[Apache Drill](https://drill.apache.org/" \t "_blank) is *an innovative distributed SQL engine designed to enable data exploration and analytics on non-relational datastores […] without having to create and manage schemas. […] It has a schema-free JSON document model similar to MongoDB and Elasticsearch; [a plethora of APIs, including] ANSI SQL, ODBC/JDBC, and HTTP[S] REST; [is] extremely user and developer friendly; [and, has a] pluggable architecture enables connectivity to multiple datastores*.

To ring in the new year the Drill team knocked out a new [1.15.0 release](https://drill.apache.org/docs/apache-drill-1-15-0-release-notes/) with a cadre of new functionality including:

* [CROSS JOIN](https://drill.apache.org/docs/from-clause/#join-types) support
* New metadata tables for [files](https://drill.apache.org/docs/querying-the-information-schema/#files), [system functiosn](https://drill.apache.org/docs/querying-system-tables/#querying-the-functions-table), and [system functions](https://drill.apache.org/docs/querying-system-tables/#querying-the-options-table)
* [pcapng](https://issues.apache.org/jira/browse/DRILL-6179) support
* enhanced datetime functions
* better security
* and, web UI improvements

One super-helpful new feature of the REST API is that it now returns query results metadata along with the query results themselves. This means REST API endpoints finally know both column order *and* column type. This gave me cause to re-visit the sergeant package [[GL](https://gitlab.com/hrbrmstr/sergeant/tree/0.8.0)|[GH](https://github.com/hrbrmstr/sergeant/tree/0.8.0)] and make some accommodations for some of these new features.

**Ushering In A New Order**

Drill REST API queries return a "columns" field and "metadata" field with the data itself. We can use that to force an order to the columns as well as *mostly* use proper types (vs JSON-parsed/guessed types). I say *mostly* since the package still uses jsonlite to parse the results and there’s no support for 64-bit integers in jsonlite (more on this later).

We’ll use the example from [DRILL-6847](https://issues.apache.org/jira/browse/DRILL-6847) and use the example provided by Charles Givre in his Jira issue since it will let me demonstrate more of that “*mostly*” comment and show off another new feature:

library(sergeant) # 0.8.0 branch of sergeant on gitlab or github

library(tidyverse)

con <- src\_drill("localhost")

x <- tbl(con, "cp.`employee.json`")

mutate(x, employee\_id = as.integer64(employee\_id)) %>%

mutate(position\_id = as.integer64(position\_id)) %>%

select(

employee\_id, full\_name, first\_name, last\_name,

position\_id, position\_title

) -> bigint\_result

The above is (logically):

SELECT

CAST (employee\_id AS INT) AS employee\_id,

full\_name,

first\_name,

last\_name,

CAST (position\_id AS BIGINT) AS position\_id,

position\_title

FROM cp.`employee.json`

What do we get when we take a preview of the result?

bigint\_result

## # Source: lazy query [?? x 6]

## # Database: DrillConnection

## employee\_id full\_name first\_name last\_name position\_id position\_title

##

## 1 1 Sheri Now… Sheri Nowmer 1 President

## 2 2 Derrick W… Derrick Whelply 2 VP Country Man…

## 3 4 Michael S… Michael Spence 2 VP Country Man…

## 4 5 Maya Guti… Maya Gutierrez 2 VP Country Man…

## 5 6 Roberta D… Roberta Damstra 3 VP Information…

## 6 7 Rebecca K… Rebecca Kanagaki 4 VP Human Resou…

## 7 8 Kim Brunn… Kim Brunner 11 Store Manager

## 8 9 Brenda Bl… Brenda Blumberg 11 Store Manager

## 9 10 Darren St… Darren Stanz 5 VP Finance

## 10 11 Jonathan … Jonathan Murraiin 11 Store Manager

## # ... with more rows

Warning message:

One or more columns are of type BIGINT. The sergeant package currently uses jsonlite::fromJSON()

to process Drill REST API result sets. Since jsonlite does not support 64-bit integers BIGINT

columns are initially converted to numeric since that's how jsonlite::fromJSON() works. This is

problematic for many reasons, including trying to use 'dplyr' idioms with said converted

BIGINT-to-numeric columns. It is recommended that you 'CAST' BIGINT columns to 'VARCHAR' prior to

working with them from R/'dplyr'.

If you really need BIGINT/integer64 support, consider using the R ODBC interface to Apache Drill

with the MapR ODBC drivers.

This informational warning will only be shown once per R session and you can disable them from

appearing by setting the 'sergeant.bigint.warnonce' option to 'FALSE'

(i.e. options(sergeant.bigint.warnonce = FALSE)).

The first thing sergeant users will notice is proper column order (before it just returned the columns in the order they came back in the JSON rows[] structure). The second thing is that we didn’t get integer64s back. Instead, we got doubles plus an information warning about why and what you can do about it. Said warning only displays once per-session and can be silenced with the option sergeant.bigint.warnonce. i.e. just put:

options(sergeant.bigint.warnonce = FALSE)

in your script or ~/.Rprofile and you won’t hear from it again.

The as.integer64() we used is not from the bit64 package but an internal sergeant package function that knows how to translate said operation to, e.g. CAST( employee\_id AS BIGINT ).

You can use the ODBC drivers to gain BIGINT support and there are plans for the 0.8.0 branch to eventually use rapidjsonr at the C++-level to provide direct in-package support for BIGINTs as well.

**Better Error Messages**

Drill query errors that the sergeant package bubbled up through its various interfaces have not been pretty or all that useful. This has changed with the 0.8.0 branch. Let’s take a look:

tbl(con, "cp.employees.json")

## # Source: table [?? x 4]

## # Database: DrillConnection

Warning message:

VALIDATION ERROR: From line 2, column 6 to line 2, column 24: Object 'cp.employees.json' not found

Original Query:

1: SELECT \*

2: FROM `cp.employees.json`

3: LIMIT 10

Query Profile Error Link:

http://localhost:8047/profiles/079fc8cf-19c6-4c78-95a9-0b949a3ecf4c

As you can see in the above output, you now get a highly-formatted return value with the original SQL query broken into lines (with line numbers) and a full link to the Drill query profile so you can dig in to the gnarly details of complex query issues. As you work with this and find edge cases I missed for messages, drop an issue on your social-coding site of choice.

**SUPPORT ALL THE PCAPs!**

Drill has had packet capture (PCAP) file support for a while now and 1.15.0 adds support for the more modern/rich [pcapng format](http://pcapng.com/). To enable support for this you need to add "pcapng": {"type": "pcapng", "extensions": ["pcapng"] }, to the "formats" section of your storage plugins and also configure a workspace directory to use that as the default (the principle of which is [covered here](https://rud.is/books/drill-sergeant-rstats/adding-or-modifying-drill-formats.html)).

We’ll use one of the [Wireshark example captures](https://wiki.wireshark.org/SampleCaptures) to demonstrate:

pcaps <- tbl(con, "dfs.caps.`\*.pcapng`")

glimpse(pcaps)

## Observations: ??

## Variables: 25

## $ tcp\_flags\_ece\_ecn\_capable 0, 0, 0, 0, 0, 0, 0, 0, 0...

## $ tcp\_flags\_ece\_congestion\_experienced 0, 0, 0, 0, 0, 0, 0, 0, 0...

## $ tcp\_flags\_psh 0, 0, 0, 0, 0, 0, 0, 0, 0...

## $ type "TCP", "TCP", "TCP", "TCP...

## $ tcp\_flags\_cwr 0, 0, 0, 0, 0, 0, 0, 0, 0...

## $ dst\_ip "74.125.28.139", "10.254....

## $ src\_ip "10.254.157.208", "74.125...

## $ tcp\_flags\_fin 1, 1, 0, 0, 0, 0, 0, 0, 0...

## $ tcp\_flags\_ece 0, 0, 0, 0, 0, 0, 0, 0, 0...

## $ tcp\_flags 17, 17, 16, 16, 16, 0, 0,...

## $ tcp\_flags\_ack 1, 1, 1, 1, 1, 0, 0, 0, 0...

## $ src\_mac\_address "00:05:9A:3C:7A:00", "00:...

## $ tcp\_flags\_syn 0, 0, 0, 0, 0, 0, 0, 0, 0...

## $ tcp\_flags\_rst 0, 0, 0, 0, 0, 0, 0, 0, 0...

## $ timestamp 2015-04-14 07:19:25, 201...

## $ tcp\_session 8.353837e+17, 8.353837e+1...

## $ packet\_data "\"3DU... "ACK|FIN", "ACK|FIN", "AC...

## $ tcp\_flags\_ns 0, 0, 0, 0, 0, 0, 0, 0, 0...

## $ src\_port 60268, 443, 60268, 58382,...

## $ packet\_length 54, 54, 54, 55, 66, 78, 7...

## $ tcp\_flags\_urg 0, 0, 0, 0, 0, 0, 0, 0, 0...

## $ tcp\_ack 662445631, 1496589825, 66...

## $ dst\_port 443, 60268, 443, 29216, 5...

## $ dst\_mac\_address "00:11:22:33:44:55", "00:...

count(pcaps, src\_ip, dst\_ip, sort=TRUE)

## # Source: lazy query [?? x 3]

## # Database: DrillConnection

## # Groups: src\_ip

## # Ordered by: desc(n)

## src\_ip dst\_ip n

##

## 1 10.254.157.208 10.254.158.25 298

## 2 10.254.158.25 10.254.157.208 204

## 3 174.137.42.81 10.254.157.208 76

## 4 10.254.157.208 10.254.158.8 54

## 5 10.254.158.8 10.254.157.208 49

## 6 74.125.28.102 10.254.157.208 49

## 7 10.254.157.208 74.125.28.102 44

## 8 10.254.157.208 174.137.42.81 41

## 9 54.84.98.25 10.254.157.208 25

## 10 157.55.56.168 10.254.157.208 25

## # ... with more rows

More work appears to be planned by the Drill team to enable digging into the packet (binary) contents.

**Drill Metadata As Data**

Drill has provided ways to lookup Drill operational information as actual tables but the Drill team has added support for even more metadata-as-data queries.

First up is finally having better access to filesystem information. Prior to 1.15.0 one could get file and path attributes as part of other queries, but now we can treat filesystems as actual data. Let’s list all the PCAPs in the above workspace:

tbl(con, "information\_schema.`schemata`") %>%

filter(SCHEMA\_NAME == "dfs.caps") %>%

print() %>%

pull(SCHEMA\_NAME) -> pcap\_schema

## # Source: lazy query [?? x 9]

## # Database: DrillConnection

## CATALOG\_NAME SCHEMA\_NAME SCHEMA\_OWNER TYPE IS\_MUTABLE

##

## 1 DRILL dfs.caps file NO

tbl(con, "information\_schema.`files`") %>%

filter(schema\_name == pcap\_schema) %>%

glimpse()

## Observations: ??

## Variables: 13

## $ SCHEMA\_NAME "dfs.caps"

## $ ROOT\_SCHEMA\_NAME "dfs"

## $ WORKSPACE\_NAME "caps"

## $ FILE\_NAME "dof-short-capture.pcapng"

## $ RELATIVE\_PATH "dof-short-capture.pcapng"

## $ IS\_DIRECTORY FALSE

## $ IS\_FILE TRUE

## $ LENGTH 634280

## $ OWNER "hrbrmstr"

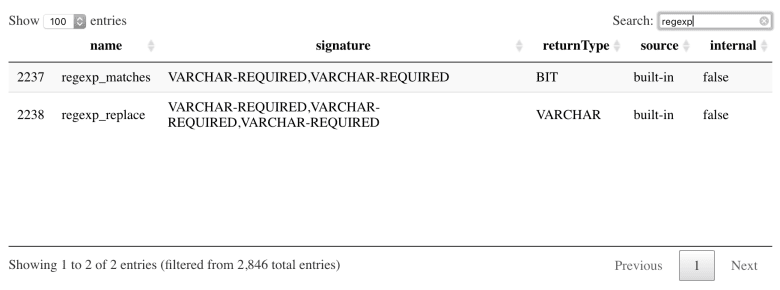
## $ GROUP "staff"

## $ PERMISSION "rw-r--r--"

## $ ACCESS\_TIME 1969-12-31 19:00:00

## $ MODIFICATION\_TIME 2019-01-01 19:12:17

The Drill system options table now has full descriptions for the options and also provides a new table that knows about all of Drills functions and all *your* custom UDFs. drill\_opts() and drill\_functions() return a data frame of all this info and have an optional browse parameter which, if set to TRUE, will show a DT interactive data table for them. I find this especially handy when I forget something like regexp\_like syntax (I use *alot* of back-ends and many are wildly different) and can now do this:

[](https://rud.is/b/2019/01/02/apache-drill-1-15-0-sergeant-0-8-0-pcapng-support-proper-column-types-mounds-of-new-metadata/drill-dt/)

**FIN**

Keep on the lookout for the rapidjsonr/BIGINT integration and more new features of the sergeant package. NOTE: The better error messages have been ported over to the sergeant.caffeinated package (the RJDBC interface) and the other niceties will make their way into that package soon as well.

So, make sure you’re using the 0.8.0 [GL](https://gitlab.com/hrbrmstr/sergeant/tree/0.8.0) / [GH](https://github.com/hrbrmstr/sergeant/tree/0.8.0), kick the tyres, file issues where you’re most comfortable working.

May your queries all be optimized and results sets complete in the new year!