Creating Tables with Avro and Parquet Files

When you create tables using files in Apache Avro or Apache Parquet formats, there are additional options specific to those file formats.

Apache Avro

To create an Avro table (that is, a table that will use the Avro format) specify STORED AS AVRO.

Avro embeds a schema definition (column names and their data types) in the file itself, but you still must provide a schema definition in your CREATE TABLE statement. There are three ways to do this:

1. The usual way of specifying columns and their data types
2. Specifying an Avro schema file
3. Providing the schema as a JSON literal string

To create an Avro table in the usual way, provide the name and data type of each column, in parentheses, with the name-type pairs separated by commas:

```
CREATE TABLE company_email_avro (id INT, name STRING, email STRING)

  STORED AS AVRO;
```

You can avoid listing the full schema in the CREATE TABLE statement by storing the table schema in a separate Avro schema file and linking to it as a table property. In your CREATE TABLE statement, specify TBLPROPERTIES ('avro.schema.url'='/*path*/*to*/*file*.avsc'); where *file*.avsc is a JSON file containing an Avro schema.

For example, you can create the same table as above using order_details.avsc with these contents:

```
{"type":"record",
            "name":"company_email_avro",
            "fields":[
                        {"name":"id","type":["null","int"]},
                        {"name":"name","type":["null","string"]},
                        {"name":"email","type":["null","string"]}
            ]}
```

(The instances of "null" preceding the names of the data types in this schema indicate that the columns can contain NULL values.)

Then the CREATE TABLE statement could look like this:

```
CREATE TABLE company_email_avro
   STORED AS AVRO
   TBLPROPERTIES ('avro.schema.url'='/path/to/company_email.avsc');
```

The URL path can point to an Avro schema file in HDFS, as shown above. You can also point to other locations, such as an HTTP server using the http:// protocol or an S3 bucket using the appropriate protocol for your configuration, such as s3a://.

Instead of using the Avro schema file, you can put the JSON for the schema into the CREATE TABLE statement as a table property using 'avro.schema.literal'. The following will also create the same table as the previous examples:

```
CREATE TABLE company_email_avro
   STORED AS AVRO
   TBLPROPERTIES ('avro.schema.literal'=
    '{"type":"record",
       "name":"company_email_avro",
        "fields":[
                   {"name":"id","type":["null","int"]},
                   {"name":"name","type":["null","string"]},
                   {"name":"email","type":["null","string"]}
      ]}');
```

If the JSON schema of an Avro data file is not available to you, you can extract it from the Avro data file by using the avro-tools command-line utility. The syntax is:

avro-tools getschema */path/to/avro_data_file.avro*

This command extracts the schema from the specified Avro data file and prints the JSON representation of the schema to the screen. You could then copy and paste this JSON into the TBLPROPERTIES clause as described above, or save this JSON to a plain text file using the .avsc extension then point to that file in the TBLPROPERTIES clause, as previously described. This avro-tools utility is installed on the course VM.

Apache Parquet

Like with Avro tables, you can create Parquet tables by specifying STORED AS PARQUET. You saw this in the Week 2 reading about the STORED AS clause. Parquet files also have the schema embedded in the file, but the table definition also needs to include the schema definition, for example:

```
CREATE TABLE investors_parquet

      (name STRING, amount INT, share DECIMAL(4,3))

   STORED AS PARQUET;
```

With Parquet files, though, Impala *(not Hive)*provides a shortcut: LIKE PARQUET '*/path/to/file.parq*'. By using this shortcut, you can take the schema directly from a Parquet file—no need for a separate schema file as Avro requires:

```
CREATE TABLE investors_parquet

     LIKE PARQUET '/user/hive/warehouse/investors_parquet/investors.parq'

    STORED AS PARQUET;
```

Note that in this case, the table being created will use the specified file as one of its data files, because the table's file location will be /user/hive/warehouse/investors_parquet and the referenced Parquet file is in that location. This does *not* need to be the case; you can use a file that exists elsewhere than where the table's files will be stored.

Note also that this is not the same as cloning a table using LIKE *tablename*.(See the “CREATE TABLE Shortcuts” reading from Week 2.) Using LIKE *tablename* points to an existing table and copies the schema information for that table from the metastore; using LIKE PARQUET '*/path/to/file.parq*' points to the path of a specific Parquet file and copies the schema information from that file. You *may* use the Parquet file for an existing table, but in that case, it's probably better to use LIKE *tablename*.

Cloudera's documentation for CREATE TABLEincludes the following notes about using LIKE PARQUET:

- Each column in the new table has a comment stating that the SQL column type was inferred from a Parquet file.
- The file format of the new table defaults to text, as with other kinds of CREATE TABLE statements. To make the new table also use Parquet format, include the clause STORED AS PARQUET in the CREATE TABLE LIKE PARQUET statement.
- If the Parquet data file comes from an existing Impala table, currently, any TINYINT or SMALLINT columns are turned into INT columns in the new table. Internally, Parquet stores such values as 32-bit integers.

Try It!

Try creating the Avro tables in all three ways using the following commands, dropping the table each time before recreating it a different way. *(Be sure you use the EXTERNAL keyword so you don't delete the file when you drop the table.)*

1. Use the usual way:

```
CREATE EXTERNAL TABLE company_email_avro

     (id INT, name STRING, email STRING)

   STORED AS AVRO

   LOCATION 's3a://training-coursera2/company_email_avro/';
```

Use the DESCRIBE command to inspect the schema, then check sample data using your favorite method to do that (such as the Sample from the data source panel, or running a SELECT * query). Drop the table when you're done.

2. Use the JSON schema file at s3a://training-coursera2/company_email_avro.avsc. First take a look at the contents of that file (using a hdfs dfs -cat or aws s3 cp command for S3 content—you can't use Hue for this), then run this command:

```
CREATE EXTERNAL TABLE company_email_avro

   STORED AS AVRO

   LOCATION 's3a://training-coursera2/company_email_avro/'

   TBLPROPERTIES ('avro.schema.url'='

              s3a://training-coursera2/company_email_avro.avsc');
```

Use the DESCRIBE command to inspect the schema, then check sample data using your favorite method to do that (such as the Sample from the data source panel, or running a SELECT * query). Drop the table when you're done.

3. Finally, use the JSON literal:

```
CREATE EXTERNAL TABLE company_email_avro

   STORED AS AVRO

   LOCATION 's3a://training-coursera2/company_email_avro/'

   TBLPROPERTIES ('avro.schema.literal'=

  '{"type":"record",

     "name":"company_email_avro",

      "fields":[
    {"name":"id","type":["null","int"]},

    {"name":"name","type":["null","string"]},

    {"name":"email","type":["null","string"]}

  ]}');
```

Use the DESCRIBE command to inspect the schema, then check the sample data using your favorite method to do that (such as the Sample from the data source panel, or running a SELECT * query). Drop the table when you're done.

4. You probably created a Parquet table already, the investors_parquet table in the reading “The STORED AS Clause” in Week 2. If you dropped that table, or didn't create it, go back and do so now. (It was the last step.) Then use LIKE PARQUET as directed above to create a new table called investors_parquet_archive. (Note that you need to specify the Parquet data file, not the existing table.) Use DESCRIBE to check the schema, then check that this cloned table has no data. Keep the table, you will use it again later.