HBase via Hive, Part 2

[](http://hive.apache.org/)

This is the second of two posts examining the use of Hive for interaction with HBase tables. This is a hands-on exploration so the [first post](http://www.n10k.com/blog/hbase-via-hive-pt1/) isn’t required reading for consuming this one. Still, it might be good context.

“Nick!” you exclaim, “that first post had too many words and I don’t care about JIRA tickets. Show me how I use this thing!”

This is post is exactly that: a concrete, end-to-end example of consuming HBase over Hive. The whole mess was tested to work on a tiny little 5-node cluster running HDP-1.3.2, which means Hive 0.11.0 and HBase 0.94.6.1.

**Grab some data and register it in Hive**

We’ll need some data to work with. For this purpose, grab some [traffic stats](http://www.mediawiki.org/wiki/Analytics/Wikistats) from wikipedia. Once we have some data, copy it up to HDFS.

|  |
| --- |
| $ mkdir pagecounts ; cd pagecounts  $ for x in {0..9} ; do wget "http://dumps.wikimedia.org/other/pagecounts-raw/2008/2008-10/pagecounts-20081001-0${x}0000.gz" ; done  $ hadoop fs -copyFromLocal $(pwd) ./ |

For reference, this is what the data looks like.

|  |
| --- |
| $ zcat pagecounts-20081001-000000.gz | head -n5  aa.b Special:Statistics 1 837  aa Main\_Page 4 41431  aa Special:ListUsers 1 5555  aa Special:Listusers 1 1052  aa Special:PrefixIndex/Comparison\_of\_Guaze%27s\_Law\_and\_Coulomb%27s\_Law 1 4332 |

As I understand it, each record is a count of page views of a specific page on [Wikipedia](http://www.wikipedia.org/). The first column is the language code, second is the page name, third is the number of page views, and fourth is the size of the page in bytes. Each file contains an hour’s worth of aggregated data. None of the above pages were particularly popular that hour.

Now that we have data and understand its raw schema, create a Hive table over it. To do that, we’ll use a DDL script that looks like this.

|  |
| --- |
| $ cat 00\_pagecounts.ddl  -- define an external table over raw pagecounts data  CREATE TABLE IF NOT EXISTS pagecounts (projectcode STRING, pagename STRING, pageviews STRING, bytes STRING)  ROW FORMAT  DELIMITED FIELDS TERMINATED BY ' '  LINES TERMINATED BY '\n'  STORED AS TEXTFILE  LOCATION '/user/ndimiduk/pagecounts'; |

Run the script to register our dataset with Hive.

|  |
| --- |
| $ hive -f 00\_pagecounts.ddl  OK  Time taken: 2.268 seconds |

Verify that the schema mapping works by calculating a simple statistic over the dataset.

|  |
| --- |
| $ hive -e "SELECT count(\*) FROM pagecounts;"  Total MapReduce jobs = 1  Launching Job 1 out of 1  ...  OK  36668549  Time taken: 25.31 seconds, Fetched: 1 row(s) |

Hive says the 10 files we downloaded contain just over 36.5mm records. Let’s just confirm things are working as expected by getting a second opinion. This isn’t that much data, so confirm on the command line.

|  |
| --- |
| $ zcat \* | wc -l  36668549 |

The record counts match up – excellent.

**Transform the schema for HBase**

The next step is to transform the raw data into a schema that makes sense for HBase. In our case, we’ll create a schema that allows us to calculate aggregate summaries of pages according to their titles. To do this, we want all the data for a single page grouped together. We’ll manage that by creating a Hive view that represents our target HBase schema. Here’s the DDL.

|  |
| --- |
| $ cat 01\_pgc.ddl  -- create a view, building a custom hbase rowkey  CREATE VIEW IF NOT EXISTS pgc (rowkey, pageviews, bytes) AS  SELECT concat\_ws('/',  projectcode,  concat\_ws('/',  pagename,  regexp\_extract(INPUT\_\_FILE\_\_NAME, 'pagecounts-(\\d{8}-\\d{6})\\..\*$', 1))),  pageviews, bytes  FROM pagecounts; |

The SELECT statement uses hive to build a compound rowkey for HBase. It concatenates the project code, page name, and date, joined by the '/' character. A handy trick: it uses a simple regex to extract the date from the source file names. Run it now.

|  |
| --- |
| $ hive -f 01\_pgc.ddl  OK  Time taken: 2.712 seconds |

This is just a view, so the SELECT statement won’t be evaluated until we query it for data. Registering it with hive doesn’t actually process any data. Again, make sure it works by querying Hive for a subset of the data.

|  |
| --- |
| $ hive -e "SELECT \* FROM pgc WHERE rowkey LIKE 'en/q%' LIMIT 10;"  Total MapReduce jobs = 1  Launching Job 1 out of 1  ...  OK  en/q:Special:Search/Blues/20081001-090000 1 1168  en/q:Special:Search/rock/20081001-090000 1 985  en/qadam\_rasul/20081001-090000 1 1108  en/qarqay/20081001-090000 1 933  en/qemu/20081001-090000 1 1144  en/qian\_lin/20081001-090000 1 918  en/qiang\_(spear)/20081001-090000 1 973  en/qin\_dynasty/20081001-090000 1 1120  en/qinghe\_special\_steel\_corporation\_disaster/20081001-090000 1 963  en/qmail/20081001-090000 1 1146  Time taken: 40.382 seconds, Fetched: 10 row(s) |

**Register the HBase table**

Now that we have a dataset in Hive, it’s time to introduce HBase. The first step is to register our HBase table in Hive so that we can interact with it using Hive queries. That means another DDL statement. Here’s what it looks like.

|  |
| --- |
| $ cat 02\_pagecounts\_hbase.ddl  -- create a table in hbase to host the view  CREATE TABLE IF NOT EXISTS pagecounts\_hbase (rowkey STRING, pageviews STRING, bytes STRING)  STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'  WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,f:c1,f:c2')  TBLPROPERTIES ('hbase.table.name' = 'pagecounts'); |

This statement will tell Hive to go create an HBase table named pagecounts with the single column family f. It registers that HBase table in the Hive metastore by the name pagecounts\_hbase with 3 columns: rowkey, pageviews, and bytes. The SerDe property hbase.columns.mapping makes the association from Hive column to HBase column. It says the Hive column rowkey is mapped to the HBase table’s rowkey, the Hive column pageviews to the HBase column f:c1, and bytes to the HBase column f:c2. To keep the example simple, we have Hive treat all these columns as the STRING type.

In order to use the HBase library, we need to make the HBase jars and configuration available to the local Hive process (at least until [HIVE-5518](https://issues.apache.org/jira/browse/HIVE-5518) is resolved). Do that by specifying a value for the HADOOP\_CLASSPATH environment variable before executing the statement.

|  |
| --- |
| $ export HADOOP\_CLASSPATH=/etc/hbase/conf:/usr/lib/hbase/hbase-0.94.6.1.3.2.0-111-security.jar:/usr/lib/zookeeper/zookeeper.jar  $ hive -f 02\_pagecounts\_hbase.ddl  OK  Time taken: 4.399 seconds |

**Populate the HBase table**

Now it’s time to write data to HBase. This is done using a regular Hive INSERT statement, sourcing data from the view with SELECT. There’s one more bit of administration we need to take care of though. This INSERT statement will run a mapreduce job that writes data to HBase. That means we need to tell Hive to ship the HBase jars and dependencies with the job.

*Note* that this is a separate step from the classpath modification we did previously. Normally you can do this with an export statement from the shell, the same way we specified the HADOOP\_CLASSPATH. However there’s a bug in HDP-1.3 that requires me to use Hive’s SET statement in the script instead.

|  |
| --- |
| $ cat 03\_populate\_hbase.hql  -- ensure hbase dependency jars are shipped with the MR job  -- Should export HIVE\_AUX\_JARS\_PATH but this is broken in HDP-1.3.x  SET hive.aux.jars.path = file:///etc/hbase/conf/hbase-site.xml,file:///usr/lib/hive/lib/hive-hbase-handler-0.11.0.1.3.2.0-111.jar,file:///usr/lib/hbase/hbase-0.94.6.1.3.2.0-111-security.jar,file:///usr/lib/zookeeper/zookeeper-3.4.5.1.3.2.0-111.jar;  -- populate our hbase table  FROM pgc INSERT INTO TABLE pagecounts\_hbase SELECT pgc.\* WHERE rowkey LIKE 'en/q%' LIMIT 10; |

*Note* there’s a big ugly [bug](https://issues.apache.org/jira/browse/HIVE-5515) in Hive 0.12.0 which means this doesn’t work with that version. Never fear though, we have a patch in progress. Follow along at [HIVE-5515](https://issues.apache.org/jira/browse/HIVE-5515).

If you choose to use a different method for setting Hive’s auxpath, be advised that it’s a tricky process – depending on how you specify it (HIVE\_AUX\_JARS\_PATH, --auxpath), Hive will interpret the argument differently. [HIVE-2349](https://issues.apache.org/jira/browse/HIVE-2349)seeks to remedy this unfortunate state of  affairs.

|  |
| --- |
| $ hive -f 03\_populate\_hbase.hql  Total MapReduce jobs = 1  Launching Job 1 out of 1  Number of reduce tasks determined at compile time: 1  ...  OK  Time taken: 40.296 seconds |

Be advised also that this step is currently broken on secured HBase deployments. Follow along [HIVE-5523](https://issues.apache.org/jira/browse/HIVE-5523) if that’s of interest to you.

**Query data from HBase-land**

40 seconds later, you now have data in HBase. Let’s have a look using the HBase shell.

|  |
| --- |
| $ echo "scan 'pagecounts'" | hbase shell  HBase Shell; enter 'help<RETURN>' for list of supported commands.  Type "exit<RETURN>" to leave the HBase Shell  Version 0.94.6.1.3.2.0-111, r410a7a1c151ca953553eae68aa84e2a9f0d6e4ca, Mon Aug 19 19:00:12 PDT 2013  scan 'pagecounts'  ROW COLUMN+CELL  en/q:Pan%27s\_Labyrinth/20081001-080000 column=f:c1, timestamp=1381534232485, value=1  en/q:Pan%27s\_Labyrinth/20081001-080000 column=f:c2, timestamp=1381534232485, value=1153  en/q:Special:Search/Jazz/20081001-080000 column=f:c1, timestamp=1381534232485, value=1  en/q:Special:Search/Jazz/20081001-080000 column=f:c2, timestamp=1381534232485, value=980  en/q:Special:Search/peinture/20081001-080000 column=f:c1, timestamp=1381534232485, value=1  en/q:Special:Search/peinture/20081001-080000 column=f:c2, timestamp=1381534232485, value=989  en/q:Special:Search/rock/20081001-080000 column=f:c1, timestamp=1381534232485, value=1  en/q:Special:Search/rock/20081001-080000 column=f:c2, timestamp=1381534232485, value=980  en/qadi/20081001-080000 column=f:c1, timestamp=1381534232485, value=1  en/qadi/20081001-080000 column=f:c2, timestamp=1381534232485, value=1112  en/qalawun%20complex/20081001-080000 column=f:c1, timestamp=1381534232485, value=1  en/qalawun%20complex/20081001-080000 column=f:c2, timestamp=1381534232485, value=942  en/qalawun/20081001-080000 column=f:c1, timestamp=1381534232485, value=1  en/qalawun/20081001-080000 column=f:c2, timestamp=1381534232485, value=929  en/qari'/20081001-080000 column=f:c1, timestamp=1381534232485, value=1  en/qari'/20081001-080000 column=f:c2, timestamp=1381534232485, value=929  en/qasvin/20081001-080000 column=f:c1, timestamp=1381534232485, value=1  en/qasvin/20081001-080000 column=f:c2, timestamp=1381534232485, value=921  en/qemu/20081001-080000 column=f:c1, timestamp=1381534232485, value=1  en/qemu/20081001-080000 column=f:c2, timestamp=1381534232485, value=1157  10 row(s) in 0.4960 seconds |

Here we have 10 rows with two columns each containing the data loaded using Hive. It’s now accessible in your online world using HBase. For example, perhaps you receive an updated data file and have a corrected value for one of the stats. You can update the record in HBase with a regular PUT command.

**Verify data from from Hive**

The HBase table remains available to you Hive world; Hive’s HBaseStorageHandler works both ways, after all.

*Note* that this command expects that the HADOOP\_CLASSPATH is still set and HIVE\_AUX\_JARS\_PATH as well if your query is complex.

|  |
| --- |
| $ hive -e "SELECT \* from pagecounts\_hbase;"  OK  en/q:Pan%27s\_Labyrinth/20081001-080000 1 1153  en/q:Special:Search/Jazz/20081001-080000 1 980  en/q:Special:Search/peinture/20081001-080000 1 989  en/q:Special:Search/rock/20081001-080000 1 980  en/qadi/20081001-080000 1 1112  en/qalawun%20complex/20081001-080000 1 942  en/qalawun/20081001-080000 1 929  en/qari'/20081001-080000 1 929  en/qasvin/20081001-080000 1 921  en/qemu/20081001-080000 1 1157  Time taken: 2.554 seconds, Fetched: 10 row(s) |

**Continue using Hive for analysis**

Since the HBase table is accessible from Hive, you can continue to use Hive for your ETL processing with mapreduce. Keep in mind that the auxpath considerations apply here too, so I’ve scripted out the query instead of just running it directly at the command line.

|  |
| --- |
| $ cat 04\_query\_hbase.hql  -- ensure hbase dependency jars are shipped with the MR job  -- Should export HIVE\_AUX\_JARS\_PATH but this is broken in HDP-1.3.x  SET hive.aux.jars.path = file:///etc/hbase/conf/hbase-site.xml,file:///usr/lib/hive/lib/hive-hbase-handler-0.11.0.1.3.2.0-111.jar,file:///usr/lib/hbase/hbase-0.94.6.1.3.2.0-111-security.jar,file:///usr/lib/zookeeper/zookeeper-3.4.5.1.3.2.0-111.jar;  -- query hive data  SELECT count(\*) from pagecounts\_hbase; |

Run it the same way we did the others.

|  |
| --- |
| $ hive -f 04\_query\_hbase.hql  Total MapReduce jobs = 1  Launching Job 1 out of 1  Number of reduce tasks determined at compile time: 1  ...  OK  10  Time taken: 19.473 seconds, Fetched: 1 row(s) |