SQL INSERT Statements

SQL INSERT statements, common for adding new rows of data to tables in RDBMSs, can be used with Hive and Impala. There are essentially two versions, INSERT INTO (which adds files without changing or deleting existing files) and INSERT OVERWRITE (which replaces existing files with new files).

If you did the “Try It!” sections in Week 2, you have used them to add a single row of data to a table you created. They can also be used to add multiple rows:

```
INSERT INTO tablename
    VALUES
        (row1col1value,row1col2value, … ),
        (row2col1value,row2col2value, … ),
        … ;
```

This can be helpful for testing a table, such as examining how it stores the data, especially with an unusual or edge case. However, as those exercises noted, in general this is an *antipattern* with Hive and Impala, and using it for data ingest in a production environment is a bad practice.

In general, files in HDFS are *immutable*—they cannot be directly modified. (A file in HDFS can be deleted, and it can be overwritten with a new version of the file, but in general a file in HDFS cannot be directly modified.) So, each time you run an INSERT statement, Hive or Impala creates a new file in the table’s storage directory to store the new data values given in the statement. So inserting data in small batches like this causes Hive or Impala to create many small files in the table’s storage directory. This is a problem.

The *small files problem* is a common problem in big data systems. Most of the tools and frameworks that run on a Hadoop cluster are designed to work with *large* files, not lots of small files. So if you have a lot of small files (anything less than about 64MB is considered small), operations such as queries in Hive and Impala become inefficient, and query performance is negatively affected. Each INSERT command creates a new file, and they will be quite small (for any amount of data that is reasonable to add using an INSERT command).

One corrective solution for this, once you find you have a lot of small files, is to rewrite the whole dataset using this command:

INSERT OVERWRITE *tablename* SELECT * FROM *tablename*;

You'll learn more about INSERT … SELECT statements in the next reading.

Note that the small files problem can arise from other uses as well. For example, images are typically separate files, and there is no easy way to combine them into one file, so a large number of images will most likely be stored as a large number of files, potentially small ones depending on the resolution and size of the image. The corrective solution described above doesn't really help in that case; there are other ways to work around that problem, but this is beyond the scope of this course. If you're interested in reading more about the small files problem, see the Cloudera blog post, The Small Files Problem.

Note: You might see the syntax INSERT INTO TABLE or INSERT OVERWRITE TABLE. The TABLE keyword is optional. You can include it or not, as you like, but it's helpful to know that both syntaxes are valid, in case you see the one you decide not to use.

You can read more about using INSERT in Hive and Impala using these links:

- Hive LanguageManual UDF, Inserting data into Hive Tables from queries
- Cloudera's Impala documentation, INSERT Statement