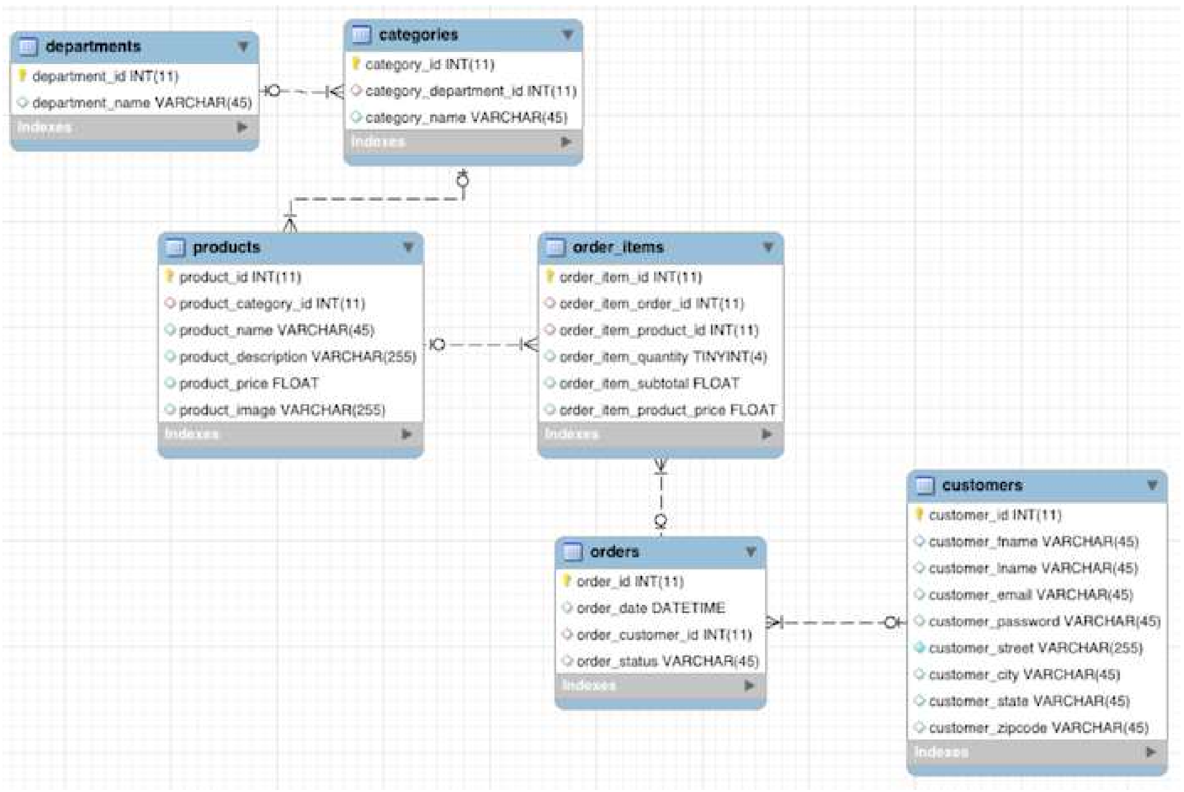
The "Getting Started with Hadoop" Tutorial

Exercise 1: Ingest and query relational 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 of Apache Hadoop 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.


departments
department_id INT(11)
department_name VARCHAR(45)
Indexes
categories
category_id INT(11)
category_department_id INT(11)
category_name VARCHAR(45)
Indexes
products
product_id INT(11)
product_category_id INT(11)
product_name VARCHAR(45)
product_description VARCHAR(255)
product_price FLOAT
product_image VARCHAR(255)
Indexes
order_items
order_item_id INT(11)
order_item_order_id INT(11)
order_item_product_id INT(11)
order_item_quantity TINYINT(4)
order_item_subtotal FLOAT
order_item_product_price FLOAT
Indexes
orders
order_id INT(11)
order_date DATETIME
order_customer_id INT(11)
order_status VARCHAR(45)
Indexes
customers
customer_id INT(11)
customer_fname VARCHAR(45)
customer_lname VARCHAR(45)
customer_email VARCHAR(45)
customer_password VARCHAR(45)
customer_street VARCHAR(255)
customer_city VARCHAR(45)
customer_state VARCHAR(45)
customer_zipcode VARCHAR(45)
Indexes

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 Apache Impala (incubating), the MPP analytic database included with CDH, and other workloads.

You should first log in to the Master Node of your cluster via a terminal. Then, launch the Sqoop job:

```
> sqoop import-all-tables \
    -m {{cluster_data.worker_node_hostname.length}} \
    --connect jdbc:mysql://{{cluster_data.manager_node_hostname}}:3306/
    --username=retail_dba \
    --password=cloudera \
    --compression-codec=snappy \
    --as-parquetfile \
    --warehouse-dir=/user/hive/warehouse \
    --hive-import
```

This command may take a while to complete, but it is doing a lot. It is launching MapReduce jobs to pull the data from our MySQL database and write the data to HDFS in parallel, distributed across the cluster in **Apache Parquet (http://parquet.apache.org)** format. It is also creating tables to represent the HDFS files in Impala/Apache Hive with matching schema.

Parquet is a format designed for analytical applications on Hadoop. Instead of grouping your data into rows like typical data formats, it groups your data into columns. This is ideal for many analytical queries where instead of retrieving data from specific records, you're analyzing relationships between specific variables across many records. Parquet is designed to optimize data storage and retrieval in these scenarios.

Verification

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

```
> hadoop fs -ls /user/hive/warehouse/
> hadoop fs -ls /user/hive/warehouse/categories/
```

These commands will show the directories and the files inside them that make up your tables:

Note: The number of .parquet files shown will be equal to what was passed to Sqoop with the -m parameter. This is the number of 'mappers' that Sqoop will use in its MapReduce jobs. It could also be thought of as the number of simultaneous connections to your database, or the number of disks / Data Nodes you want to spread the data across. So on a single-node you will just see one, but larger clusters will have a greater number of files.

Hive and Impala also allow you to create tables by defining a schema over existing files with 'CREATE EXTERNAL TABLE' statements, similar to traditional relational databases. But Sqoop already created these tables for us, so we can go ahead and query them.

We're going to use Hue's Impala app to query our tables. Hue provides a web-based interface for many of the tools in CDH and can be found on port 8888 of your Manager Node. In the QuickStart VM, the administrator username for Hue is 'cloudera' and the password is 'cloudera'.

Once you are inside of Hue, click on Query Editors, and open the Impala Query Editor.

To save time during queries, Impala does not poll constantly for metadata changes. So the first thing we must do is tell Impala that its metadata is out of date. Then we should see our tables show up, ready to be queried:

```
invalidate metadata;

show tables;
```

You can also click on the "Refresh Table List" icon on the left to see your new tables in the side menu.

Now that your transaction data is readily available for structured queries in CDH, it's time to address DataCo's business question. Copy and paste or type in the following standard SQL example queries for calculating total revenue per product and showing the top 10 revenue generating products:

```
-- Most popular product categories
select c.category_name, count(order_item_quantity) as count
from order_items oi
inner join products p on oi.order_item_product_id = p.product_id
inner join categories c on c.category_id = p.product_category_id
group by c.category_name
order by count desc
limit 10;
```

You should see results of the following form:

Clear out the previous query, and replace it with the following:

```
-- top 10 revenue generating products
select p.product_id, p.product_name, r.revenue
from products p inner join
(select oi.order_item_product_id, sum(cast(oi.order_item_subtotal as fl
from order_items oi inner join orders o
on oi.order_item_order_id = o.order_id
where o.order_status <> 'CANCELED'
and o.order_status <> 'SUSPECTED_FRAUD'
group by order_item_product_id) r
on p.product_id = r.order_item_product_id
order by r.revenue desc
limit 10;
```

You should see results similar to this:

You may notice that we told Sqoop to import the data into Hive but used Impala to query the data. This is because Hive and Impala can share both data files and the table metadata. Hive works by compiling SQL queries into MapReduce jobs, which makes it very flexible, whereas Impala executes queries itself and is built from the ground up to be as fast as possible, which makes it better for interactive analysis. We'll use Hive later for an ETL (extract-transform-load) workload.

Conclusion:

Now that you have gone through the first basic steps to Sqoop structured data into HDFS, transform it into Parquet file format, and create hive tables for use when you query this data.

You have also learned how to query tables using Impala and that you can use regular interfaces and tools (such as SQL) within a Hadoop environment as well. The idea here being that you can do the same reports you usually do, but where the architecture of Hadoop vs traditional systems provides much larger scale and flexibility.

GO TO NEXT STEP (/developers/get-started-with-hadoop-tutorial/showing-big-data-value.html) (https://www.linkedin.com/company/cloudera) (https://www.facebook.com/cloudera)

(https://twitter.com/cloudera) (/contact-us.html)

United States: +1 888 789 1488 (tel:18887891488)
Outside the US: +1 650 362 0488 (tel:16503620488)

English