Scanning One Table to Build Another

Providing you’ve stopped the import script from running, we can move on to the next task: extracting information from the imported wiki contents.

Wiki syntax is filled with links, some of which link internally to other articles and some of which link to external resources.

This interlinking contains a wealth of topological data. Let’s capture it!

Our goal is to capture the relationships between articles as directional links, pointing one article *to* another or receiving a link *from* another.

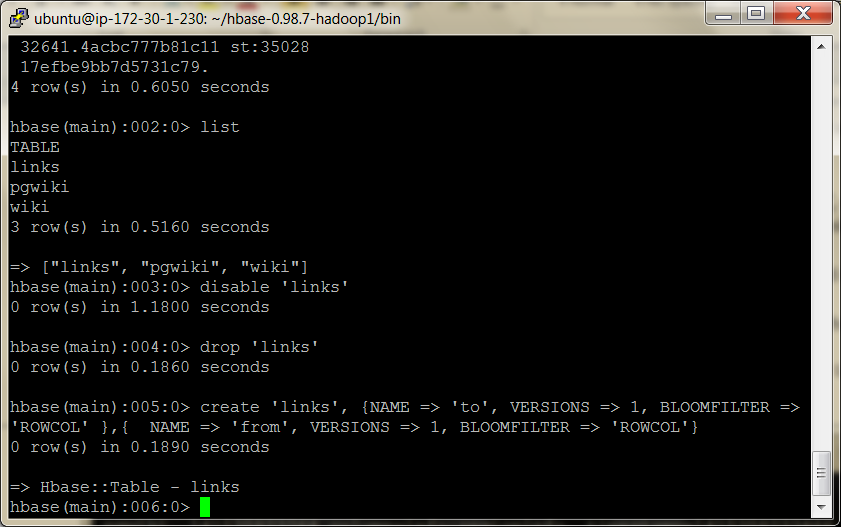
An internal article link in wikitext looks like this: *[[<target name>|<alt text>]]*, where *<target name>* is the article to link to, and *<alt text>* is the alternative text to display (optional).

For example, if the text of the article on *Star Wars* contains the string *"[[Yoda|jedi master]]"*, we want to store that relationship twice—once as an outgoing link from *Star Wars* and once as an incoming link to Yoda.

Storing the relationship twice means that it’s fast to look up both a page’s outgoing links and its incoming links.

To store this additional link data, we’ll create a new table. Head over to the shell and enter this:

**hbase> create 'links', {**NAME => 'to', VERSIONS => 1, BLOOMFILTER => 'ROWCOL' },{ NAME => 'from', VERSIONS => 1, BLOOMFILTER => 'ROWCOL'}



In principle, we could have chosen to shove the link data into an existing column family or merely added one or more additional column families to the wiki table, rather than create a new one.

Creating a separate table has the advantage that the tables have separate regions.

This means that the cluster can more effectively split regions as necessary.

The general guidance for column families in the HBase community is to try to keep the number of families per table down.

You can do this either by combining more columns into the same families or by putting families in different tables entirely.

The choice is largely decided by whether and how often clients will need to get an entire row of data (as opposed to needing just a few column values).

In our wiki case, we need the text and revision column families to be on the same table so that when we put new revisions in, the metadata and the text share the same timestamp.

The links content, by contrast, will never have the same timestamp as the article from which the data came.

Further, most client actions will be interested either in the article text or in the extracted information about article links but probably not in both at the same time.

So, splitting out the to and from column families into a separate table makes sense.

Constructing the Scanner

With the links table created, we’re ready to implement a script that will scan all the rows of the wiki table.

Then, for each row, retrieve the wikitext and parse out the links.

Finally, for each link found, create incoming and outgoing link table records.

See book p117 for script

Running the Script

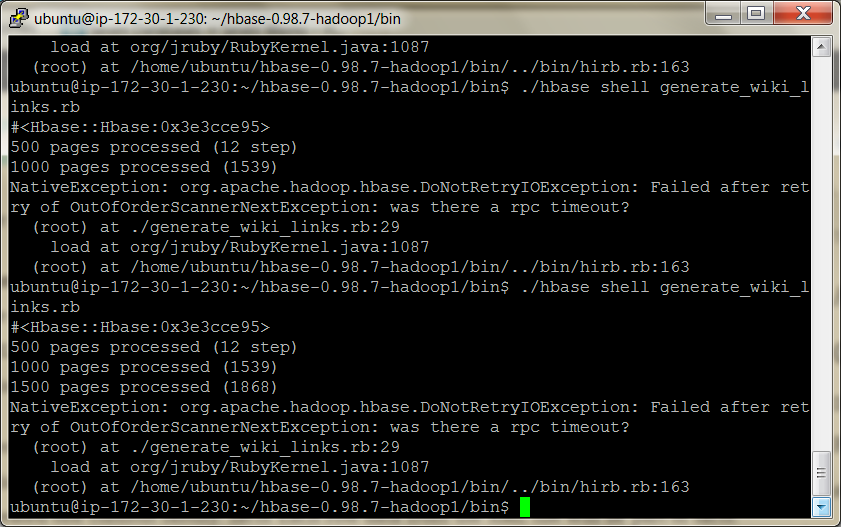
kick off the script.

${HBASE\_HOME}/bin/hbase shell generate\_wiki\_links.rb

It should produce output like this:

500 pages processed (10 petametres)  
1000 pages processed (1259)  
1500 pages processed (1471 BC)  
2000 pages processed (1683)

...



As with the previous script, you can let it run as long as you like, even to completion.

If you want to stop it, press CTRL+C.

You can monitor the disk usage of the script using du as we’ve done before.

You’ll see new entries for the links table we just created, and the size counts will increase as the script runs.

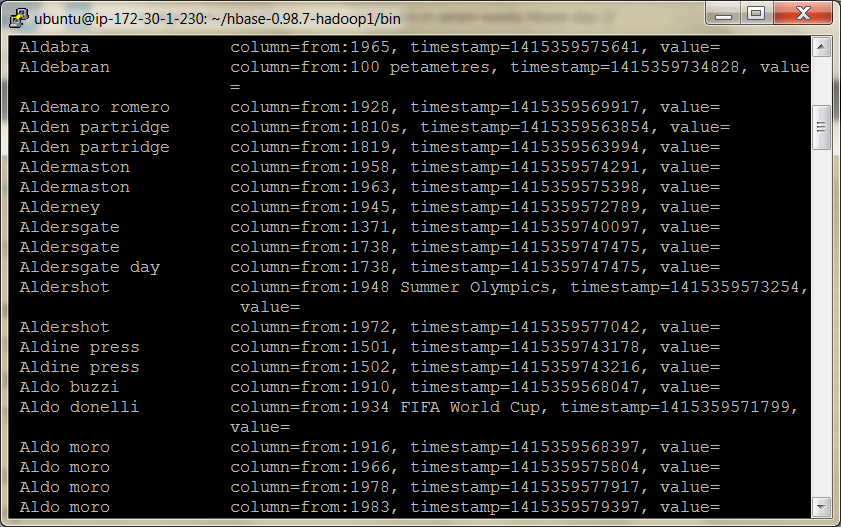
Examining the Output

Now we’ll use the shell’s scan command to simply dump part of a table’s contents to the console.

For each link the script finds in a text: blob, it will indiscriminately create both to and from entries in the links table.

To see the kinds of links being created, head over to the shell and scan the table.

**hbase> scan 'links', STARTROW => "Admiral Ackbar", ENDROW => "It's a Trap!"**



You should get a whole bunch of output. Of course, you can use the get command to see the links for just a single article.

**hbase> get 'links', 'Star Wars'**

COLUMN CELL

...  
 links:from:Admiral Ackbar  
 links:from:Adventure  
 links:from:Alamogordo, New Mexico  
 links:to:"weird al" yankovic  
 links:to:20th century fox

links:to:3-d film  
 links:to:Aayla secura  
 ...

timestamp=1300415922636, value=  
timestamp=1300415927098, value=  
timestamp=1300415953549, value=  
timestamp=1300419602350, value=  
timestamp=1300419602350, value=

timestamp=1300419602350, value=  
timestamp=1300419602350, value=

In the wiki table, the rows are very regular with respect to columns

As you recall, each row has text:, revision:author, and revision:comment columns.

The links table has no such regularity. Each row may have one column or hundreds. And the variety of column names is as diverse as the row keys themselves (titles of Wikipedia articles).

That’s OK! HBase is a so-called sparse data store for exactly this reason.

Our experimentation so far has focused on accessing a single local server.

In reality, if you choose to use HBase, you’ll want to have a good sized cluster in order to realize the performance benefits of its distributed architecture.