

< Getting Started

Tutorial Exercise 2 >

Tutorial Exercise 1

Ingest Structured Data

In this scenario, DataCo's business question is: *What products do our customers like to buy?* To answer this question, the first thought might be to look at the transaction data, which should indicate what customers actually do buy and like to buy, right?

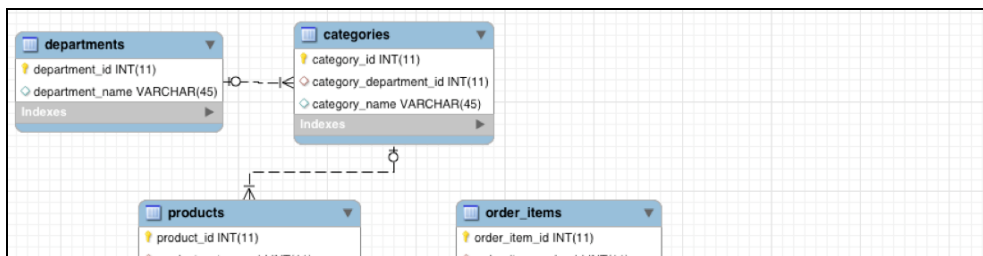
This is probably something you can do in your regular RDBMS environment, but a benefit with Cloudera's platform is that you can do it at greater scale at lower cost, on the same system that you may also use for many other types of analysis.

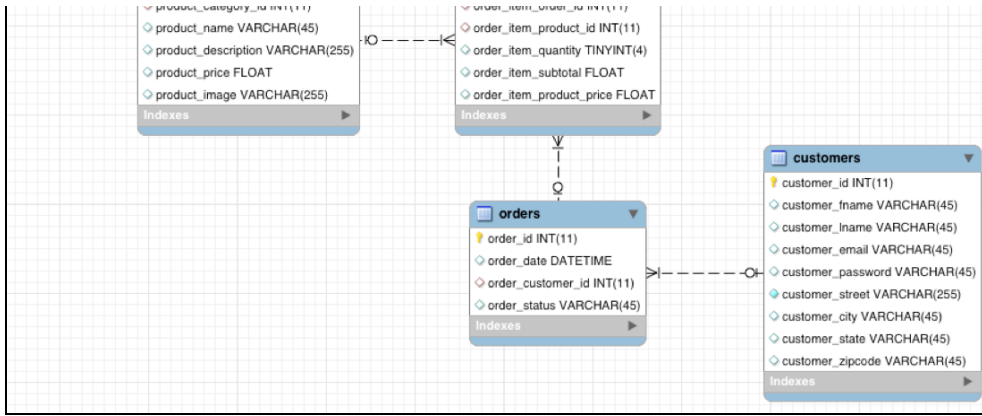
What this exercise demonstrates is how to do exactly the same thing you already know how to do, but in CDH. Seamless integration is important when evaluating any new infrastructure. Hence, it's important to be able to do what you normally do, and not break any regular BI reports or workloads over the dataset you plan to migrate.

About Sqoop:

Apache Sqoop is a tool that uses MapReduce to transfer data between Hadoop clusters and relational databases very efficiently. It works by spawning tasks on multiple data nodes to download various portions of the data in parallel. When you're finished, each piece of data is replicated to ensure reliability, and spread out across the cluster to ensure you can process it in parallel on your cluster.

There are 2 versions of Sqoop included in Cloudera's platform. Sqoop 1 is a "thick client" and is what you use in this tutorial. The command you run will directly submit the MapReduce jobs to transfer the data. Sqoop 2 consists of a central server that submits the MapReduce jobs on behalf of clients, and a much lighter weight client that you use to connect to the server. The "Sqoop" you see in Cloudera Manager is the Sqoop 2 server, although Cloudera Manager will make sure that both the "sqoop" and "sqoop2" command are correctly configured on all your machines.





To analyze the transaction data in the new platform, we need to ingest it into the Hadoop Distributed File System (HDFS). We need to find a tool that easily transfers structured data from a RDBMS to HDFS, while preserving structure. That enables us to query the data, but not interfere with or break any regular workload on it.

Apache Sqoop, which is part of CDH, is that tool. The nice thing about Sqoop is that we can automatically load our relational data from MySQL into HDFS, while preserving the structure.

With a few additional configuration parameters, we can take this one step further and load this relational data directly into a form ready to be queried by Impala (the open source analytic query engine included with CDH). Given that we may want to leverage the power of the Apache Avro file format for other workloads on the cluster (as Avro is a Hadoop optimized file format), we will take a few extra steps to load this data into Impala using the Avro file format, so it is readily available for Impala as well as other workloads.

You should first open a terminal, which you can do by clicking the black "Terminal" icon at the top of your screen. Once it is open, you can launch the Sqoop job:

[Copy](#)

```
[cloudera@quickstart ~]$ sqoop import-all-tables \
  -m 1 \
  --connect jdbc:mysql://quickstart:3306/retail_db \
  --username=retail_dba \
  --password=cloudera \
  --compression-codec=snappy \
  --as-avrodatafile \
  --warehouse-dir=/user/hive/warehouse
```

This command may take a while to complete, but it is doing a lot. It is launching MapReduce jobs to export the data from our MySQL database, and put those export files in Avro format in HDFS. It is also creating the Avro schema, so that we can easily load our Hive tables for use in Impala later.

Verification step:

When this command is complete, confirm that your Avro data files exist in HDFS.

[Copy](#)

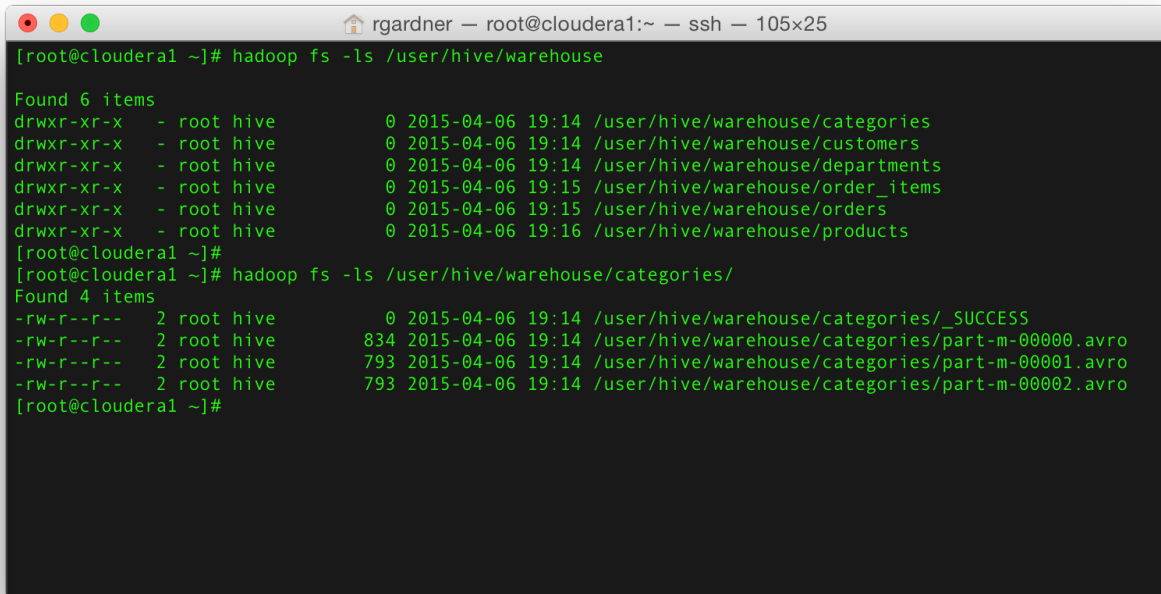
```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse
```

Will show a folder for each of the tables.

[Copy](#)

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/categories/
```

Will show the files that live inside of the categories folder.



```
rgardner — root@cloudera1:~ — ssh — 105x25
[root@cloudera1 ~]# hadoop fs -ls /user/hive/warehouse
Found 6 items
drwxr-xr-x - root hive      0 2015-04-06 19:14 /user/hive/warehouse/categories
drwxr-xr-x - root hive      0 2015-04-06 19:14 /user/hive/warehouse/customers
drwxr-xr-x - root hive      0 2015-04-06 19:14 /user/hive/warehouse/departments
drwxr-xr-x - root hive      0 2015-04-06 19:15 /user/hive/warehouse/order_items
drwxr-xr-x - root hive      0 2015-04-06 19:15 /user/hive/warehouse/orders
drwxr-xr-x - root hive      0 2015-04-06 19:16 /user/hive/warehouse/products
[root@cloudera1 ~]#
[root@cloudera1 ~]# hadoop fs -ls /user/hive/warehouse/categories/
Found 4 items
-rw-r--r--  2 root hive      0 2015-04-06 19:14 /user/hive/warehouse/categories/_SUCCESS
-rw-r--r--  2 root hive    834 2015-04-06 19:14 /user/hive/warehouse/categories/part-m-00000.avro
-rw-r--r--  2 root hive    793 2015-04-06 19:14 /user/hive/warehouse/categories/part-m-00001.avro
-rw-r--r--  2 root hive    793 2015-04-06 19:14 /user/hive/warehouse/categories/part-m-00002.avro
[root@cloudera1 ~]#
```

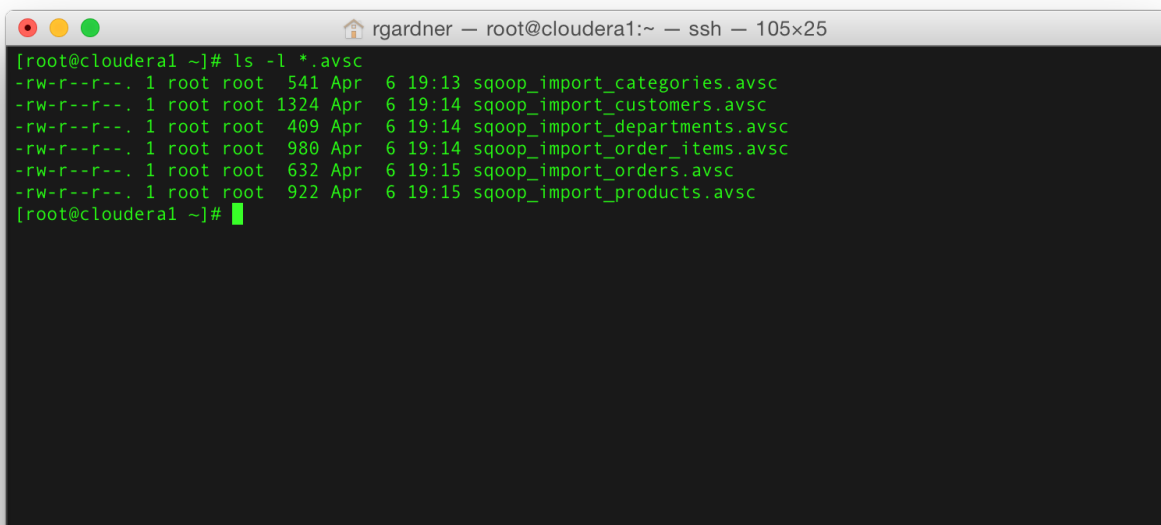
Sqoop should also have created schema files for this data in your home directory.

Avro schema files:

[Copy](#)

```
[cloudera@quickstart ~]$ ls -l *.avsc
```

Should show .avsc files for the six tables that were in our retail_db.



```
rgardner — root@cloudera1:~ — ssh — 105x25
[root@cloudera1 ~]# ls -l *.avsc
-rw-r--r--  1 root root   541 Apr  6 19:13 sqoop_import_categories.avsc
-rw-r--r--  1 root root  1324 Apr  6 19:14 sqoop_import_customers.avsc
-rw-r--r--  1 root root   409 Apr  6 19:14 sqoop_import_departments.avsc
-rw-r--r--  1 root root   980 Apr  6 19:14 sqoop_import_order_items.avsc
-rw-r--r--  1 root root   632 Apr  6 19:15 sqoop_import_orders.avsc
-rw-r--r--  1 root root   922 Apr  6 19:15 sqoop_import_products.avsc
[root@cloudera1 ~]#
```

Note that the schema and the data are stored in separate files. The schema is only applied when the data is queried, a technique called 'schema-on-read'. This gives you the flexibility to query the data with SQL while it's still in a format usable by other systems as well. Whereas a traditional database requires the schema to be defined before entering any data, we have already imported a lot of data and will only now specify how its structure should be interpreted.

Apache Hive will need the schema files too, so let's copy them into HDFS where Hive can easily access them.

[Copy](#)

```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir /user/examples
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -chmod +rw /user/examples
[cloudera@quickstart ~]$ hadoop fs -copyFromLocal ~/.avsc /user/examples/
```

Now that we have the data, we can prepare it to be queried. We're going to do this in the next section using Impala, but you may notice we imported this data into Hive's directories. Hive and Impala both read their data from files in HDFS, and they even share metadata about the tables. The difference is that Hive executes queries by compiling them to MapReduce jobs. As you will see later, this means it can be more flexible, but is much slower. Impala is an MPP query engine that reads the data directly from the file system itself. This allows it to execute queries fast enough for interactive analysis and exploration.

If one of these steps fails, please reach out to our Cloudera Live Forum (<http://community.cloudera.com/t5/Cloudera-Live/bd-p/ClouderaLive>) and get help.

CONCLUSION

Now you have gone through the first basic steps to Sqoop structured data into HDFS, transform it into Avro file format (you can read about the benefits of Avro as a common format in Hadoop here (<http://blog.cloudera.com/blog/2011/07/avro-data-interop/>)), and import the schema files for use when we query this data.

[< Getting Started](#)[Tutorial Exercise 2 >](#)

© 2015 Cloudera (<http://www.cloudera.com>), Inc. All rights reserved | Terms & Conditions (<http://www.cloudera.com/content/cloudera/en/terms-of-service.html>) | Privacy Policy (<http://www.cloudera.com/content/cloudera/en/privacy-policy.html>)

Hadoop and the Hadoop elephant logo are trademarks of the Apache Software Foundation.