Denodo HBase CustomWrapper - User Manual

Revision 20130705

NOTE
This document is confidential and proprietary of **Denodo Technologies**.
No part of this document may be reproduced in any form by any means without prior written authorization of **Denodo Technologies**.

Copyright © 2013
Denodo Technologies Proprietary and Confidential

CONTENTS

1 INTRODUCTION

The hbase-customwrapper is a Virtual DataPort custom wrapper that enables VDP to perform read operation on an HBase database.

HBase is a column-oriented noSQL database management system that runs on top of Hadoop infrastructure, that allows you to store and process large amounts of data on multiple machines.

As usually happens in the NoSQL database world, Hbase does not support SQL queries, therefore it is necessary to use a binary client API to access to the database system.

***NOTE**: The scope of this wrapper is limited because it depends on the specific version and configuration of the HBase server you are accessing. Besides, there are direct dependencies between the binary files of the server and the client. For this reason, it is likely that it will be necessary to perform minor code-level modifications to the wrapper in order to make it compatible with specific HBase installations: version of the API client libraries, configuration entries at the hbase-site.xml file, etc.*
Also note that this wrapper contains the hbase client libraries themselves (a requirement of HBase), so in order to guarantee the wrapper will work, the machine on which VDP runs must meet all the HBase client libraries configuration requirements (network reachability, DNS configuration, etc.). Said in other words: this wrapper should work OK if a standalone java application using the HBase client libraries work OK, in the same machine.

2 FEATURES

2.1 JAVA ARTIFACT ORGANIZATION

This wrapper uses the HBase Java client API, which in turn has a dependency on the hadoop.core, zookeeper, and thrift libraries. These binary dependencies have two undesirable, but unavoidable effects:

1. Increase the size of the binary artifacts deployed into Virtual DataPort.
2. Un-generalize the custom wrapper, by binding it (and its distribution) to specific hadoop/HBase software versions.

2.2 ARCHITECTURE

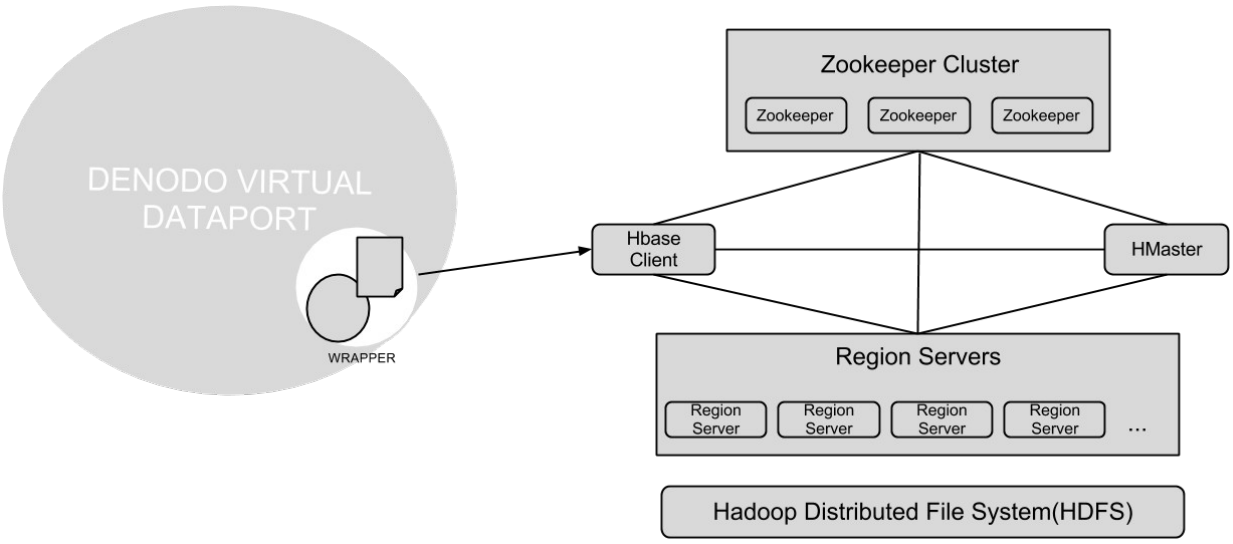
The HBase architecture has two main components:

1. One **HMaster** node/process that is responsible for coordinating the cluster and execute administrative operations like region allocation and failover, log splitting, and load balancing.

1. One or several **HRegionServers** that are responsible for handling a subset of the table's data and I/O requests to the hosting regions, flushing the in-memory data store (MemStore) to HDFS, and splitting and compacting regions.

The HBase client provides the wrapper with the required API to access to the HBase Cluster. The ZooKeeper cluster acts as a coordination service for the entire HBase cluster, handling master selection, root region server lookup, node registration,configuration information, hierarchical naming space, and so on.

The next diagram provides a general overview of the system architecture:



2.3 DATA MODEL

1. **Tables** in HBase are made up of rows and columns.
2. **Columns** are grouped into family columns. A column name is made of its column family name and a qualifier familyName:columnName.
3. **Family Column** regroups data of the same nature, and they are stored together on the filesystem.
4. **Rows** have one row key. They are lexicographically sorted with the lowest order appearing first in a table.
5. **Cells** are identified by row, column and version. The cell contents are an uninterpreted array of bytes.
6. **Versions** are cells that have the same column and table, every version has associated a timestamp, that it is a long(milliseconds) . The HBase version dimension is stored in decreasing order, the latest value is founded first. The number of versions that we can retrieve when a cell is retrieved is configurable. HBase, by default, returns the latest version.

You can read more about the data model in the next link: http://hbase.apache.org/book/datamodel.html.

2.4 CAPABILITIES

This wrapper is only able to carry out read operations. It can delegate to HBase the following query artifacts and operators:

1. Operators: =, <>,REGEXP_LIKE, LIKE, IN, IS NULL, IS NOT NULL, CONTAINS,IS_CONTAINED, CONTAINS_AND, CONTAINS_OR.
2. AND operations.

3. OR operations.

Owing to the delegation of the previous operators, the performance of the custom wrapper is better because many operations will be performed by HBase itself, instead of having to rely on VDP in-memory post-filtering.

2.5 LIMITATIONS

There are some limitations which should be taken into account when using this wrapper:

1. Read-only. Update and delete operations could be developed in the future, but they aren't available as of current versions.
2. Row keys are uninterpreted bytes, so we have to be careful if we want to make VDP read data types different to the string one.
3. The <,<=,>,=> operators cannot be delegated to HBase because its API-driven comparison operations are byte-only based, which makes it unsuitable for comparing numbers.
4. The wrapper cannot not make the most out of the possibilities of HBase because of the difference in paradigm between this column-oriented noSQL system and the relational paradigm VDP implements.
5. No authentication supported yet.

3 USAGE

In order to use the hbase-customwrapper, first it is necessary to add its jar file (the jar-with-dependencies version) into the Denodo VDP Tool in “File -> Extensions”. It is possible that you need to reserve more memory for the Virtual DataPort Administration Tool. For this, in Denodo Platform Control Center you have to go to Configuration->JVM Options and change the parameter Virtual DataPort Administration Tool. It is owed to this wrapper has a large size, because it require jars with a big size like HBase.

After this, a data source should be created in “File -> DataSource -> Custom”, selecting the pertinent jar, and writing a name for the data source.

Because of the size of the jar file, it can be necessary to increase the size of the memory dedicated to the Virtual Data Port Administration Tool (the wrapper needs the hadoop-core, hbase and related libraries, which are quite heavy). So, if an error shows when we try to add the jar we should to go to the Denodo Platform Control Center -> Configure -> JVM Options and increase the heap size assigned to the VDP Administration Tool.

The hbase-customwrapper include the configuration file hbase-site.xml, in the file are the values by default commented. It is possible that it will be necessary to change any of these parameter, so we have to uncomment that specific attribute and change its value. For instance in the Virtual Machine of Hortonworks, Sandbox in its version 1.3, it is necessary to change the property zookeeper.znode.parent from its value by default /hbase to /hbase-unsecure.

This wrapper has been tested with two versions of the virtual machine of Hortonworks, Sandbox, 1.2.1 and 1.3.1, and they use the versions of Hbase 0.94.2 and 0.94.6.1, respectively.

3.1 CREATING BASE VIEWS

In order to create a base view, we will have to fill in the fields we can see in the following image:

The hbaseIP field is the IP through which we want to access the database system (this parameter can be a list of HBase IPs separated by commas), hbasePort is the ZooKeeper port, an optional field with a default value of 2181.

tableName is the HBase table from which we want to extract the data.

The tableMapping field expects a fragment of JSON, giving information about the queried HBase data structure and defining the elements that will be projected into the base view. An example:

```
{
   \'post\': {
      \'title\': \'text\',
      \'body\': \'text\'
   },
   \'image\': {
      \'bodyimage\': \'text\',
      \'header\': \'text\',
      \'active\': \'boolean\'
   }
}
```

*post and image are column families, and the other sub-elements are the columns.

3.2 EXAMPLES

In the next example we want to extract the rows of the table blogposts table which title value (in the post family), starts with 'Hello World'. So, in the HBase Shell we execute the next command.

```
scan 'blogposts', {FILTER =>
"(SingleColumnValueFilter('post','title',=,'regexstring:Hello World'))"}.
```

In the next image we can see the results of this query:

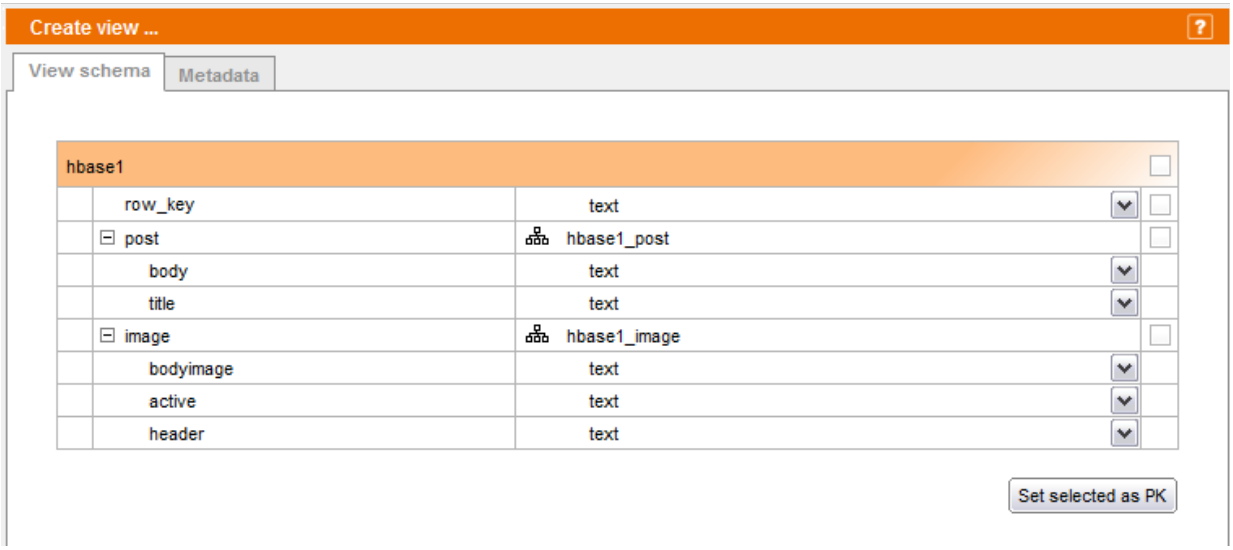
```
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.94.2.21, rUnknown, Thu Jan 10 03:45:56 PST 2013

hbase(main):001:0> scan 'blogposts', {FILTER => "(SingleColumnValueFilter('post','title',=,'regexstring:Hello World'))"}
ROW COLUMN+CELL
post1 column=image:active, timestamp=1368536115329, value=true
post1 column=image:bodyimage, timestamp=1368536102858, value=5
post1 column=image:header, timestamp=1368536090247, value=image1
.jpg
post1 column=post:author, timestamp=1368536076921, value=The Aut
hor
post1 column=post:body, timestamp=1368536083159, value=This is a
blog post
post1 column=post:title, timestamp=1368536070847, value=Hello Wo
rld
post2 column=image:active, timestamp=1368537884925, value=false
post2 column=image:bodyimage, timestamp=1368537896082, value=0
post2 column=image:header, timestamp=1368537831101, value=imghea
der2
post2 column=post:author, timestamp=1368537743990, value=PLC
post2 column=post:body, timestamp=1368537800137, value=This is t
he second history
post2 column=post:title, timestamp=1368537706887, value=Hello Wo
rld2
post3 column=image:bodyimage, timestamp=1368537903556, value=10
post3 column=image:header, timestamp=1368537838765, value=imghea
der3
post3 column=post:title, timestamp=1368537714692, value=Hello Wo
rld3
3 row(s) in 2.7590 seconds

hbase(main):002:0>
```

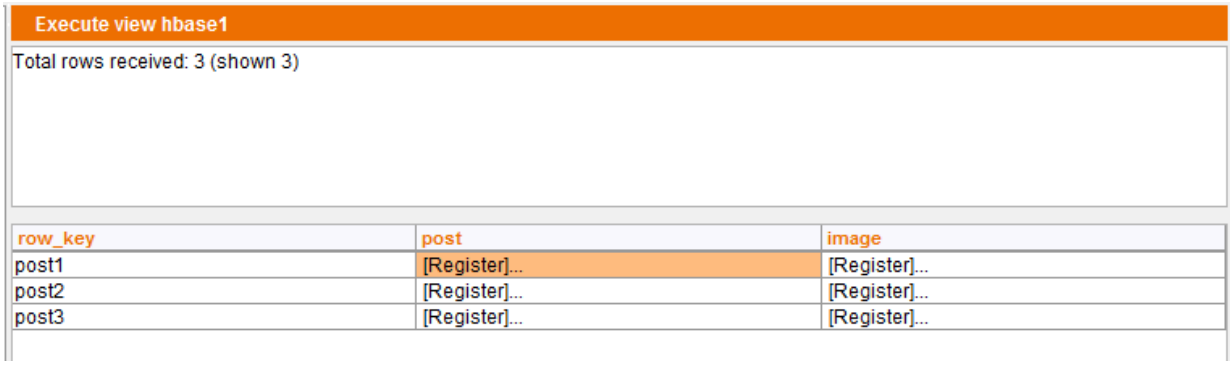
It is possible to reach to the same result using VDP and the hbase-customwrapper:

Using the JSON fragment we saw in a previous section as table mapping, we create a base view:



Once the base view is created, it can be used in VDP just like any other views. Let's see a VQL sentence equivalent to the above HBase shell command:

SELECT * FROM hbase_example WHERE (post).title like 'Hello World%'



From the row_key values, we can check that the same results are obtained as if the HBase shell had been used instead.

The previous image does not allow to see the array itself (it would require flattening), so here it is (for the first returned tuple):