HIVE-6109: Support custom location for External table in case of dynamic partitioning

Description: Currently when dynamic partitions are created by HCatalog, the underlying directories for the partitions are created in a fixed 'Hive-style' format, i.e. root_dir/key1=value1/key2=value2/.... and so on. However in case of external table, user should be able to control the format of directories created for dynamic partitions.

E.g. suppose a table user_logs is partitioned by (year, month, day, country) and stored at location "hdfs://hcat/data/user_logs". User should be able to specify the format of locations where dynamic partitions should be created.

Approach:

- The approach taken by the patch is to allow user to provide the custom location path in the form of 'patterns'.
- Each dynamic partition column must be present in the custom location path in the following format:
  - ${column_name}
- The custom location path must consist of all dynamic partition columns.
- In the above example of table user_logs, following are some valid custom path strings:
  - "data/${year}/${month}/${day}/${country}"
  - "${year}-${month}-${day}/country=${country}"
  - "output/yr=${year}/mon=${month}/day=${day}/geo=${country}"

For this purpose, a new API is exposed in OutputJobInfo:

public void setCustomDynamicLocation(String customDynamicRoot, String customDynamicPath)

The argument customDynamicRoot is the relative path w.r.t. table location that acts as the root of all dynamic partitions. It is optional and can be null, in which case table location becomes the parent path.

E.g. for table user_logs, if this API is called as:
setCustomDynamicLocation("data", "${year}/${month}/${day}/${country}")

then the expected final output of dynamic partitions would be something like:

- hdfs://hcat/data/user_logs/data/2013/10/20/US
- hdfs://hcat/data/user_logs/data/2014/01/14/UK

Discovery of custom dynamic partition locations:

The next important question is that during commit phase, how do we discover which are the dynamic partitions that have got created? Till now, the paths of dynamic partitions were always created in a fixed format

“key1=value1/key2=value2/…”. So it was easy to parse the created paths backwards and extract the key-value pairs of dynamic partitions. However in case of custom paths, it is not mandatory that column names will even be there. It is also not possible to figure out the values of dynamic partition columns just by looking at the created path string.

For this purpose, the custom dynamic partition paths are created in a special intermediate format that helps us in performing discovery later during commit phase.

In FileRecordWriterContainer.write() method, the dynamic partition values are used to resolve the custom path pattern string. However the resolution just replaces the column name by its value in the custom path pattern – it still retains the enclosing ${…} matching groups.

E.g. for a particular dynamic partition, the intermediate resolved path that is created on HDFS would be:
hdfs://hcat/data/user_logs/data/_DYN0.5667352602407899/${2014}/{10}/{20}/${US}

This resolution is performed by util methods in the class **HCatFileUtil.java**.

Later during commit phase, all dynamic paths are searched by performing glob search, where the column name in custom path pattern is replaced by ‘*’. In this way, for each dynamic partition created, we have 2 strings: the user-provided custom path pattern that contains column names within ${..}, and the HDFS path that contains column values within ${…} in the same order as the pattern! We now simply parse these 2 strings in one pass and get the map of all dynamic partition name-value pairs. These pairs are then used to create the partitions and register them.

Registration of custom dynamic partitions:

In the registration step we need to calculate the final destination of each custom dynamic partition path and use that to register inside HCatalog partition as well as in moving data. For this purpose, we again resolve the intermediate custom path but this time we enable a flag to skip the regex pattern. Also the parent for final destination path is calculated using table location as well as custom dynamic root. Hence the final destination string would come out to be:

hdfs://hcat/data/user_logs/data/2013/10/20/US

The data is then moved from intermediate dynamic paths to final dynamic destinations.