Accessing Hive/Impala from Neo4j

Quite frequently, I get asked about how you could import data from Hadoop and bring it into Neo4j. More often than not, the request is about importing from Impala or Hive. Today, Neo4j isn't able to directly access an HDFS file system so we have to use a different approach.

For this example, we will use a Neo4j Unmanaged Extension. Unmanaged extensions provide us with the ability to hook into the native Java API and expand the capabilities of the Neo4j server. The unmanaged extension is reached via a REST API. With the extension, we can connect to Impala/Hive, query the database, return the data and then manipulate the data based on what we need.

Let's create a simple example of an Java unmanaged extension which exposes a Neo4j REST endpoint that connects to an Hive instance running on another machine, query the database and create nodes.

What is Cloudera Impala
A good overview of Impala is in this presentation on SlideShare. Impala provides interactive SQL, nearly ANSI-92 standard SQL queries and provides a JDBC connector allowing external tools to access Impala. Cloudera Impala is the industry's leading massively parallel processing (MPP) SQL query engine that runs natively in Apache Hadoop. The Apache-licensed, open source Impala project combines modern, scalable parallel database technology with the power of Hadoop, enabling users to directly query data stored in HDFS and Apache HBase without requiring data movement or transformation.

What is Cloudera Hive
From Cloudera, "Hive enables analysis of large data sets using a language very similar to standard ANSI SQL. This means anyone who can write SQL queries can access data stored on the Hadoop cluster. This discussion introduces the functionality of Hive, as well as its various applications for data analysis and data warehousing."

Installing a Hadoop Cluster
For this demonstration, we will use the Cloudera QuickStart VM running on VirtualBox. You can download the VM and run it in VMWare, KVM and VirtualBox. I chose VirtualBox. The VM gives you access to a working Impala and Hive instance without manually installing each of the components.

Connecting to Impala/Hive
Cloudera provides a download page for the JDBC driver. Once you download the jdbc driver, you will want to unzip it and copy the jar files from the Cloudera_ImpalaJDBC4_2.5.5.1007 directory into your plugins directory.

Hortonworks
For Hortonworks, you can download a Sandbox for VirtualBox or VMWare. Once installed, you will need to set the IP address for the Hortonworks machine. The Hortonworks sandbox has the same sample data set in Hive.

Unmanaged Extension
Our unmanaged extension (code below), queries against the default database in Hive and creates an equipment node with two properties for each record in the sample_07 table. The query is hardcoded but could easily be passed in as a parameter with the POST request.

Deploying
1. Build the Jar
2. Copy target/neo4j-hadoop.jar to the plugins/ directory of your Neo4j server.
3. Copy the Cloudera JDBC jar files to your plugins directory.
4. Configure Neo4j by adding a line to conf/neo4j-server.properties:

```
org.neo4j.server.thirdparty_jaxrs_classes=com.neo4j.hadoop.example=/v1
```

5. Check that it is installed correctly over HTTP:

```
:GET /v1/service/helloworld
```

6. Query the database:

```
:POST /v1/service/equipment
```

7. Open the Neo4j browser and query the Equipment nodes:

```
MATCH (e:Equipment) return count(e);
```

Results in Neo4j

Summary

Cloudera's JDBC driver combined with Neo4J's Unmanaged Extension provides an easy and fast way to connect to a Hadoop instance and populate a Neo4J instance. While this example doesn't show relationships, you can easily modify the example to create the relationships.

Full code for the project is on Github.

Source Code for the example is below:
https://gist.github.com/davidfauth/af0cf23bb46e9a76cdab