The LOCATION Clause

Unless otherwise specified, Hive and Impala store table data under the warehouse directory, which is by default the HDFS directory /user/hive/warehouse/. However, this may not be where you want to store some table data. For example, the data may already exist elsewhere in HDFS, and you may not have permission to move or copy it. If there's a lot of data, copying it into the warehouse directory would be inefficient compared to querying it in its current location. Even if there's not a lot of data, moving a copy to the warehouse directory means that the copy will be outdated as soon as changes are made to the original version.

The LOCATION clause allows you to create a table whose data is stored at a specified directory, which can be outside the warehouse directory. The syntax is simply:

```
LOCATION 'path/to/location/'
```

The example shown below creates a new table named jobs_training whose data resides in the HDFS directory /user/training/jobs/. The specified storage directory may already exist, but if it does not, Hive or Impala will create it.

```
CREATE TABLE jobs_training
    (id INT, title STRING, salary INT, posted TIMESTAMP)
LOCATION '/user/training/jobs_training/';
```

To specify a storage directory outside of HDFS, you will need a fully qualified path in the LOCATION clause, including a protocol at the beginning of the path. For example, to specify a storage directory in Amazon S3, you will typically need to use LOCATION 's3a://*bucket/folder/*'.

Do Not Confuse This with the EXTERNAL Keyword

In practice, the EXTERNAL keyword is often used in conjunction with the LOCATION clause to specify a storage directory outside the warehouse directory. But when you use the keyword EXTERNAL, that does *not* necessarily mean that the data is stored outside the warehouse directory. In fact, if you use the keyword EXTERNAL, but you do not specify a directory with the LOCATION clause, then Hive and Impala will store the table data under the warehouse directory. On the other hand, if you use the LOCATION clause without the keyword EXTERNAL, dropping the table could delete the data, even if it's stored outside the warehouse directory.

So think of EXTERNAL as meaning externally *managed* (that is, managed by some software other than Hive or Impala) and *not* meaning externally *stored*. Remember that the LOCATION clause, not the EXTERNAL keyword, determines where the table data is stored.

Try It!

1. If you haven't already, use the Hue File Browser or the data source panel to examine the /user/hive/warehouse directory. (Using the data source panel, you need to click the files icon rather than the database icon.) You should see directories for databases (with .db at the end) and for any test tables that you've created in the default database and didn't drop. (If you haven't created any tables yet, you should still see some directories corresponding to tables in the default database, such as customers and employees.) Look inside one of these table directories and note whether there is a file inside. If the table has data, it will have at least one file here.

Next, do the following to create and examine a table whose data does *not* go into that default directory.

2. Go to the Impala query editor, select default as the active database, and execute the following statement:

```
CREATE TABLE jobs_training
    (id INT, title STRING, salary INT, posted TIMESTAMP)
LOCATION '/user/training/jobs_training/';
```

3. Use the data source panel (so you can leave the query editor in the main panel) to find the table directory for this new table. Note that it's *not* in /user/hive/warehouse/ as the other tables in the default database are. It should be under /user/training/ instead. Check that the new jobs_training directory is empty, since the table was created without data and none has been inserted.

4. Insert some data into the new table using the following statement in the query editor:

```
INSERT INTO jobs_training
    VALUES (1,'Data Analyst',135000,'2016-12-21 15:52:03');
```

5. Look again inside the jobs_training directory and verify that a new file has been added. You might need to click the refresh button to refresh the display. You can take a look at the file to see that it's the data you just inserted.

6. Drop the table.

Now create a table to query some existing data that’s located in an S3 bucket. We've created a bucket and given read-only access to the VM. The same row of data you have been using in these tests is saved in a delimited text file in the S3 bucket named training-coursera, in a folder named jobs. The file uses the default delimiter (Control+A) so you do not need a ROW FORMAT clause.

7. Execute the following statement:

```
CREATE EXTERNAL TABLE jobs_s3
    (id INT, title STRING, salary INT, posted TIMESTAMP)
  LOCATION 's3a://training-coursera1/jobs/';
```

8. Verify that the table has one row of data, either using the data source panel or by executing a SELECT * query.

9. Drop the table. *Note:* Typically you need to use the EXTERNAL keyword to ensure you don't accidentally delete data when you drop the table. In this case, because you don't have write access to the bucket, you will not delete the data even if you forgot the EXTERNAL keyword. Still, it's a good practice to use EXTERNAL for data that other people might be relying on, too, so you don't risk destroying their work as well as your own!