Input and Output Interfaces

Table of contents

1 Set Up

No HCatalog-specific setup is required for the HCatInputFormat and HCatOutputFormat interfaces.

Authentication

If a failure results in a message like "2010-11-03 16:17:28,225 WARN hive.metastore ... - Unable to connect metastore with URI thrift://..." in /tmp/<username>/hive.log, then make sure you have run "kinit <username>@FOO.COM" to get a kerberos ticket and to be able to authenticate to the HCatalog server.

2 HCatInputFormat

The HCatInputFormat is used with MapReduce jobs to read data from HCatalog managed tables.

HCatInputFormat exposes a new Hadoop 20 MapReduce API for reading data as if it had been published to a table. If a MapReduce job uses this InputFormat to write output, the default InputFormat configured for the table is used as the underlying InputFormat and the new partition is published to the table after the job completes. Also, the maximum number of partitions that a job can work on is limited to 100K.

2.1 API

The API exposed by HCatInputFormat is shown below.

To use HCatInputFormat to read data, first instantiate a `HCatTableInfo` with the necessary information from the table being read and then call setInput on the `HCatInputFormat`.

You can use the `setOutputSchema` method to include a projection schema, to specify specific output fields. If a schema is not specified, this default to the table level schema.

You can use the `getTableSchema` methods to determine the table schema for a specified input table.

```
/**
 * Set the input to use for the Job. This queries the metadata server with
 * the specified partition predicates, gets the matching partitions, puts
 * the information in the conf object. The inputInfo object is updated with
 * information needed in the client context
 * @param job the job object
 * @param inputInfo the table input info
 * @throws IOException the exception in communicating with the metadata server
 */
public static void setInput(Job job, HCatTableInfo inputInfo) throws IOException;

/**
 * Set the schema for the HCatRecord data returned by HCatInputFormat.
```

```
     * @param job the job object
     * @param hcatSchema the schema to use as the consolidated schema
     */
    public static void setOutputSchema(Job job,HCatSchema hcatSchema) throws Exception;

    /**
     * Gets the HCatalog schema for the table specified in the HCatInputFormat.setInput
 call
     * on the specified job context. This information is available only after
 HCatInputFormat.setInput
     * has been called for a JobContext.
     * @param context the context
     * @return the table schema
     * @throws Exception if HCatInputFormat.setInput has not been called for the current
 context
     */
    public static HCatSchema getTableSchema(JobContext context) throws Exception
```

3 HCatOutputFormat

HCatOutputFormat is used with MapReduce jobs to write data to HCatalog managed tables.

HCatOutputFormat exposes a new Hadoop 20 MapReduce API for writing data to a table. If a MapReduce job uses this OutputFormat to write output, the default OutputFormat configured for the table is used as the underlying OutputFormat and the new partition is published to the table after the job completes.

3.1 API

The API exposed by HCatOutputFormat is shown below.

The first call on the HCatOutputFormat must be `setOutput`; any other call will throw an exception saying the output format is not initialized. The schema for the data being written out is specified by the `setSchema` method. If this is not called on the HCatOutputFormat, then by default it is assumed that the the partition has the same schema as the current table level schema.

```

/**
     * Set the info about the output to write for the Job. This queries the metadata server
     * to find the StorageDriver to use for the table.  Throws error if partition is
 already published.
     * @param job the job object
     * @param outputInfo the table output info
     * @throws IOException the exception in communicating with the metadata server
     */
    public static void setOutput(Job job, HCatTableInfo outputInfo) throws IOException;

    /**
     * Set the schema for the data being written out to the partition. The
     * table schema is used by default for the partition if this is not called.
     * @param job the job object
     * @param schema the schema for the data
     * @throws IOException the exception
     */
    public static void setSchema(Job job, HCatSchema schema) throws IOException;

    /**
     * Gets the table schema for the table specified in the HCatOutputFormat.setOutput call
     * on the specified job context.
     * @param context the context
     * @return the table schema
     * @throws IOException if HCatOutputFormat.setOutput has not been called for the passed
 context
     */
    public static HCatSchema getTableSchema(JobContext context) throws IOException
```

3.2 Partition Schema Semantics

The partition schema specified can be different from the current table level schema. The rules about what kinds of schema are allowed are:

- If a column is present in both the table schema and the partition schema, the type for the column should match.
- If the partition schema has lesser columns that the table level schema, then only the columns at the end of the table schema are allowed to be absent. Columns in the middle cannot be absent. So if table schema is "c1,c2,c3", partition schema can be "c1" or "c1,c2" but not "c1,c3" or "c2,c3"
- If the partition schema has extra columns, then the extra columns should appear after the table schema. So if table schema is "c1,c2", the partition schema can be "c1,c2,c3" but not "c1,c3,c4". The table schema is automatically updated to have the extra column. In the previous example, the table schema will become "c1,c2,c3" after the completion of the job.
- The partition keys are not allowed to be present in the schema being written out.