PIG

Índice

Conceptos básicos

Las Bases de Datos Relacionales tienen tablas, filas, columnas y campos

Asumiendo la siguiente representación:

name	price	country
Alice	2999	us
Bob	3625	ca
Carlos	2764	mx
Dieter	1749	de
Étienne	2368	fr
Fredo	5637	it

Conceptos básicos

Un ***field*** es un elemento en sí mismo.

name	price	country
Alice	2999	us
Bob	3625	ca
Carlos	2764	mx
Dieter	1749	de
Étienne	2368	fr
Fredo	5637	it

Conceptos básicos

Una ***collection*** de valores es llamado ***tuple***.

name	price	country
Alice	2999	us
Bob	3625	ca
Carlos	2764	mx
Dieter	1749	de
Étienne	2368	fr
Fredo	5637	it

Conceptos básicos

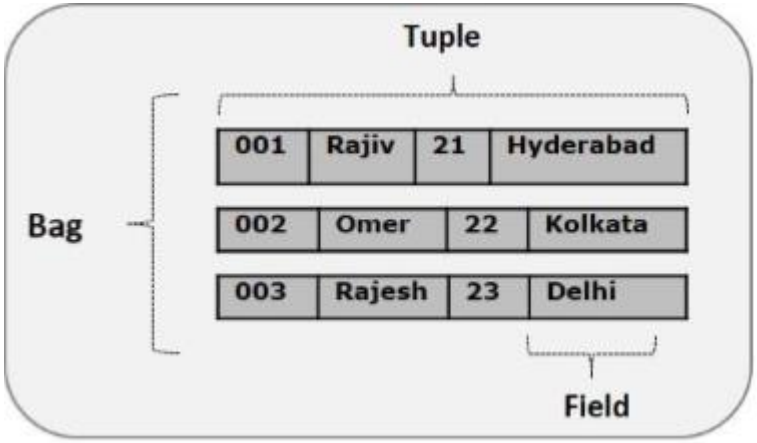
Una ***collection*** de ***tuples*** es llamado ***bag***.

name	price	country
Alice	2999	us
Bob	3625	ca
Carlos	2764	mx
Dieter	1749	de
Étienne	2368	fr
Fredo	5637	it

Conceptos básicos

Una **relación** es simplemente una **bag** con un nombre asignado (**alias**)

La mayoría de las sentencias PIG Latin son creaciones de nuevas relaciones



Índice

PIG LATIN

- Pig Latin es un lenguaje de flujo de datos
  - El flujo de datos es representado por una secuencia de instrucciones

- El ejemplo siguiente es un script de Pig Latin para cargar, filtrar y almacenar los datos

```
allsales = LOAD 'sales' AS (name, price);

bigsales = FILTER allsales BY price > 999; -- in US cents

/*
 * Save the filtered results into a new
 * directory, below my home directory.
 */
STORE bigsales INTO 'myreport';
```

KEY WORDS

- Las **palabras clave** son palabras reservadas por PIG. No es posible utilizar dichas palabras para nombrar cosas

```
allsales = LOAD 'sales' AS (name, price);

bigsales = FILTER allsales BY price > 999; -- in US cents

/*
 * Save the filtered results into a new
 * directory, below my home directory.
 */
STORE bigsales INTO 'myreport';
```

IDENTIFICADORES

- Los **identificadores** son los nombres asignados a campos u otras estructuras

```
allsales = LOAD 'sales' AS (name, price);

bigsales = FILTER allsales BY price > 999; -- in US cents

/*
 * Save the filtered results into a new
 * directory, below my home directory.
 */
STORE bigsales INTO 'myreport';
```

COMENTARIOS

- Pig Latin soporta dos tipos de comentarios
  - Línea sencilla comentada comenzando por --
  - Múltiples líneas comentando comenzando por /* y finalizando por */

```
allsales = LOAD 'sales' AS (name, price);

bigsales = FILTER allsales BY price > 999; -- in US cents

/*
 * Save the filtered results into a new
 * directory, below my home directory.
 */
STORE bigsales INTO 'myreport';
```

OPERACIONES COMUNES

- Muchos operadores son similares a los utilizados en SQL
  - Pig Latin usa == y != para las comparaciones

Arithmetic	Comparison	Null	Boolean
+	==	IS NULL	AND
-	!=	IS NOT NULL	OR
*	<		NOT
/	>		
%	<=		
	>=		

Índice

CARGA BÁSICA DE DATOS

- La función por defecto para la carga de datos se denomina PigStorage
  - El nombre de la función está implícito en la instrucción LOAD
  - PigStorage asume el formato de texto separando las columnas por tabulador

```
Alice	2999
Bob	3625
Carlos	2764
```

```
allsales = LOAD 'sales' AS (name, price);
```

ORÍGENES DE DATOS: FICHEROS Y DIRECTORIOS

- El siguiente ejemplo carga el fichero en la variable ‘sales’

```
allsales = LOAD 'sales' AS (name, price);
```

- Si no es una ruta absoluta, será una ruta relativa al directorio del home del usuario
  - El directorio home en el HDFS normalmente es /user/idusuario/
  - Podemos especificar una ruta absoluta (“/tmp/sales/2015/”)

ESPECIFICACIÓN DE COLUMNA EN LA CARGA

- El siguiente ejemplo asigna nombres a cada una de las columnas

```
allsales = LOAD 'sales' AS (name, price);
```

- Asignar nombres a las columnas no es obligatorio
  - En caso de no indicar nombre, podremos recorrer las columnas a través de los identificadores $0, $1, $2….

```
allsales = LOAD 'sales';
```

DELIMITACIÓN DE COLUMNAS ALTERNATIVOS

- Es posible indicar delimitadores de columna alternativos como argumento de la función PigStorage

```
allsales = LOAD 'sales.csv' USING PigStorage(',') AS
(name, price);
```

```
allsales = LOAD 'sales.txt' USING PigStorage('|');
```

ALMACENAMIENTO DE DATOS

- El comando usado para obtener la salida nos indicará el formato de la misma
  - **DUMP**: Muestra la salida por pantalla
  - **STORE**: Envía los resultados al disco (HDFS)

Ejemplo del comando DUMP: **DUMP result**; Siendo result un **Identificador**

```
(Alice,2999,us)
(Bob,3625,ca)
(Carlos,2764,mx)
(Dieter,1749,de)
(Étienne,2368,fr)
(Fredo,5637,it)
```

ALMACENAMIENTO DE DATOS

- El comando **STORE** es usado para almacenar los datos en **HDFS**
  - Igual que **LOAD**, pero realiza la escritura en vez de la lectura
  - El path de salida se corresponde con el directorio de salida
    - El directorio no debe existir

- Con el comando **LOAD**, el uso del 'PigStorage' está implicito
  - El delimitador de los campos es el de por defecto (tab)

```
STORE bigsales INTO 'myreport';
```

  - Es posible indicar un delimitador propio

```
STORE bigsales INTO 'myreport' USING PigStorage(',');
```

Índice

TIPOS DE DATOS SIMPLES

- Pig soporta muchos tipos de datos básicos
  - Similar a muchas bases de datos o lenguajes de programación

- Pig trata a los “fields” no identificados como array de bytes
  - Llamado *bytearray* en Pig

```
allsales = LOAD 'sales' AS (name, price);
```

TIPOS DE DATOS SIMPLES

TIPO	EJEMPLO
int	2013
long	5,365,214,142L
float	3.14159F
double	3.141592653589793846
boolean*	true
datetime*	2013-05-30T14:52:39.000-04:00
chararray	Alice
bytearray	N/A

ESPECIFICACIÓN DE LOS TIPOS DE DATOS

- Pig realiza la mejor operación para determinar el mejor tipo de datos basado en el contexto
  - Por ejemplo, tu puedes calcular la comisión de las ventas con price * 0.1
  - En este caso, **Pig asumirá que el tipo** de este valor es de tipo double

- De todas formas, es **mejor especificar** el tipo de datos cuando sea posible
  - Es importante elegir el mejor tipo de dato para evitar perder en precisión y ganar velocidad

```
allsales = LOAD 'sales' AS (name:chararray, price:int);
```

DATOS INVÁLIDOS

- Cuando nos encontramos con datos inválidos, Pig los sustituye por **NULL**
  - Por ejemplo, si un campo de tipo int recibe un String

- Es posible utilizar **IS NULL** y **IS NOT NULL** para filtrar los registros erróneos

```
hasprices = FILTER Records BY price IS NOT NULL;
```

Índice

FILTRADO DE REGISTROS

- El comando **FILTER** nos permitirá extraer las tuplas que cumplan un determinado requisito

```
bigsales = FILTER allsales BY price > 3000;
```

- Es posible combinar diferentes requisitos con **AND** y **OR**

```
somesales = FILTER allsales BY name == 'Dieter' OR (price >
3500 AND price < 4000);
```

COMPARACIÓN DE REGISTROS

- El operador **==** es soportado por cualquier tipo de dato

```
alices = FILTER allsales BY name == 'Alice';
```

- Pig Latin soporta la utilización de patrones a través de **expresiones regulares de Java**
  - Esto es posible a través del operador **MATCHES**

```
a_names = FILTER allsales BY name MATCHES 'A.*';
```

```
spammers = FILTER senders BY email_addr
           MATCHES '.*@example\\.com$';
```

SELECCIÓN Y GENERACIÓN DE CAMPOS

- Es posible realizar la extracción de columnas
  - Esto es posible gracias a los operadores FOREACH y GENERATE

```
twofields = FOREACH allsales GENERATE amount, trans_id;
```

- Con los operadores FOREACH y GENERATE podemos generar nuevos campos
  - Por ejemplo, deseamos crear un nuevo campo basado en el precio

```
t = FOREACH allsales GENERATE price * 0.07;
```

  - Es posible indicarle el nombre

```
t = FOREACH allsales GENERATE price * 0.07 AS tax;
```

  - Es posible indicar el tipo

```
t = FOREACH allsales GENERATE price * 0.07 AS tax:float;
```

ELIMINACIÓN Y ORDENACIÓN DE RESULTADOS

- El comando **DISTINCT** elimina los registros duplicados de una "**bag**"
  - Todos los campos deben ser iguales para considerar el registro como duplicado

```
unique_records = DISTINCT all_alices;
```

- El comando **ORDER….BY** permite ordenar todos los registros de una "**bag**" en orden ascendente
  - Para ordenar de forma descendente habrá que añadir el operador **DESC**

```
sortedsales = ORDER allsales BY country DESC;
```

Índice

FUNCIONES BÁSICAS

- En PIG podemos utilizar diferente funciones predefinidas, como por ejemplo:

Función	Entrada	Salida
UPPER(country)	uk	UK
TRIM(name)	_Bob_	Bob
RANDOM()		0.48161326652569
ROUND(price)	37.19	37
SUBSTRING(name, 0, 2)	Alice	Al

```
rounded = FOREACH allsales GENERATE ROUND(price);
```

Índice

Ejercicio Práctico

- Disponemos un fichero de entrada con la siguiente estructura (campos separados por comas) :

Campo	Tipo
Key	Chararray
Campaña	Chararray
Fecha	Chararray
Tiempo	Chararray
Display	Chararray
Acción	Int
Cpc	Int
Pais	Chararray
Lugar	Chararray

Ejercicio Práctico

- Ejemplo del Fichero:

lightweight	D8 SIDE	05/01/2013	00:00:08	gamersite.example.com	0	72	USA
accelerometer	B1 INLINE	05/01/2013	00:00:10	datawire.example.com	0	78	USA
pc	D3 BOTTOM	05/01/2013	00:00:16	datasnap.example.com	0	49	USA
dualcore	D7 SIDE	05/01/2013	00:00:22	datawire.example.com	0	58	USA
apps	C2 INLINE	05/01/2013	00:00:23	albumreview.example.com	0	72	NETHERLANDS
review	D7 SIDE	05/01/2013	00:00:37	amateurcoder.example.com	0	66	USA
browser	D5 INLINE	05/01/2013	00:00:39	datascientist.example.com	0	79	USA
touchscreen	B2 SIDE	05/01/2013	00:00:47	burritofinder.example.com	0	84	USA
social	A2 TOP	05/01/2013	00:01:06	photosite.example.com	0	82	USA

Ejercicio Práctico

Ver hoja de ejercicios

Dudas y Debate