Data Connectors in Hive

What is a Data connector?

Data connectors (referred to as "connector" in Hive Query Language) are top level objects in Hive where users can define a set of properties required to be able to connect to a datasource from hive. So a connector has a type (closed enumerated set) that allows Hive to determine the driver class (for JDBC) and other URL params, a URL and a set of properties that could include the default credentials for the remote datasource. Once defined, users can use the same connector object to map multiple databases from the remote datasource to local hive metastore.

With JDBC Storage Handlers, users define a table in hive metastore for which data resides in a remote JDBC datastore. This hive table's metadata is persisted locally in hive metastore's backend. When Hiveserver2 runs a query against this table, data is retrieved from the remote JDBC table. While this is very powerful in itself, it has limitations.

- Each remote table in remote datasource has to be individually mapped to a local hive table. It becomes tedious if you have to map an entire database of many tables.
- Any new tables in the remote datasource are not automatically visible and will need to be manually mapped in hive.
- The metadata for the mapped table is static. It does not track changes to the remote table. If remote table has changed (added columns or dropped columns), the mapped hive table has to be dropped and re-created. This management compounds when one has constantly changing tables.

Data connectors allow users to map a database/schema in the remote datasource to a Hive database. Such databases are referred to as REMOTE databases in hive.

```
CREATE REMOTE DATABASE postgres_db1 USING <connectorName> WITH DBPROPERTIES ('connector.
remoteDbName'='db1');
```

- All the tables within a mapped database (*db1*) are automatically visible from Hive. The metadata for these tables are not persisted in Hive metastore's backend. They are retrieved at runtime via a live connector.
- All future tables will automatically be visible in Hive.
- The columns and their datatypes are mapped to a compatible hive datatype at runtime as well. So any metadata changes to tables in the remote datasource are immediately visible in hive.
- The same connector can be used to map another database to a hive database. All the connection information is shared between these REMOTE databases.

How do I use it?

1. Create a connector first.
   ```
   CREATE CONNECTOR pg_local TYPE 'postgres' URL 'jdbc:postgresql://localhost:5432' WITH DCPROPERTIES
   ("hive.sql.dbcp.username"="postgres", "hive.sql.dbcp.password"="postgres");
   ```

2. Create a database of type REMOTE in hive using the connector from Step1. This maps a remote database named "*hive_hms_testing*" to a hive database named "*pg_hive_testing*" in hive.

   ```
   CREATE REMOTE DATABASE pg_hive_testing USING pg_local WITH DBPROPERTIES ("connector.remoteDbName"="
   hive_hms_testing");

   // USING keystore instead of cleartext passwords in DCPROPERTIES
   CREATE CONNECTOR pg_local_ks TYPE 'postgres' URL 'jdbc:postgresql://localhost:5432/hive_hms_testing'
   WITH DCPROPERTIES("hive.sql.dbcp.username"="postgres","hive.sql.dbcp.password.keystore"="jceks://app/local/hive
   /secrets.jceks" "hive.sql.dbcp.password.key"="postgres.credential");
   CREATE REMOTE DATABASE pg_ks_local USING pg_local_ks("connector.remoteDbName"="hive_hms_testing");
   ```

3. Use the tables in REMOTE database much like the JDBC-storagehandler based tables in hive. One big difference is that the metadata for these tables are never persisted in hive. Currently, create/alter/drop table DDLs are not supported in REMOTE databases.

   ```
   USE pg_hive_testing;
   SHOW TABLES;
   DESCRIBE [formatted] <tablename>;
   SELECT <col1> from <tablename> where <filter1> and <filter2>;
   ```