PIG DETAILS

Contents

Pig Latin Relational Operators

Category	Operator	Description
Loading and Storing	LOAD	Loads data from the filesystem or otherstorage into a relation
	STORE	Saves a relation to the filesystem or other storage
	DUMP	Prints a relation to the console
Filtering	FILTER	Removes nwanted 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 a 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

Pig Latin diagnostic operators

Operator	Descriptions
DESCRIBE	Prints a relation's schema
EXPLAIN	Prints the logical and physical plans
ILLUSTRATE	Shows a sample execution of the logical plan, using a generated subset of input

Pig Latin macro and UDF statements

Statement	Descriptions
REGISTER	registers a JAR file with the PIG runtime
DEFINE	Creates an alias for a macro, a UDF, streaming script, or a command specification
IMPORT	Import macros defined in a separate file into a script

Pig Latin Commands

Category	Command	Description
Hadoop Filesystem	cat	Prints the contents of one or more files
	copyFromLocal or put	Copies a local file or directory to a Hadoop Filesystem
	copyToLocal or get	Copies a file or directory on a Hadoop filesystem to the local filesystem
	cp	Copies file or directory to another directory
	fs	Access Hadoop's filesystem shell
	ls	List Files
	mkdir	Creates a new directory
	mv	Moves a file or directory to another directory
	rm	Deletes file or directory
	rmf	Forcibly deletes a file or directory
Hadoop Mapreduce	kill	Kills a MapReduce job
	exec	Runs a script in a new Grunt shell in a batch mode
	help	Shows the available commands and options
	quit	exits the interpreter
	run	Runs the script within the existing Grunt shell
	set	Sets Pig options and MapReduce job properties
	sh	Run a shell command from within Grunt

Pig Latin Types

Category	Type	Description	Literal Example
Numeric	int	32-bit signed integer	1
	long	64-bit signed integer	1L
	float	32-bit floating point number	1.0F
	double	64-bit floating point number	1.0
Text	Chararray	Character Array in UTF-16 format	('a')
Binary	bytearray	Byte Array	Not Supported
Complex	tuple	sequence of fields of any type	(1, 'pomegranate')
	bag	An unordered collection of tuples, possibly with duplicates	{(1, 'pomegranate'), (2)}
	map	A set of key value pairs	['a'#'pomegranate']

Pig Example

Writing a program to calculate the maximum temperature recorded by the year for the weather dataset.

PIG Latin

```
-- max_temp.pig: Finds the maximum 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;
```

Java

MAPPER

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MaxTemperatureMapper
	extends Mapper<LongWritable, Text, Text, IntWritable> {
	private static final int MISSING = 9999;
	@Override
	public void map(LongWritable key, Text value, Context context)
			throws IOException, InterruptedException {
		String line = value.toString();
		String year = line.substring(15, 19);
		int airTemperature;
		if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
			airTemperature = Integer.parseInt(line.substring(88, 92));
		} else {
			airTemperature = Integer.parseInt(line.substring(87, 92));
			}
			String quality = line.substring(92, 93);
			if (airTemperature != MISSING && quality.matches("[01459]")) {
			context.write(new Text(year), new IntWritable(airTemperature));
			}
		}
	}
```

REDUCER

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class MaxTemperatureReducer
	extends Reducer<Text, IntWritable, Text, IntWritable> {
	@Override
	public void reduce(Text key, Iterable<IntWritable> values,
		Context context)
		throws IOException, InterruptedException {

	int maxValue = Integer.MIN_VALUE;
	for (IntWritable value : values) {
		maxValue = Math.max(maxValue, value.get());
	}
	context.write(key, new IntWritable(maxValue));
}
}
```

RUN THE MAPREDUCE

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MaxTemperature {

        public static void main(String[] args) throws Exception {
        if (args.length != 2) {
        System.err.println("Usage: MaxTemperature <input path> <output path>");
        System.exit(-1);
        }
        Job job = new Job();
        job.setJarByClass(MaxTemperature.class);
        job.setJobName("Max temperature");
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setMapperClass(MaxTemperatureMapper.class);
        job.setReducerClass(MaxTemperatureReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```