DLI Accelerated Data Science Teaching Kit

Lecture 10.6 - Pig and Hive

Pig

- High-level language
  - instead of writing low-level map and reduce functions
- Easier to program, understand and maintain
- Created at Yahoo!
- Produces sequences of Map-Reduce programs

Pig

Your program becomes a data flow sequence (i.e., data transformations).

Input ➡ data flow ➡ output

You specify data flow in Pig Latin (Pig's language). Then, Pig turns the data flow into a sequence of MapReduce jobs automatically!

Pig: 1st Benefit

- Save time and effort!
- Write only a few lines of Pig Latin
- Quite easy to learn

Pig: 2nd Benefit

- Pig can perform a sample run on representative subset of your input data automatically!
- Helps debug your code in smaller scale (much faster!), before applying on full data

What Pig is good for?

Batch processing

- Since it’s built on top of MapReduce
- Not for random query/read/write

May be **slower** than MapReduce programs coded from scratch

- You trade ease of use + coding time for some execution speed

How to Run Pig

- Pig is a client-side application (run on your computer)
- Nothing to install on Hadoop cluster

How to Run Pig: 2 modes

- Local Mode
  - Run on your computer (e.g., laptop)
  - Great for trying out Pig on small datasets
- MapReduce Mode
  - Pig translates your commands into MapReduce jobs
  - Remember you can have a single-machine cluster set up on your computer

Difference between PIG local and mapreduce mode:
http://stackoverflow.com/questions/11669394/difference-between-pig-local-and-mapreduce-mode

Pig program: 3 ways to write

- Script
- **Grunt** (interactive shell)
  - Great for debugging
  - Provides code completion (press Tab key)
- Embedded (into Java program)
  - Use PigServer class (like JDBC for SQL)
  - Use PigRunner to access Grunt

Find Highest Temperature by Year

```
grunt>
records = LOAD 'input/ncdc/micro-tab/sample.txt'
  AS (year:chararray, temperature:int, quality:int);
```

```
grunt> DUMP records;
```

```
(1950,0,1)
(1950,22,1)
(1950,-11,1)
(1949,111,1)
(1949,78,1)
```

called a "tuple"

```
grunt> DESCRIBE records;
```

```
records: {year: chararray, temperature: int, quality: int}
```

Find Highest Temperature by Year

```
grunt>
filtered_records =
  FILTER records BY temperature != 9999
  AND (quality == 0 OR quality == 1 OR
       quality == 4 OR quality == 5 OR
       quality == 9);

grunt> DUMP filtered_records;
```

```
(1950,0,1)
(1950,22,1)
(1950,-11,1)
(1949,111,1)
(1949,78,1)
```

In this example, no tuple is filtered out

Find Highest Temperature by Year

```
grunt> grouped_records = GROUP filtered_records BY year;

grunt> DUMP grouped_records;
```

```
(1949,{(1949,111,1), (1949,78,1)})
(1950,{(1950,0,1),(1950,22,1),(1950,-11,1)})
```

called a "bag"

= unordered collection of tuples

```
grunt> DESCRIBE grouped_records;
```

alias that Pig created

```
grouped_records: {group: chararray, filtered_records:
{year: chararray, temperature: int, quality: int}}
```

Find Highest Temperature by Year

```
(1949,{(1949,111,1), (1949,78,1)})
(1950,{(1950,0,1),(1950,22,1),(1950,-11,1)})
```

```
grouped_records: {group: chararray, filtered_records: {year:
chararray, temperature: int, quality: int}}
```

```
grunt> max_temp = FOREACH grouped_records GENERATE
  group, MAX(filtered_records.temperature);

grunt> DUMP max_temp;
```

```
(1949,111)
(1950,22)
```

Find Highest Temperature by Year

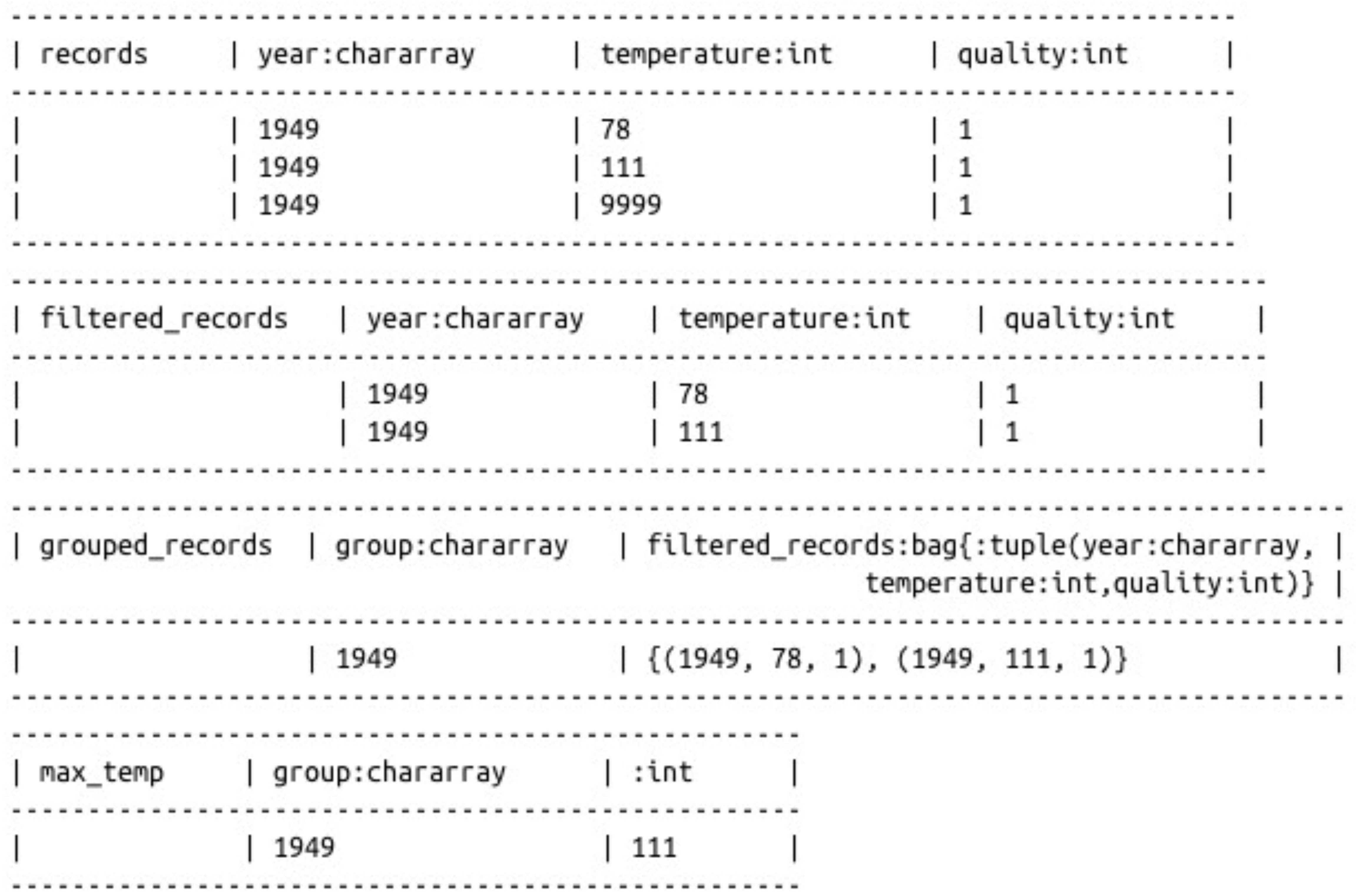
```
records = LOAD 'input/ ncdc/ micro-tab/ sample.txt'
  AS (year:chararray, temperature:int, quality:int);

filtered_records =
  FILTER records BY temperature != 9999
  AND (quality = = 0 OR quality = = 1 OR
       quality = = 4 OR quality = = 5 OR
       quality = = 9);

grouped_records = GROUP filtered_records BY year;

max_temp = FOREACH grouped_records GENERATE
  group, MAX(filtered_records.temperature);

DUMP max_temp;
```

Run Pig on a Subset of Your Data

- You saw an example run on a tiny dataset
- How to test your program on a larger dataset, without having to wait for a long time?
    - Use the ILLUSTRATE command to generate sample dataset

Run Pig on a Subset of Your Data

```
grunt> ILLUSTRATE max_temp;
```

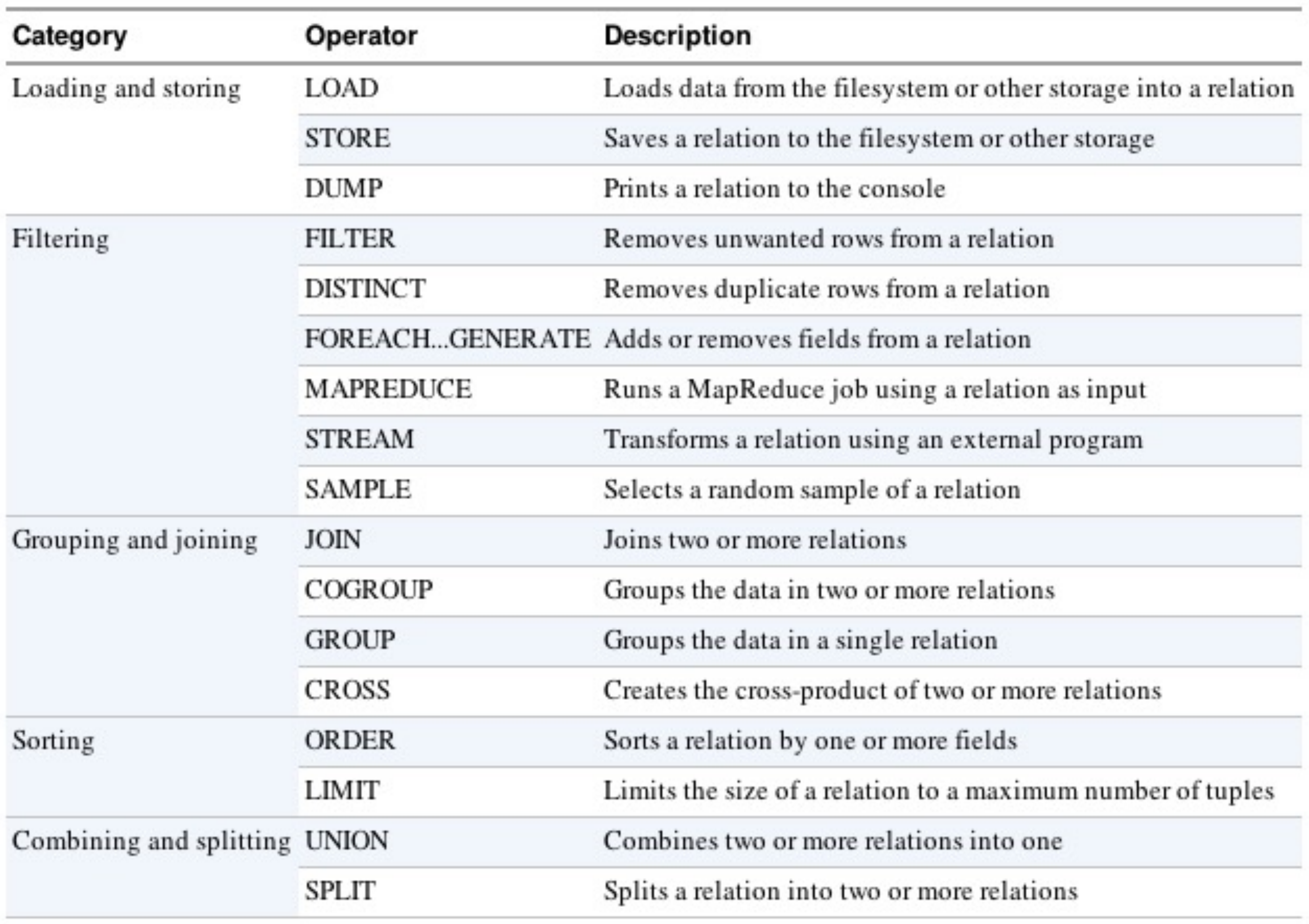


```
-------------------------------------------------------------------------------
| records       | year:chararray       | temperature:int       | quality:int      |
-------------------------------------------------------------------------------
|               | 1949                 | 78                    | 1                |
|               | 1949                 | 111                   | 1                |
|               | 1949                 | 9999                  | 1                |
-------------------------------------------------------------------------------

---------------------------------------------------------------------------------
| filtered_records    | year:chararray     | temperature:int     | quality:int     |
---------------------------------------------------------------------------------
|                     | 1949               | 78                  | 1               |
|                     | 1949               | 111                 | 1               |
---------------------------------------------------------------------------------

--------------------------------------------------------------------------------------
| grouped_records   | group:chararray    | filtered_records:bag{:tuple(year:chararray, |
                                                      temperature:int,quality:int)} |
--------------------------------------------------------------------------------------
|                   | 1949               | {(1949, 78, 1), (1949, 111, 1)}           |
--------------------------------------------------------------------------------------

---------------------------------------------------
| max_temp       | group:chararray       | :int       |
---------------------------------------------------
|                | 1949                  | 111        |
---------------------------------------------------
```

Much More to Learn About Pig

Relational Operators, Diagnostic Operators (e.g., describe, explain, illustrate), utility commands (cat, cd, kill, exec), etc.

Table 11-1. Pig Latin relational operators



Category	Operator	Description
Loading and storing	LOAD	Loads data from the filesystem or other storage into a relation
	STORE	Saves a relation to the filesystem or other storage
	DUMP	Prints a relation to the console
Filtering	FILTER	Removes unwanted rows from a relation
	DISTINCT	Removes duplicate rows from a relation
	FOREACH...GENERATE	Adds or removes fields from a relation
	MAPREDUCE	Runs a MapReduce job using a relation as input
	STREAM	Transforms a relation using an external program
	SAMPLE	Selects a random sample of a relation
Grouping and joining	JOIN	Joins two or more relations
	COGROUP	Groups the data in two or more relations
	GROUP	Groups the data in a single relation
	CROSS	Creates the cross-product of two or more relations
Sorting	ORDER	Sorts a relation by one or more fields
	LIMIT	Limits the size of a relation to a maximum number of tuples
Combining and splitting	UNION	Combines two or more relations into one
	SPLIT	Splits a relation into two or more relations

Hive

http://hive.apache.org

Use SQL to run queries on large datasets

Developed at Facebook

Similar to Pig, Hive runs on client computer that submit jobs (no need to install on Hadoop cluster)

- You write HiveQL (Hive's query language), which gets converted into MapReduce jobs

Example: create table, load data

```
CREATE TABLE records (year STRING, temperature INT, quality INT)
ROW FORMAT DELIMITED
   FIELDS TERMINATED BY ‘\t’;
```

Specify that data file is tab-separated

```
LOAD DATA LOCAL INPATH 'input/ncdc/micro-tab/sample.txt'
OVERWRITE INTO TABLE records;
```

Overwrite old file

This data file will be copied to Hive’s internal data directory

Example: Query

```
hive> SELECT year, MAX(temperature)
    > FROM records
    > WHERE temperature != 9999
    >    AND (quality =0 OR quality =1 OR quality =4 OR
quality = 5 OR quality = 9)
    > GROUP BY year;
1949  111
1950  22
```

Find Highest Temperature by Year

```
hive> SELECT year, MAX(temperature)
    > FROM records
    > WHERE temperature != 9999
    >    AND (quality =0 OR quality =1 OR quality =4 OR
quality = 5 OR quality = 9)
    > GROUP BY year;
1949  111
1950  22
```

Same Thing Done using Pig

```
records = LOAD 'input/ ncdc/ micro-tab/ sample.txt'
  AS (year:chararray, temperature:int, quality:int);

filtered_records =
  FILTER records BY temperature != 9999
  AND (quality = = 0 OR quality = = 1 OR
       quality = = 4 OR quality = = 5 OR
       quality = = 9);

grouped_records = GROUP filtered_records BY year;

max_temp = FOREACH grouped_records GENERATE
  group, MAX(filtered_records.temperature);

DUMP max_temp;
```

Hive (~SQL) vs Pig

1. Pig is procedural (SQL is declarative)
2. Checkpointing data in the pipeline
3. Use specific operator implementations vs. relying on optimizer
4. Splitting pipeline
   e.g., do multiple things to intermediate data
5. Use developer’s own code
   e.g., different ways of loading data

DLI Accelerated Data Science Teaching Kit

Thank You