Pig Latin

Podstawowe typy danych	Opis & przykład
int, long	Reprezentuje liczby całkowite. Np. 8
float, double	Reprezentuje liczby zmiennoprzecinkowe. Np. 3.14
chararray	Reprezentuje tablice znaków – tekst. Np. ‘witaj’
bytearray	Reprezentuje tablice bajtów – plik.
boolean	Przyjmuje zawrtość *true* lub *false*.

Złożone typy danych	Opis & przykład
tuple	Krotka jest uporządkowanym zbiorem pól. (1,Polska,Warszawa,37.97)
bag	Torba to zbiór krotek. Często jest też nazywany relacją. {(1,Polska,Warszawa,37.97),(2,Niemcy,Berlin,82.79),(3,Austria,Wiedeń,8.77)}
map	To para klucz-wartość służąca do reprezentowania danych. [Nr#1,Kraj#Polska,Stolica#Warszawa,LiczbaLudnosci#37.97]

Operatory	Opis
LOAD	służy do wczytywania danych z lokalnego systemu plików lub HDFS’a do Pig.
LIMIT	służy do uzyskania ograniczonej liczby krotek z relacji.
DUMP	służy to wyświetlania relacji na konsoli.
DESCRIBE	służy do przeglądania schematu relacji.
FILTER	służy do wyboru wymaganych krotek z relacji na podstawie warunku.
STORE	służy do zapisywania relacji w określonym pliku.
GROUP	służy do grupowania danych w jedną lub więcej relacji. Gromadzi dane mające ten sam klucz.
FOREACH	służy do generowania określonych transformacji dla danej kolumny.
ORDER BY	służy do wyświetlania zawartości relacji w uporządkowanej kolejności na podstawie jednego lub więcej pól.
DISTINCT	służy do usuwania zbędnych (zduplikowanych) krotek z relacji.
JOIN	służy do łączenia rekordów z dwóch lub więcej relacji.
SPLIT	służy do podziału relacji na dwie lub więcej relacji.
UNION	służy do łączenia dwóch relacji. Aby wykonać operację UNION na dwóch relacjach, ich kolumny i typy muszą być identyczne.

LOAD - służy do wczytywania danych.

Składnia:

relation_name = LOAD 'input_file_path' USING function as schema;

gdzie:

relation_name - nazwa relacji w której będziemy przechowywać dane.

input_file_path - to ścieżka dostępu do pliku.

function - to funkcja do wczytywania odpowiedniego rodzaju danych. Mamy 4 rodzaje funkcji do wczytywania danych:

- BinStorage() – używana wewnętrznie przez Pig do przechowywania tymczasowych danych;
- PigStorage('field_delimiter') – używana do wczytywania ustrukturyzowanych danych, gdzie field_delimiter oznacza separator.
- TextLoader() – używana do wczytywania nieustrukturyzowanych danych. W wyniku każda krotka zawiera tylko jedno pole z jedna linią z pliku wejściowego.
- JsonLoader() – używana do wczytywania danych w formacie JSON.

schema – to schemat naszych danych w następującym formacie: (column1:data type, column2: data type, column3:data type);

Przykłady:

```
wszystkie_loty = LOAD 'danilewicz/1987.csv' USING PigStorage(',');
```

```
wszystkie_loty = LOAD 'danilewicz/1987.csv' USING PigStorage(',') as (Year:int, Month:int, DayOfMonth:int, DayOfWeek:int, DepTime:int, CRSDepTime:int, ArrTime:int, CRSArrTime:int, UniqueCarrier:chararray, FlightNum:int, TailNum:chararray, ActualElapsedTime:int, CRSElapsedTime:int, AirTime:chararray, ArrDelay:int, DepDelay:int, Origin:chararray, Dest:chararray, Distance:int, TaxiIn:chararray, TaxiOut:chararray, Cancelled:int, CancellationCode:chararray, Diverted:int, CarrierDelay:int, WeatherDelay:int, NASDelay:int, SecurityDelay:int, LateAircraftDelay:int);
```

LIMIT - służy do uzyskania ograniczonej liczby krotek z relacji.

Składnia:

wynik = LIMIT relation_name required_number_of_tuples;

gdzie:

relation_name - to nazwa relacji w której chcemy ograniczoną ilość krotek.

required_number_of_tuples - to liczba krotek którą chcemy otrzymać.

Przykład:

```
poczatek = LIMIT dane_wejsciowe 20;
```

DUMP - służy do wyświetlania relacji na konsoli.

Składnia:

DUMP relation_name;

gdzie:

relation_name - to nazwa relacji którą chcemy wyświetlić.

Przykład:

```
DUMP poczatek;
```

DESCRIBE - służy do przeglądania schematu naszej relacji.

Składnia:

DESCRIBE relation_name;

gdzie:

relation_name - to nazwa relacji której schemat chcemy sprawdzić.

Przykład:

```
DESCRIBE wszystkie_loty;
```

FILTER - służy do wyboru wymaganych krotek z relacji na podstawie warunku.

Składnia:

wynik = FILTER relation_name BY condition;

gdzie:

relation_name - to nazwa relacji którą będziemy filtrować.

condition - to warunek który ma być spełniony.

Przykłady:

```
loty_na_lax = FILTER wszystkie_loty BY Dest == 'LAX ';

loty_po_godzinie_15 = FILTER wszystkie_loty BY DepTime > 1500;

loty_pomiedzy_15_a_17 = FILTER wszystkie_loty BY (DepTime > 1500) AND (DepTime < 1700);

loty_z_lotniska_na_litere_R = FILTER wszystkie_loty BY ( SUBSTRING(Origin, 0, 1) == 'R' );
```

STORE - służy do zapisywania relacji w określonym pliku.

Składnia:

STORE relation_name INTO 'output_directory_path' USING function;

gdzie:

relation_name - to nazwa relacji, którą będziemy zapisywać.

output_directory_path - to ścieżka do folderu w którym ma być zapisany wynik. Folder ten nie powinien istnieć.

function - to fukcja analogiczna z operatora LOAD i służy do zapisywania w odpowiednim formacie.

Przykłady:

```
STORE poczatek INTO 'danilewicz/wynik1';

STORE poczatek INTO 'danilewicz/wynik2' USING PigStorage(';');
```

GROUP - służy do grupowania danych w jedną lub więcej relacji. Gromadzi dane mające ten sam klucz.

Składnia:

wynik = GROUP relation_name BY key;

gdzie:

relation_name - to nazwa relacji, którą będziemy grupować.

key - to klucz (kolumna) po której będziemy grupować.

Przykłady:

```
pogrupowane_po_odlotach = GROUP wszystkie_loty BY Origin;

pogrupowane_po_lotniskach = GROUP wszystkie_loty BY (Origin, Dest);
```

FOREACH - służy do generowania określonych transformacji dla danej kolumny.

Składnia:

wynik = FOREACH relation_name GENERATE (required_data);

gdzie:

relation_name - to nazwa relacji, którą będziemy iterować.

required_data - to dane które chcemy otrzymać.

Przykłady:

wybrane_kolumny = FOREACH wszystkie_loty GENERATE DayOfWeek, Dest;

liczba_startow_z_lotniska = FOREACH pogrupowane_po_odlotach GENERATE group AS Lotnisko, COUNT(wszystkie_loty.Origin) AS LiczbaLotow;

ORDER BY - służy do wyświetlania zawartości relacji w uporządkowanej kolejności na podstawie jednego lub więcej pól.

Składnia:

wynik = ORDER relation_name BY key (ASC|DESC);

gdzie:

relation_name - to nazwa relacji, którą będziemy sortować.

key - to kolumna po której będziemy sortować.

Przykłady:

posortowane_po_liczbie_lotow = ORDER liczba_startow_z_lotniska BY LiczbaLotow DESC;

DISTINCT - służy do usuwania zbędnych (zduplikowanych) krotek z relacji.

Składnia:

wynik = DISTINCT relation_name;

gdzie:

relation_name - to nazwa relacji z której chcemy usunąć duplikaty.

Przykład:

```
dni_tygodnia = FOREACH wszystkie_loty GENERATE DayOfWeek;

unikalne_dni_tygodnia = DISTINCT dni_tygodnia;
```

JOIN - służy do łączenia rekordów z dwóch lub więcej relacji.

Składnia:

Inner Join:

wynik = JOIN relation_name_1 BY column_name_1, relation_name_2 BY column_name_2;

gdzie:

relation_name_1, relation_name_2 - to relacje które będziemy łączyć.

Column_name_1, column_name_2 - to kolumny po których będziemy łączyć.

Przykład:

```
legenda_dni_tygodnia = LOAD 'danilewicz/dni_tygodnia.csv ' USING PigStorage(', ') as (Nr:int,
Dzien:chararray);

wynik = JOIN wszystkie_loty BY DayOfWeek, dni_tygodnia BY Nr;
```

SPLIT - służy do podziału relacji na dwie lub więcej relacji.

Składnia:

SPLIT relation_name INTO relation_name_1 IF (condition_1), relation_name_2 IF (condition_2);

gdzie:

relation_name - to nazwa relacji, którą będziemy rozbijać.

relation_name_1 - to nazwa nowej relacji, która powstanie po spełnieniu warunku pierwszego.

condition_1 - to warunek który ma być spełniony dla relacji pierwszej.

Przykład:

```
SPLIT wszystkie_loty INTO loty_weekendowe IF (DayOfWeek == 6 OR DayOfWeek ==7),
pozostale_loty IF DayOfWeek < 6;
```

UNION - służy do łączenia dwóch relacji. Aby wykonać operację UNION na dwóch relacjach, ich kolumny i typy muszą być identyczne.

Składnia:

wynik = UNION relation_name_1, relation_name_2;

gdzie:

relation_name_1, relation_name2 - to nazwy relacji które chcemy połączyć.

Przykład:

```
dni_tygodnia1 = LOAD 'danilewicz/dni_tygodnia.csv ' USING PigStorage(', ') as (Nr:int,
Dzien:chararray);

dni_tygodnia2 = LOAD 'danilewicz/dni_tygodnia.csv ' USING PigStorage(', ') as (Nr:int,
Dzien:chararray);

wynik = UNION dni_tygodnia_1, dni_tygodnia_2;

loty = UNION loty_weekendowe, pozostale_loty;
```

Wykonywanie skryptu Pig z wiersza poleceń

- **w trybie lokalnym:**
  pig -x local moj_skrypt.pig
- **w trybie MapReduce:**
  pig -x mapreduce moj_skrypt.pig
- **w trybie Tez:**
  pig -x tez moj_skrypt.pig

Przykład:

pig -x mapreduce hdfs://sandbox-hdp.hortonworks.com:8020/tmp/.pigscripts/cwiczenie1.pig

Zadanie 1.

Policz liczbe lotów w poszczególnych miesiącach.

Odp.:
10, 448620
11, 422803
12, 440403

Zadanie 2.

Znajdź 3 lotniska z których startowało najwięcej lotów.

Odp.:
ORD, 67216
ATL, 66309
DFW, 51860

Zadanie 3.

Policz średnie spóźnienie w poszczególnych miesiącach.

Odp.:
10, 6.004
11, 8.467
12, 13.987

Zadanie 4.

Policz średnie spóźnienie w poszczególnych dniach tygodnia.

Odp.:
1, 9.42
2, 11.50
3, 11.23
4, 9.76
5, 8.79
6, 5.92
7, 9.15

Zadanie 5.

Znajdź przewoźnika który miał najwięcej odwołanych lotów w 1987 roku.

Odp.:
UA, 3436

Zadanie 6.

Znajdź odwołane loty w wigilie 1987 roku.

Odp.:

zad6.txt

Zadanie 7.

Znajdź lot z największym spóźnieniem w wigilie.

```
Odp.:
(1987,12,24,4,2310,1700,2309,1600,NW,257,NA,1499,1440,NA,429,370,PHX,SNA,33
8,NA,NA,0,NA,0,,,,,)
```

Zadanie 8.

Znajdź 3 przewodników, którzy mieli największy procent odwołanych lotów.

Odp.:
EA, 2.256
UA, 2.251
PS, 2.107

Zadanie 9.

Znajdź skąd dokąd odbywały się najdłuższe loty pod względem dystansu.

Odp.:
({(4983,JFK,HNL),(4983,HNL,JFK)})