WHAT IS APACHE HCATALOG? HCATALOG EXPLAINED

Here we explain what HCatalog is and why it is useful to Hadoop programmers.

Basically, HCatalog provides a consistent interface between Apache Hive, Apache Pig, and MapReduce. Since it ships with Hive, you could consider it an extension of Hive.

(We have written tutorials here on Apache Pig, MapReduce, and Hive.)

Why this Matters

To understand why this is important, consider that, for example, MapReduce programs run a map and optional reduce operation over data values to produce **key->value** pairs. Apache Pig creates **tuples** of data. And data stored in Hive is in **table records**, just like a relational database. So the three have different ways of storing data. So they are not so easy to use together.

For example, an Apache Pig tuple can look like this:

(1, {(1,2,3,4,5), (1,2,3,4,5)})

MapReduce stores data in key->value pairs, like this:

{
name -> Walker Rowe,
Position -> freelance technical writer
}

An Apache Hive table looks just like an RDBMS table created with a SQL command:

col_name	data_type
station	string
station_name	string
wdate	date
prcp	float
wind	float
snow	float

Using Hive Metadata

How is HCatalog useful? One example is clear right away.

Once you use HCatalog then you no longer have to use file paths or even the schema as Hive knows all about that. For example, the Pig statement below loads a file from a Hadoop file system and specifies the schema.

```
a = LOAD 'hdfs://localhost:9000/user/hadoop/sales.csv' USING PigStorage(',') AS (shop:chararray,employee:chararray,sales:int);
```

With HCatalog that becomes the far simpler:

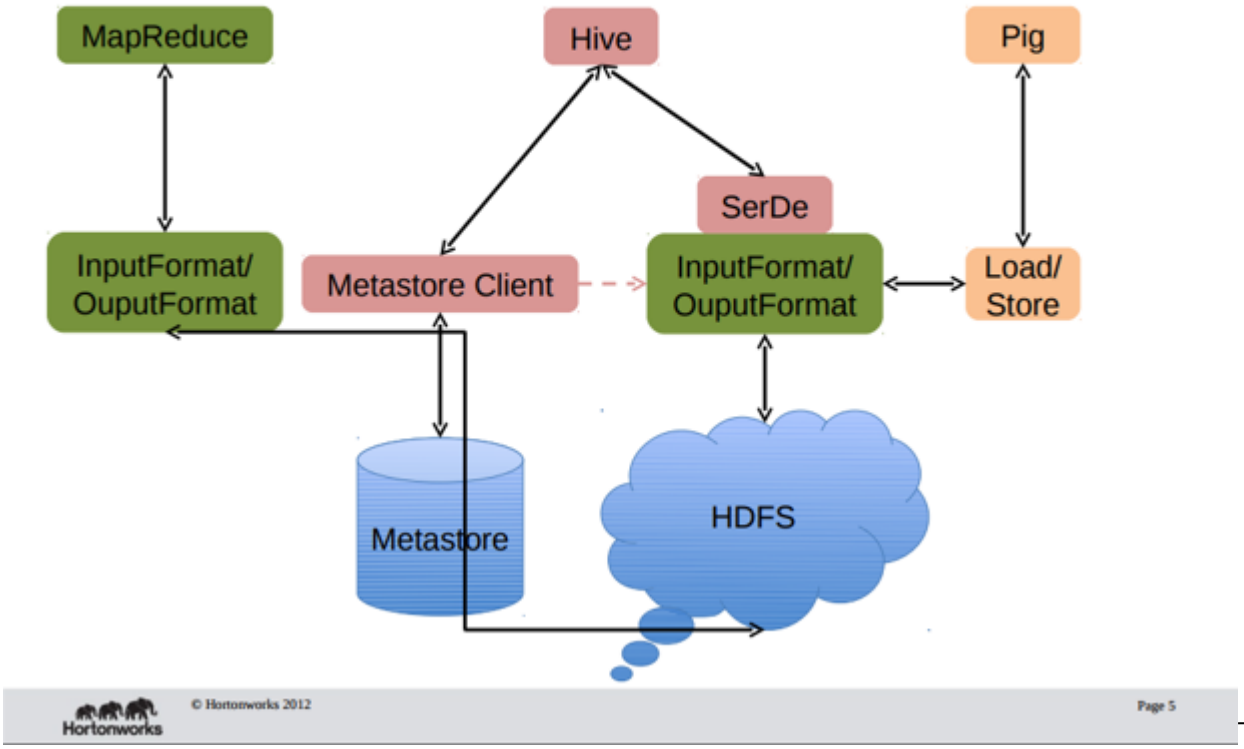
```
a = LOAD 'sales.csv' using HCatLoader();
```

So how does a MapReduce programmer use that. Without HCatalog, they would have to write a program to consume Apache Pig tuples stored on the Hadoop file system.

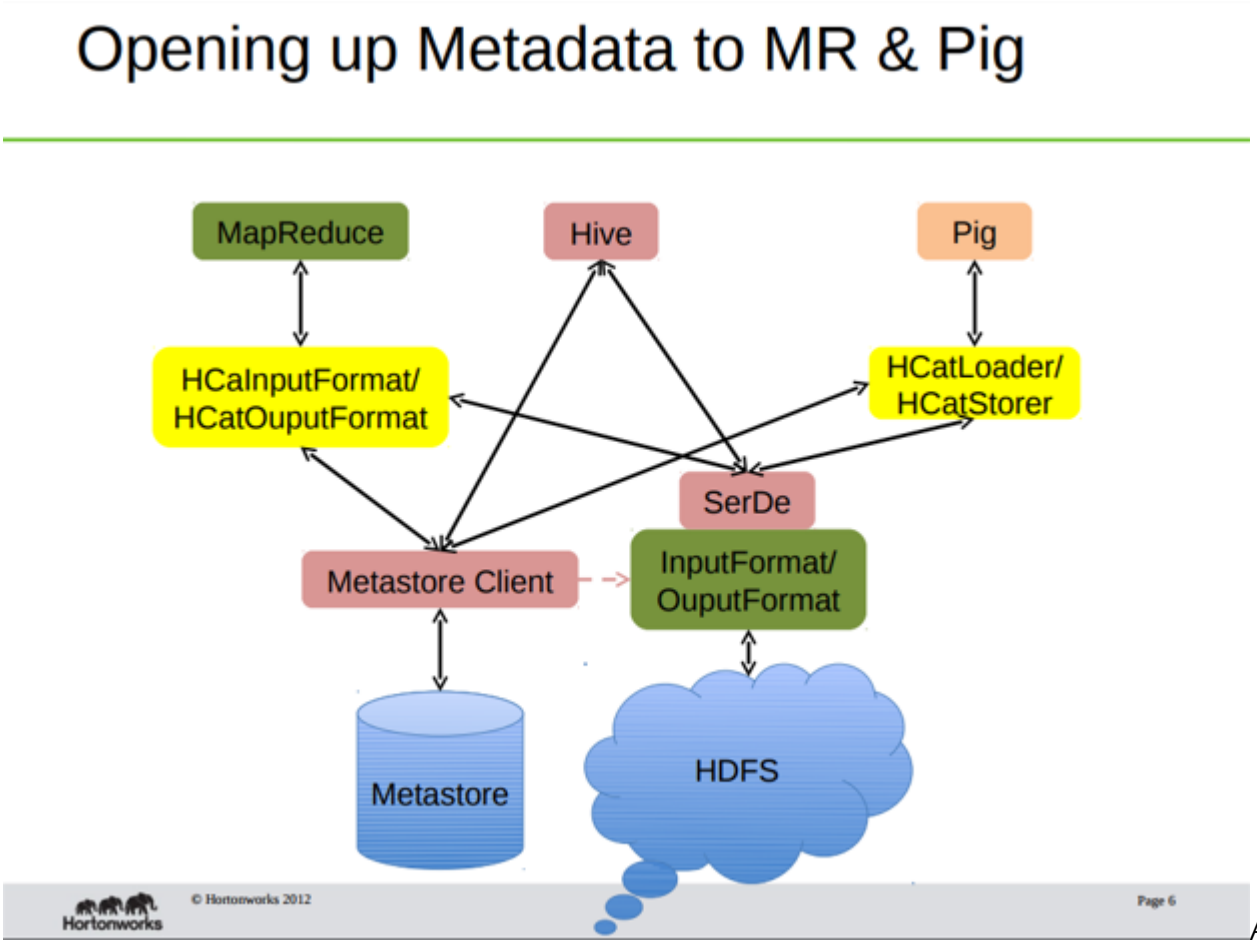
The goal of HCatalog is to allow Pig and MapReduce to be able to use the same data structures as Hive. Then there is no need to convert data.

This concept is best visualized in these two graphic from HortonWorks

The first shows that all three products use Hadoop to store data. Hive stores its metadata (i.e., schema) in MySQL or Derby. The other two do that using code written into the programs and input and output operations.



The second graphic shows that HCatalog exposes Hive data and metadata to MapReduce and Pig directly. This is done using the interfaces shown in yellow. The end result is that the user can work with Hive tables as if they were MapReduce key->value pairs or Pig tuples.



All of this supports sharing data between programs and programmers too. For example, data can be shared with other programs as a REST web service as HCatalog exposes that. And when an Hcatalog task finishes, it can create a JMS message to signal an Apache Pig program to run.

Interface Abstraction

In the words of Apache HCatalog, "HCatalog supports reading and writing files in any format for which a SerDe (serializer-deserializer) can be written." That means in addition to your own CSV, JSON, RCFile, and SequenceFile, and ORC file formats you could write your own. For example, here is a discussion of how to do that with Apache Hive.

Using HCatalog and Apache Pig

You can run Pig like this to tell it to use HCatalog.

```
pig -useHCatalog
```

Then it uses what is called **HCatLoader** to work with data managed by HCatalog. But to do this you need to set the PIG_CLASSPATH and PIG_OPTS environment variables to tell Pig where to find the HCatalog tables.

MapReduce

MapReduce code is usually Java code.

HCatalog **HCatLoader** and **HCatStore** are implementations of the Hadoop **InputFormat** and

OutputFormat interfaces. That gives you **org.apache.hadoop.mapreduce.RecordReader** and **org.apache.pig.backend.hadoop.executionengine.mapReduceLayer** to read data from Hive and run the MapReduce operations over that. Then you save your results back into Hive.

HCatalog Partitions and the Hive Column-oriented Database

If you are familiar with Apache Cassandra, which we wrote about here, then you know that is a column-oriented database. HCatalog does the same thing by letting you create **partitions**. The whole point with column-oriented databases is you can group common fields on the same storage for fast retrieval. That makes a lot more sense when you want to:

```
select one_field from table;
```

Instead of retrieving the whole table, you just retrieves columns that you need, boosting speed, since the data is next to each other, and saving memory too. HCatalog does that by letting you divide tables into files called partitions as well. So you can take advantage of this with Pig and MapReduce by reading data much faster than you would with Pig LOAD or reading files using a **buffered stream** from Hadoop.

Installation and using the HCatalog CLI

To use HCatalog, first install Hadoop and Hive. See the instructions on the Hadoop and Hive web sites for that. It will take a while. My advise is to use MySQL instead of Apache Derby for the Hive installation as many users have complained on StackOverflow about the difficulty of getting Derby to work. That has been my experience is well.

Hcatalog is installed with Hive. So there is nothing to do to use it except:

```
export PATH=$PATH:$HIVE_HOME/hcatalog/bin
export HCAT_HOME=$HIVE_HOME/hcatalog
```

Now run **hcat**. It should echo some command line options and then return the command prompt.

Now you can run Hive DDL and SQL commands from the command prompt, like this command:

```
hcat -e 'show tables';
```

In this case, it will show you the **weather** table that we created in the article **Apache Hive Beeline Client, Import CSV File into Hive**.

```
hcat -e 'describe weather';
```

station	string
station_name	string
wdate	string
prcp	float
wind	int
snow	int