## Operacje na HDFS-ie

| | |
|---|---|
| Kopiowanie pliku z lokalnego systemu plików do HDFS-a | hadoop fs -copyFromLocal ${src_local} ${dst_HDFS} |
| Usuwanie pliku na HDFS-ie | hadoop fs -rm ${to_remove_-on_HDFS} |
| Przenoszenie pliku w HDFS-ie | hadoop fs -mv ${src_HDFS} ${dst_HDFS} |
| Kopiowanie wyników z HDFS-a do lokalnego systemu plików | hadoop fs -getmerge ${src_-HDFS} {dst_local} |

Aby korzystać z Pig-a w trybie rozproszonym (wywołanie `pig` lub `pig -x mapreduce`) należy umieścić pliki wejściowe w HDFS-ie. Po dokonaniu przetworzenia można ponownie ściągnąć dane do lokalnego systemu plików

## Typy danych (Java/Pig)

| | |
|---|---|
| Integer | int |
| Long | long |
| Float | float |
| Double | double |
| String | chararray |
| Byte[] | bytearray |
| Boolean | boolean |
| Tuple | tuple:(sth) |
| Bag | bag:{tuples:(text:chararray)} |
| Map | map:['key'#value] |

Podstawowe typy danych używane w Pig-u i ich interpretacja w Javie

## Schemat danych

| | |
|---|---|
| Wczytanie danych bez wskazania schematu | A = load 'input_file.tsv'; |
| Wczytanie danych ze wskazaniem nazw kolumn (danym przypisany jest typ bytearray) | A = load 'input_file.tsv' as (col1, col2, col3); |
| Wczytanie danych ze wskazaniem pełnego schematu (nazwy kolumn i ich typy) | A = load 'input_file.tsv' as (col1:int, col2:chararray, col3:long); |
| j.w., inny przykład | A = load 'input_file.tsv' as (col1:int,col2:chararray,col3:-{tup:(f1:int, f2:long)}); |

## Schemat danych (cont)

| | |
|---|---|
| WERYFIKACJA schematu danych w tabeli A | describe A; |

Użytkownik wczytując dane może podać ich schemat.
Dzieje się tak kiedy podaje się nazwy kolumn, bądź też nazwy kolumn z konkretnymi typami.

## Produkcja danych

| | |
|---|---|
| Wypisanie tabeli A w konsoli | dump A; |
| Zapisanie tabeli A do pliku DST | store A into 'DST'; |

Aby sekwencja operacji - czy w interaktywnej konsoli, czy w wykonywanym skrypcie - została wykonana konieczne jest umieszczenie operacji produkującej wynik.
Operacje nie prowadzące do wyprodukowania wyniku są ignorowane.

## Wykorzystanie UDFow

| | |
|---|---|
| Zarejestruj plik JAR z lokalizacji wykonania skryptu | register 'myjar.jar' |
| Zarejestruj wszystie pliki JAR z lokalizacji wykonania skryptu | register '*.jar' |
| Zarejestruj wszystkie pliki JAR z lokalizacji wykonania skryptu oraz poziom wyżej | register '*./jar,../*.jar' |
| Użyj UDF-a z klasy pl.edu.example.TOLOWER | B = foreach A generate FLATTE-N(pl.edu.example.TOLOWER-(col1)) as lowerCol1; |
| Zarejestruj plik z UDFem Jythonowym | register judf.py using jython as judf; |
| Użyj UDF-a z zarejestrowanego skryptu judf.py | B = foreach A generate FLATTE-N(judf.TOLOWER(col1)) as lowerCol1; |

Pig umożliwia korzystanie ze zdefiniowanych w Javie, Pythonie, bądź w innym języku zdefiniowanych przez użytkownika funkcji (User Defined Functions, UDFs). Aby skorzystać z tej możliwości należy zarejestrować paczkę JAR, bądź skrypt Pythonowy, etc.

## Pythonowe UDFy

```
@outputSchema('chararray')
def TOLOWE R(i nStr):
  return inStr.l ower()
```