rittmanmead (http://www.rittmanmead.com/)

Using SQL to Query JSON Files with Apache Drill

01 NOVEMBER 2016 on jq (/blog/tag/jq/), apache drill (/blog/tag/apache-drill/), json (/blog/tag/json/), sql (/blog/tag/sql/), sql on hadoop (/blog/tag/sql-on-hadoop/)

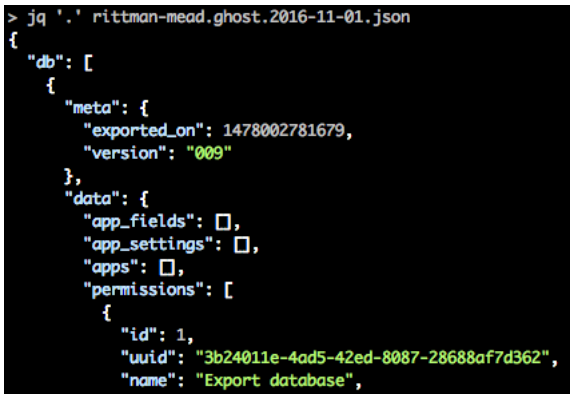
I wrote recently about what Apache Drill is (https://www.rittmanmead.com/blog/2016/08/an-introduction-to-apache-drill/), and how to use it with OBIEE (https://www.rittmanmead.com/blog/2016/08/using-apache-drill-with-obiee-12c/). In this post I wanted to demonstrate its great power in action for a requirement that came up recently. We wanted to analyse our blog traffic, broken down by blog author. Whilst we have Google Analytics to provide the traffic, it doesn't include the blog author. This is held within the blog platform, which is Ghost. The common field between the two datasets is the post "slug". From Ghost we could get a dump of the data (https://help.ghost.org/hc/en-us/articles/224112927-Import-Export-Data) in JSON format. We needed to find a quick way to analyse and extract from this JSON a list of post slugs and associated author.

One option would be to load the JSON into a RDBMS and process it from within there, running SQL queries to extract the data required. For a long-term large-scale solution, maybe this would be appropriate. But all we wanted to do here was query a single file, initially just as a one-off. Enter Apache Drill (https://drill.apache.org/). Drill can run on a single laptop (or massively clustered, if you need it). It provides a SQL engine on top of various data sources, including text data on local or distributed file systems (such as HDFS).

You can use Drill to dive straight into the json:

```
0: jdbc:drill:zk=local> use dfs;
+-------+-----------------------------------+
|  ok   |              summary              |
+-------+-----------------------------------+
| true  | Default schema changed to [dfs]   |
+-------+-----------------------------------+
1 row selected (0.076 seconds)
0: jdbc:drill:zk=local> select * from `/Users/rmoff/Downloads/rittman-mead.ghost.2016-11-01.json` limit 1;
+----+
| db |
+----+
| [{"meta":{"exported_on":1478002781679,"version":"009"},"data":{"permissions":[{"id":1,"uuid":"3b24011e-4ad5-42ed-8087-28688af7d36
2","name":"Export database","object_type":"db","action_type":"exportContent","created_at":"2016-05-23T11:24:47.000Z","created_by":1,"up
dated_at":"2016-05-23T11:24:47.000Z","updated_by":1},{"id":2,"uuid":"55b92b4a-9db5-4c7f-8fba-8065c1b4b7d8","name":"Import database","ob
ject_type":"db","action_type":"importContent","created_at":"2016-05-23T11:24:47.000Z","created_by":1,"updated_at":"2016-05-23T11:24:47.
000Z","updated_by":1},{"id":3,"uuid":"df98f338-5d8c-4683-8ac7-fa94dd43d2f1","name":"Delete all content","object_type":"db","action_typ
e":"deleteAllContent","created_at":"2016-05-23T11:24:47.000Z","created_by":1,"updated_at":"2016-05-23T11:24:47.000Z","updated_by":1},
{"id":4,"uuid":"a3b8c5c7-7d78-442f-860b-1cea139e1dfc","name":"Send mail","object_type":"mail","action_
```

But from this we can see the JSON object is a single column `db` of array type. Let's take a brief detour into one of my favourite commandline tools - `jq` (https://stedolan.github.io/jq/). This let's you format, filter, and extract values from JSON. Here we can use it to get an idea of how the data's structured. We *can* do this in Drill, but `jq` gives us a headstart:



We can see that under the `db` array are two elements; `meta` and `data`. Let's take `meta` as a simple example to expose through Drill, and then build from there into the user data that we're actually after.

Since the root data element (`db`) is an array, we need to `FLATTEN` (https://drill.apache.org/docs/flatten/) it:

```
0: jdbc:drill:zk=local> select flatten(db) from `/Users/rmoff/Downloads/rittman-mead.ghost.2016-11-01.json` limit 1;
+--------+
| EXPR$0 |
+--------+
| {"meta":{"exported_on":1478002781679,"version":"009"},"data":{"permissions":[{"id":1,"uuid":"3b24011e-4ad5-42ed-8087-28688af7d362","n
ame":"Export database","object_type":"db","action_type":"exportContent","created_at":"2016-05-23T11:24:47.000Z","created_by":1,"updated
_at":"2016-05-23T11:24:47.000Z","updated_by":1},{"id":2,"uuid":"55b92b4a-9db5-4c7f-8fba-8065c1b4b7d8","name":"Import database","object_
type":"db","action_type":"importContent","created_at":"2016-05-23T11:24:47.000Z","created_by":1,"updated_at":"2016-05-23T11:24:47.000
Z","u
```

Now let's query the `meta` element itself:

```
0: jdbc:drill:zk=local> with db as (select flatten(db) from `/Users/rmoff/Downloads/rittman-mead.ghost.2016-11-01.json`) select db.meta
from db limit 1;
Nov 01, 2016 2:18:31 PM org.apache.calcite.sql.validate.SqlValidatorException <init>
SEVERE: org.apache.calcite.sql.validate.SqlValidatorException: Column 'meta' not found in table 'db'
Nov 01, 2016 2:18:31 PM org.apache.calcite.runtime.CalciteException <init>
SEVERE: org.apache.calcite.runtime.CalciteContextException: From line 1, column 108 to line 1, column 111: Column 'meta' not found in t
able 'db'
Error: VALIDATION ERROR: From line 1, column 108 to line 1, column 111: Column 'meta' not found in table 'db'

SQL Query null

[Error Id: 9cb4aa98-d522-42bb-bd69-43bc3101b40e on 192.168.10.72:31010] (state=,code=0)
```

This didn't work, because if you look closely at the above `FLATTEN`, the resulting column is called `EXPR$0`, so we need to alias it in order to be able to reference it:

```
0: jdbc:drill:zk=local> select flatten(db) as db from `/Users/rmoff/Downloads/rittman-mead.ghost.2016-11-01.json`;
+----+
| db |
+----+
| {"meta":{"exported_on":1478002781679,"version":"009"},"data":{"permissions":[{"id":1,"uuid":"3b24011e-4ad5-42ed-8087-28688af7d362","n
ame":"Export database","object_type":"db","action_type":"exportConten
```

Having done this, I'll put the `FLATTEN` query as a subquery using the `WITH` syntax, and from that `SELECT` just the `meta` elements:

```
0: jdbc:drill:zk=local> with ghost as (select flatten(db) as db from `/Users/rmoff/Downloads/rittman-mead.ghost.2016-11-01.json`) selec
t ghost.db.meta from ghost limit 1;
+------------------------------------------------+
|                     EXPR$0                     |
+------------------------------------------------+
| {"exported_on":1478002781679,"version":"009"}  |
+------------------------------------------------+
1 row selected (0.317 seconds)
```

Note that the column is `EXPR$0` because we've not defined a name for it. Let's fix that:

```
0: jdbc:drill:zk=local> with ghost as (select flatten(db) as db from `/Users/rmoff/Downloads/rittman-mead.ghost.2016-11-01.json`) selec
t ghost.db.meta as meta from ghost limit 1;
+------------------------------------------------+
|                      meta                      |
+------------------------------------------------+
| {"exported_on":1478002781679,"version":"009"}  |
+------------------------------------------------+
1 row selected (0.323 seconds)
0: jdbc:drill:zk=local>
```

Why's that matter? Because it means that we can continue to select elements from within it.

We could continue to nest the queries, but it gets messy to read, and complex to debug any issues. Let's take this `meta` element as a base one from which we want to query, and define it as a `VIEW`:

```
0: jdbc:drill:zk=local> create or replace view dfs.tmp.ghost_meta as with ghost as (select flatten(db) as db from `/Users/rmoff/Downloa
ds/rittman-mead.ghost.2016-11-01.json`) select ghost.db.meta as meta from ghost;
+-------+-----------------------------------------------------------+
|  ok   |                          summary                          |
+-------+-----------------------------------------------------------+
| true  | View 'ghost_meta' created successfully in 'dfs.tmp' schema  |
+-------+-----------------------------------------------------------+
1 row selected (0.123 seconds)
```

Now we can select from the view:

```
0: jdbc:drill:zk=local> select m.meta.exported_on as exported_on, m.meta.version as version from dfs.tmp.ghost_meta m;
+---------------+----------+
|  exported_on  | version  |
+---------------+----------+
| 1478002781679 | 009      |
+---------------+----------+
1 row selected (0.337 seconds)
```

Remember that when you're selected nested elements you *must* alias the object that you're selecting from. If you don't, then Drill assumes that the first element in the column name (for example, `meta.exported_on`) is the table name (`meta`), and you'll get an error:

```
Error: VALIDATION ERROR: From line 1, column 8 to line 1, column 11: Table 'meta' not found
```

So having understood how to isolate and query the `meta` element in the JSON, let's progress onto what we're actually after - the name of the author of each post, and associated 'slug'.

Using `jq` again we can see the structure of the JSON file, with the code taken from here (https://github.com/stedolan/jq/issues/243#issuecomment-45460474):

```
> jq 'path(..)|[.[]|tostring]|join("/")' rittman-mead.ghost.2016-11-01.json |grep --color=never post|more
"db/0/data/posts"
"db/0/data/posts/0"
"db/0/data/posts/0/id"
"db/0/data/posts/0/uuid"
"db/0/data/posts/0/title"
[...]
```

So Posts data is under the `data.posts` element, and from manually poking around we can see that user data is under `data.users` element.

Back to Drill, we'll create views based on the same pattern as we used for `meta` above; flattening the array and naming the column:

```
use dfs.tmp;
create or replace view ghost_posts as select flatten(ghost.db.data.posts) post from ghost;
create or replace view ghost_users as select flatten(ghost.db.data.users) `user` from ghost;
```

The `ghost` view is the one created above, in the `dfs.tmp` schema. With these two views created, we can select values from each:

```
0: jdbc:drill:zk=local> select u.`user`.id,u.`user`.name from ghost_users u where u.`user`.name = 'Robin Moffatt';
+---------+----------------+
| EXPR$0  |     EXPR$1     |
+---------+----------------+
| 15      | Robin Moffatt  |
+---------+----------------+
1 row selected (0.37 seconds)

0: jdbc:drill:zk=local> select p.post.title,p.post.slug,p.post.author_id from ghost_posts p where p.post.title like '%Drill';
+----------------------------------+----------------------------------+---------+
|              EXPR$0              |              EXPR$1              | EXPR$2  |
+----------------------------------+----------------------------------+---------+
| An Introduction to Apache Drill  | an-introduction-to-apache-drill  | 15      |
+----------------------------------+----------------------------------+---------+
1 row selected (0.385 seconds)
```

and join them:

(https://twitter.com/intent/tweet?text=Using%20SQL%20to%20Query%20JSON%20Files%20with%20Apache%20Drill&url=https://www.rittmanmead.com/blog/2016/11/using-sql-to-query-json-files-with-apache-drill/) (https://www.facebook.com/sharer/sharer.php?u=https://www.rittmanmead.com/blog/2016/11/using-sql-to-query-json-files-with-apache-drill/) (https://plus.google.com/share?url=https://www.rittmanmead.com/blog/2016/11/using-sql-to-query-json-files-with-apache-drill/)

TECHNICAL INSIGHTS (/BLOG/TAG/TECHNICAL)

BUSINESS INSIGHTS (/BLOG/TAG/BUSINESS-INSIGHTS)

RITTMAN MEAD LIFE (/BLOG/TAG/RITTMAN-MEAD-LIFE)

Recent Posts

- OA Summit 2020: OA Roadmap Summary (/blog/2020/06/oa-summit-2020-oracle-analytics-roadmap-summary/)
- Data Virtualization: What is it About? (/blog/2020/06/data-virualization-what-is-it/)
- Getting Smart View to work with OAC (/blog/2020/05/getting-smart-view-to-work-with-oac/)
- Oracle Analytics: Everything you always wanted to know (But were afraid to ask) (/blog/2020/02/oracle-analytics-everything-you-always-wanted-to-know-but-were-afraid-to-ask/)
- Oracle Data Science - Accelerated Data Science SDK Configuration (/blog/2020/02/accelerated-data-science-sdk-configuration/)

Sign Up for Our Newsletter

email address

SUBSCRIBE

(/blog/2016/11/whatever-next-the-battle-to-keep-up-with-changes-in-technology/)

READ THIS NEXT

Whatever next? The battle to keep up with changes in technology

I've been managing Rittman Mead's Training services in the UK & Europe for over a year now,…

(/blog/2016/11/connecting-oracle-data-visualization-desktop-to-google-analytics-and-google-drive/)

YOU MIGHT ENJOY

Connecting Oracle Data Visualization Desktop to Google Analytics and Google Drive

To use Data Visualisation Desktop (DVD) with data from Google Analytics or Google Drive, you need to set up…

About Us

Rittman Mead is a data and analytics company who specialise in data visualisation, predictive analytics, enterprise reporting and data engineering.

(http//www.rittmanmead.com/feed/) (http//twitter.com/rittmanmead)

(https//www.facebook.com/rittmanmead/) (http//www.linkedin.com/company/rittman-mead)

Contact Us

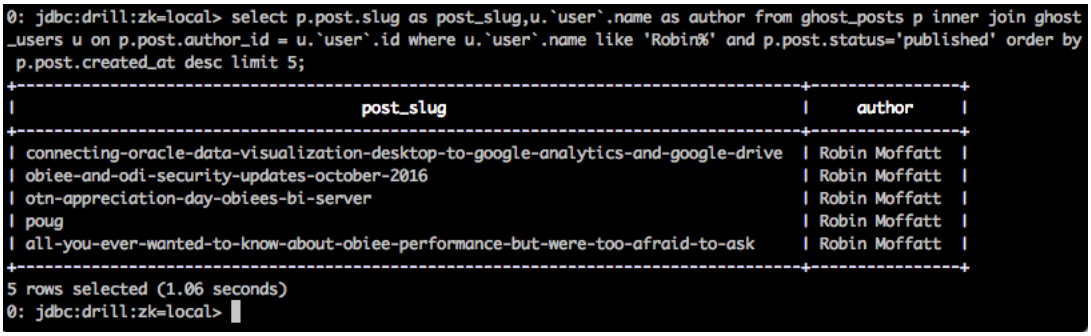
Rittman Mead Consulting Ltd.
Platf9rm, Hove Town Hall
Tisbury Road,
Brighton, BN3 3BQ
United Kingdom

Tel: (Phone) +44 1273 053956
Email: (Email)info@rittmanmead.com (mailto:info@rittmanmead.com)

```
0: jdbc:drill:zk=local> select p.post.slug as post_slug,u.`user`.name as author from ghost_posts p inner join ghost_users u on p.post.a
uthor_id = u.`user`.id where u.`user`.name like 'Robin%' and p.post.status='published' order by p.post.created_at desc limit 5;
+------------------------------------------------------------------------------+----------------+
|                                  post_slug                                   |     author     |
+------------------------------------------------------------------------------+----------------+
| connecting-oracle-data-visualization-desktop-to-google-analytics-and-google-drive | Robin Moffatt  |
| obiee-and-odi-security-updates-october-2016                                  | Robin Moffatt  |
| otn-appreciation-day-obiees-bi-server                                        | Robin Moffatt  |
| poug                                                                         | Robin Moffatt  |
| all-you-ever-wanted-to-know-about-obiee-performance-but-were-too-afraid-to-ask | Robin Moffatt  |
+------------------------------------------------------------------------------+----------------+
5 rows selected (1.06 seconds)
```

This is pretty cool. From a 32MB single-row JSON file:

to being able to query it with standard SQL like this:



all with a single tool that can run on a laptop or desktop, and supports ODBC and JDBC (https://drill.apache.org/docs/interfaces-introduction/) for use with your favourite BI tools. For data exploration and understanding new datasets, Apache Drill really does rock!

(/blog/auth moffatt/)

Robin Moffatt (/blog/author/robin-moffatt/)

Read more posts (/blog/author/robin-moffatt/) by this author.

Yorkshire, UK *https://www.linkedin.com/in/robinmoffatt (https://www.linkedin.com/in/robinmoffatt)*

Share this Post