Dynamic Partitioning

Table of contents

1 Overview

When writing data in HCatalog it is possible to write all records to a single partition. In this case the partition column(s) need not be in the output data.

The following Pig script illustrates this:

```
A = load 'raw' using HCatLoader();
...
split Z into for_us if region='us', for_eu if region='eu', for_asia if region='asia';
store for_us into 'processed' using HCatStorer("ds=20110110, region=us");
store for_eu into 'processed' using HCatStorer("ds=20110110, region=eu");
store for_asia into 'processed' using HCatStorer("ds=20110110, region=asia");
```

In cases where you want to write data to multiple partitions simultaneously, this can be done by placing partition columns in the data and not specifying partition values when storing the data.

```
A = load 'raw' using HCatLoader();
...
store Z into 'processed' using HCatStorer();
```

The way dynamic partitioning works is that HCatalog locates partition columns in the data passed to it and uses the data in these columns to split the rows across multiple partitions. (The data passed to HCatalog **must** have a schema that matches the schema of the destination table and hence should always contain partition columns.) It is important to note that partition columns can’t contain null values or the whole process will fail.

It is also important to note that all partitions created during a single run are part of one transaction; therefore if any part of the process fails, none of the partitions will be added to the table.

2 Usage with Pig

Usage from Pig is very simple! Instead of specifying all keys as one normally does for a store, users can specify the keys that are actually needed. HCatOutputFormat will trigger on dynamic partitioning usage if necessary (if a key value is not specified) and will inspect the data to write it out appropriately.

So this statement...

```
store A into 'mytable' using HCatStorer("a=1, b=1");
```

...is equivalent to any of the following statements, if the data has only values where a=1 and b=1:

```
store A into 'mytable' using HCatStorer();

store A into 'mytable' using HCatStorer("a=1");

store A into 'mytable' using HCatStorer("b=1");
```

On the other hand, if there is data that spans more than one partition, then HCatOutputFormat will automatically figure out how to spray the data appropriately.

For example, let's say a=1 for all values across our dataset and b takes the values 1 and 2. Then the following statement...

```
store A into 'mytable' using HCatStorer();
```

...is equivalent to either of these statements:

```
store A into 'mytable' using HCatStorer("a=1");

split A into A1 if b='1', A2 if b='2';
store A1 into 'mytable' using HCatStorer("a=1, b=1");
store A2 into 'mytable' using HCatStorer("a=1, b=2");
```

3 Usage from MapReduce

As with Pig, the only change in dynamic partitioning that a MapReduce programmer sees is that they don't have to specify all the partition key/value combinations.

A current code example for writing out a specific partition for (a=1, b=1) would go something like this:

```
Map<String, String> partitionValues = new HashMap<String, String>();
partitionValues.put("a", "1");
partitionValues.put("b", "1");
HCatTableInfo info = HCatTableInfo.getOutputTableInfo(dbName, tblName, partitionValues);
HCatOutputFormat.setOutput(job, info);
```

And to write to multiple partitions, separate jobs will have to be kicked off with each of the above.

With dynamic partitioning, we simply specify only as many keys as we know about, or as required. It will figure out the rest of the keys by itself and spray out necessary partitions, being able to create multiple partitions with a single job.