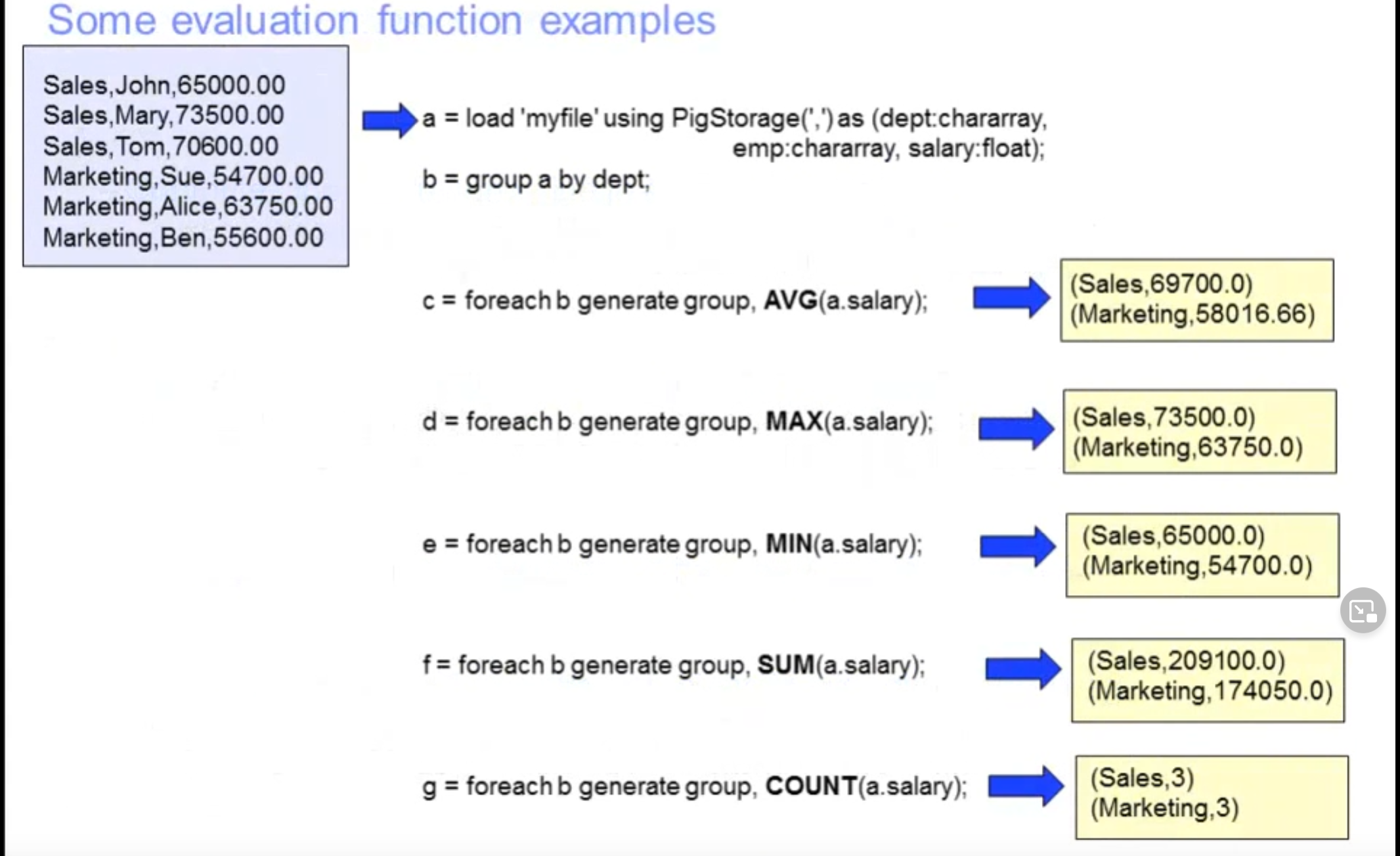
### SUM

* Computers the sum of the numeric values in a single column bag
  + Requires a preceding GROUP ALL statement for global counts
  + Requires a preceding GROUP BY statement for group counts

### TOKENIZE

* Splits a string and outputs a bag of words

### Examples



### MATH

* Pig has number of available math functions based on the Java class MATH

#### ABS

absolute values

#### CEIL

rounds up to nearest integer

#### Trig functions

#### Etc

### STRING

* Based on Java class String

#### STRSPLIT

Split a string around matches of a given regular expression

#### SUBSTRING

Returns a substring from a given string

#### REPLACE

Replaces old characters with new characters

#### REGEX\_EXTRACT

#### REGEX\_EXTRACT\_ALL

Performs regular expression matching and extracts the matched groups

### Tuple, bag, and map

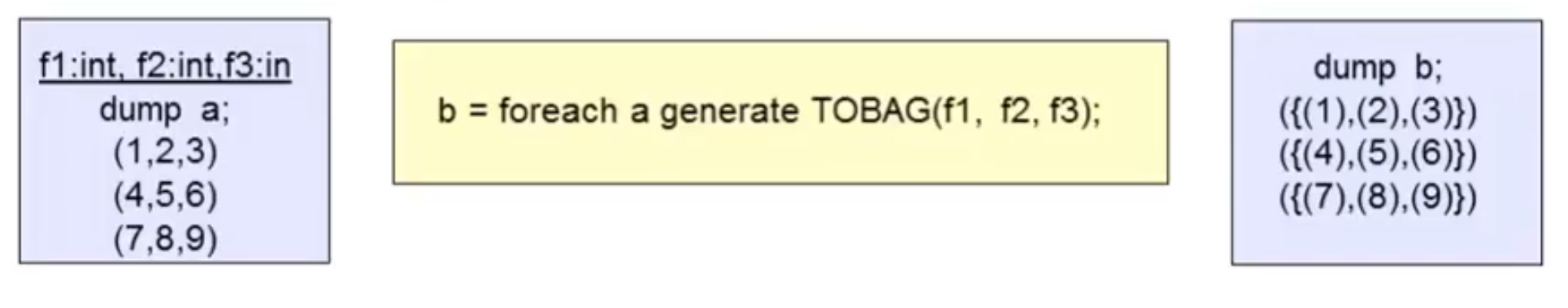
#### TOTUPLE

* Converts one or more expressions into a tuple



#### TOBAG

* Converts one or more expressions into a type bag



#### TOMAP

* Converts pairs to expressions into a map

## Describe how to invoke HDFS shell commands from a Pig script or the Grunt shell

### MAPREDUCE

* Executes a native MapReduce jobs inside a pig script

### STREAM

* Sends data to an external script or program

### UDF STATEMENT

#### DEFINE

* Assigns an alias to a user defined function or a streaming command
* Helps if name of UDF is a long package name or the streaming command is complex

#### REGISTER

* Registers a Jar file so that the UDFs in the file can be used

### FS

* Invokes any FSShell command from within a Pig script or the Grunt shell
  + Fs -mkdir /mydir

## Explain how to execute a Pig script from the Grunt shell

### UTILITY COMMANDS - EXEC

* Allows you to execute commands from within the Grunt shell
  + Grunt > exec myscript.pig

## Describe the use of the Explain operator

### Explain

* Used to review the logical, physical, and MapReduce execution plans

### Example

