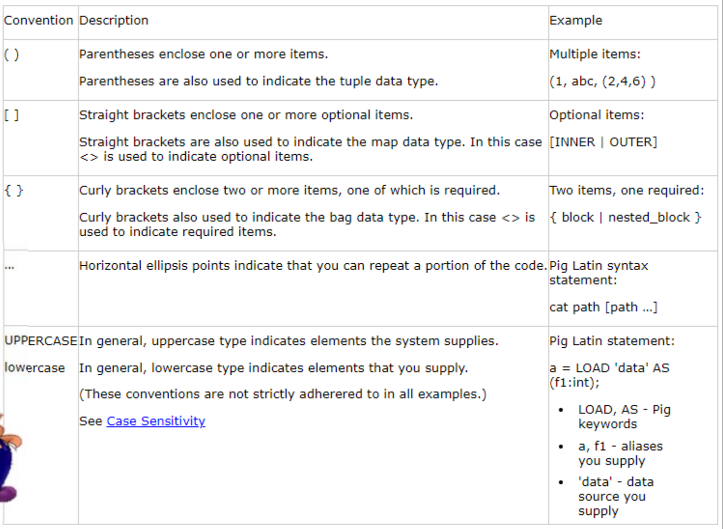
University of
South Australia

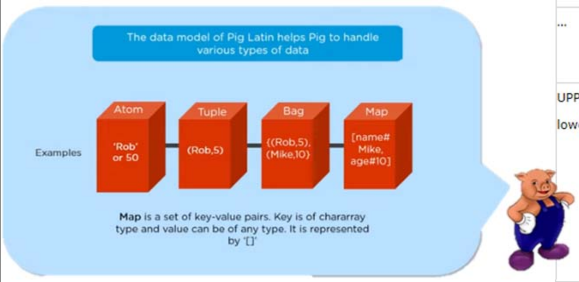
PigLatin

1

PigLatin Conventions



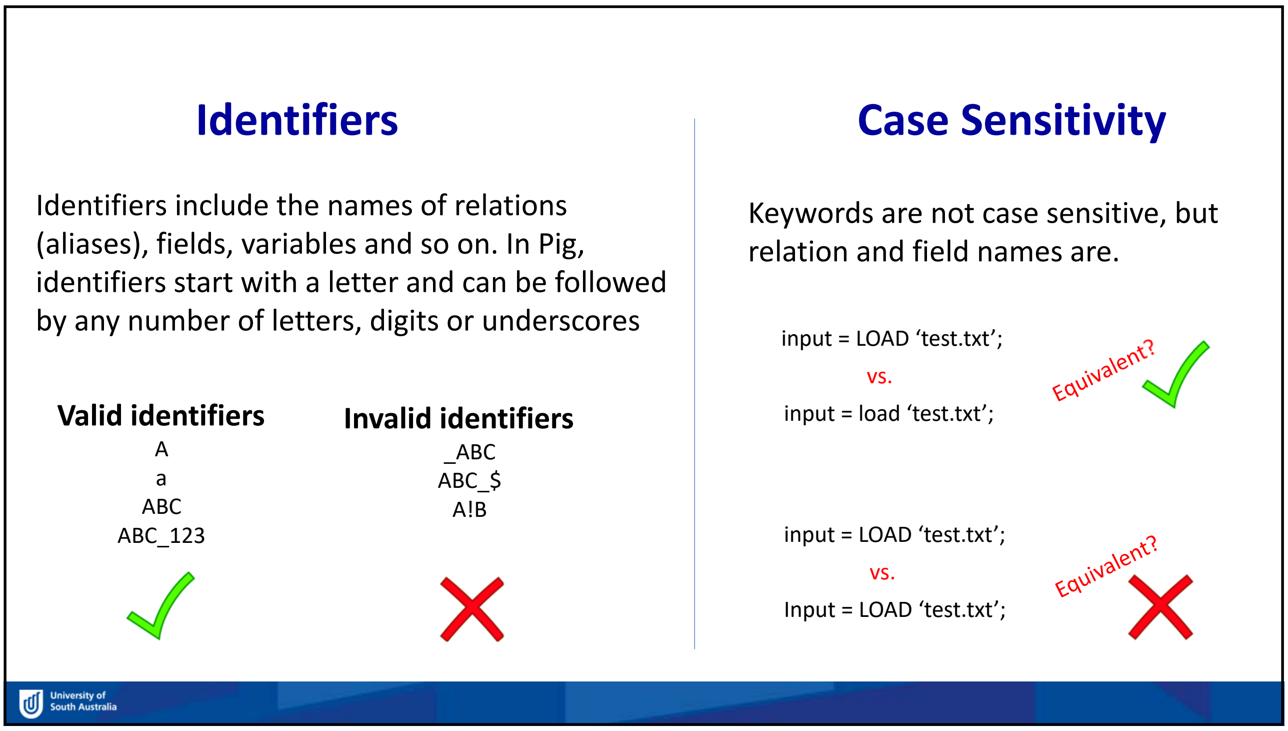
Convention	Description	Example
( )	Parentheses enclose one or more items. Parentheses are also used to indicate the tuple data type.	Multiple items: (1, abc, (2,4,6))
[ ]	Straight brackets enclose one or more optional items. Straight brackets are also used to indicate the map data type. In this case <> is used to indicate optional items.	Optional items: [INNER | OUTER]
{ }	Curly brackets enclose two or more items, one of which is required. Curly brackets also used to indicate the bag data type. In this case <> is used to indicate required items.	Two items, one required: { block | nested_block }
...	Horizontal ellipsis points indicate that you can repeat a portion of the code.	Pig Latin syntax statement: cat path [path ...]
UPPERCASE lowercase	In general, uppercase type indicates elements the system supplies. In general, lowercase type indicates elements that you supply. (These conventions are not strictly adherered to in all examples.) See Case Sensitivity	Pig Latin statement: a = LOAD 'data' AS (f1:int); • LOAD, AS - Pig keywords • a, f1 - aliases you supply • 'data' - data source you supply


The data model of Pig Latin helps Pig to handle various types of data
Examples
Atom
'Rob' or 50
Tuple
(Rob,5)
Bag
{(Rob,5),(Mike,10)}
Map
[name# Mike, age#10]
Map is a set of key-value pairs. Key is of chararray type and value can be of any type. It is represented by '[]'

Simpilearn

University of
South Australia

2



Identifiers

Identifiers include the names of relations (aliases), fields, variables and so on. In Pig, identifiers start with a letter and can be followed by any number of letters, digits or underscores

Valid identifiers	Invalid identifiers
A	_ABC
a	ABC_$
ABC	A!B
ABC_123	
✓	✗

Case Sensitivity

Keywords are not case sensitive, but relation and field names are.

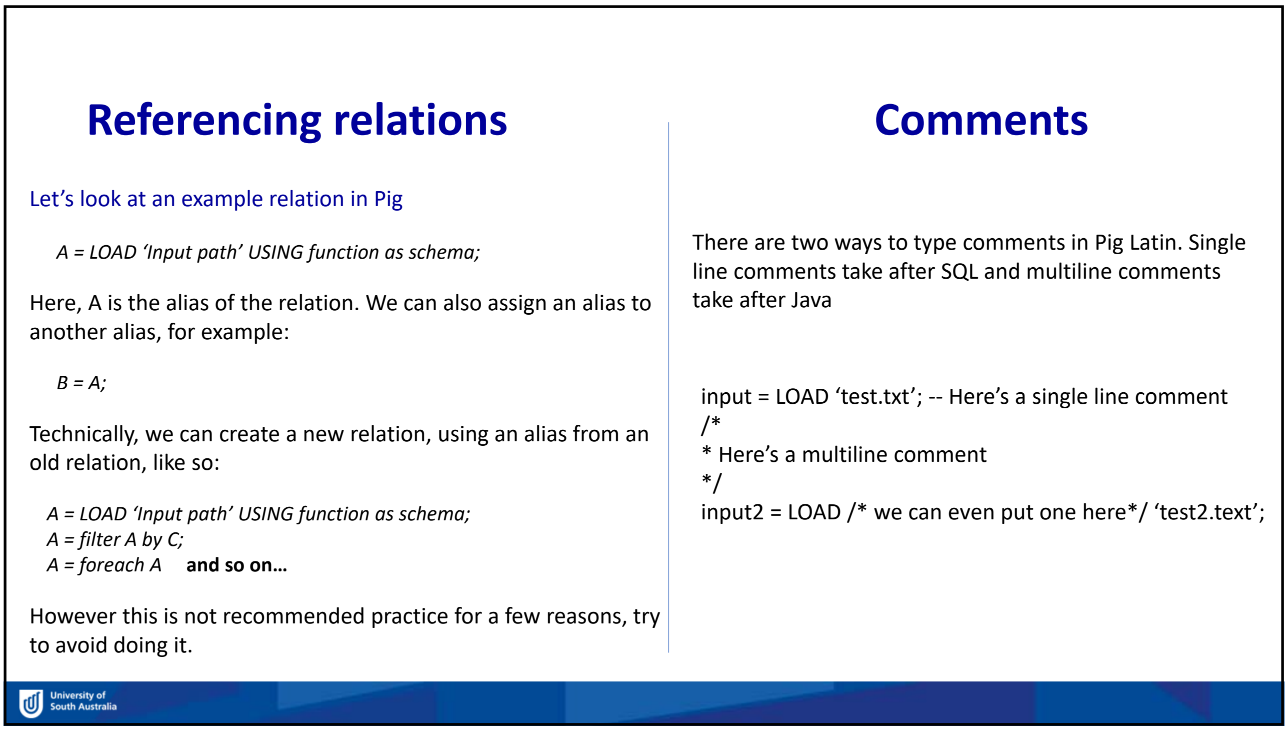
```
input = LOAD ‘test.txt’;
        vs.
input = load ‘test.txt’;
```

Equivalent? ✓

```
input = LOAD ‘test.txt’;
        vs.
Input = LOAD ‘test.txt’;
```

Equivalent? ✗

3



Referencing relations

Let’s look at an example relation in Pig

A = LOAD ‘Input path’ USING function as schema;

Here, A is the alias of the relation. We can also assign an alias to another alias, for example:

B = A;

Technically, we can create a new relation, using an alias from an old relation, like so:

A = LOAD ‘Input path’ USING function as schema;
A = filter A by C;
A = foreach A **and so on…**

However this is not recommended practice for a few reasons, try to avoid doing it.

Comments

There are two ways to type comments in Pig Latin. Single line comments take after SQL and multiline comments take after Java

```
input = LOAD ‘test.txt’; -- Here’s a single line comment
/*
* Here’s a multiline comment
*/
input2 = LOAD /* we can even put one here*/ ‘test2.text’;
```

4

The LOAD operator

On the left hand side of the **=** sign we need to define the name of the relation where our data will be stored. On the right hand side we need to define how our relation will be loaded.

relation_name = LOAD 'Input path' USING function as schema;

relation_name	Input path	function	schema
The name of the relation in which we want to store the data	The directory where the file is stored	A set of load functions are provided by Pig, we choose which one to use	We can define the schema of the data

relation_name = LOAD 'Input path' USING function as schema;

'function'

pigStorage
A load function that parses a line of input into fields using a character delimiter. The default delimiter is a tab.

CSVLoader
A load function based on PigStorage that implements part of the CSV "standard" This loader properly supports double-quoted fields that contain commas and other double-quotes escaped with backslashes.

'schema'

If we specify a schema it should be done in the following manner

(column1: data type, column2: data type, column3: data type);

For example

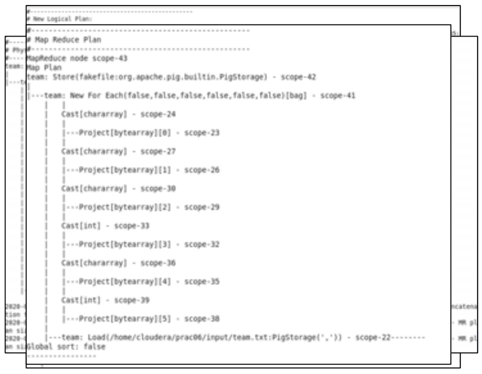
(name: chararray, age: int, salary: float

If we don't specify a schema, we can simply refer to the columns as

$01, $02 etc.

Diagnostic Operators

explain *team;*



```
#-----------------------------------------------
# New Logical Plan:
#-----------------------------------------------
# Map Reduce Plan
#-----------------------------------------------
MapReduce node scope-43
Map Plan
team: Store(fakefile:org.apache.pig.builtin.PigStorage) - scope-42
|
|---team: New For Each(false,false,false,false,false,false)[bag] - scope-41
    |   |
    |   Cast[chararray] - scope-24
    |   |
    |   |---Project[bytearray][0] - scope-23
    |   |
    |   Cast[chararray] - scope-27
    |   |
    |   |---Project[bytearray][1] - scope-26
    |   |
    |   Cast[chararray] - scope-30
    |   |
    |   |---Project[bytearray][2] - scope-29
    |   |
    |   Cast[int] - scope-33
    |   |
    |   |---Project[bytearray][3] - scope-32
    |   |
    |   Cast[chararray] - scope-36
    |   |
    |   |---Project[bytearray][4] - scope-35
    |   |
    |   Cast[int] - scope-39
    |   |
    |   |---Project[bytearray][5] - scope-38
    |
    |---team: Load(/home/cloudera/prac06/input/team.txt:PigStorage(',')) - scope-22--------
Global sort: false
----------------
```

dump *team;*

```
(Taylor,Walker,fwd,13,Broken Hill,30)
(Rory,Sloane,mid,9,Melbourne,30)
(Daniel,Talia,def,12,Kilmore,28)
(Brad,Crouch,mid,2,Ballarat,26)
(Matthew,Crouch,mid,5,Ballarat,25)
```

describe *team;*

```
grunt> describe team
team: {firstName: chararray,lastName: chararray,position: chararray,playerNum: int,hometown: chararray,age: int}
```

illustrate *team;*

```
----------------------------------------------------------------------------------------------------------------------------------------
| team     | firstName:chararray     | lastName:chararray     | position:chararray     | playerNum:int     | hometown:chararray     | age:int     |
----------------------------------------------------------------------------------------------------------------------------------------
|          | Brad                    | Crouch                 | mid                    | 2                 | Ballarat               | 26          |
----------------------------------------------------------------------------------------------------------------------------------------
```