Carbondata Compaction

Objective:

Design a methodology multiple segments of Carbondata files in memory into smaller number of segments, either based on size or count constraints as supplied by the user. The Compaction operation is valid for both partitioned tables and simple non-partitioned tables.

Outcome:

After the Compaction operation (aka VACUUM), the number of segments maintained in memory for the data will be reduced.

The following figure shows the structure of the files stored in memory after the Compaction operation is completed-

```
Found 6 items
drwxr-xr-x   - root supergroup          0 2020-06-08 20:32 /user/hive/warehouse/carbon.store/customer/customer_rec_2/Fact/Part0/Segment_0
drwxrwxrwx   - root supergroup          0 2020-06-22 14:42 /user/hive/warehouse/carbon.store/customer/customer_rec_2/Fact/Part0/Segment_0.1
drwxr-xr-x   - root supergroup          0 2020-06-08 20:37 /user/hive/warehouse/carbon.store/customer/customer_rec_2/Fact/Part0/Segment_1
drwxr-xr-x   - root supergroup          0 2020-06-08 20:37 /user/hive/warehouse/carbon.store/customer/customer_rec_2/Fact/Part0/Segment_2
drwxr-xr-x   - root supergroup          0 2020-06-08 20:37 /user/hive/warehouse/carbon.store/customer/customer_rec_2/Fact/Part0/Segment_3
drwxr-xr-x   - root supergroup          0 2020-06-08 20:55 /user/hive/warehouse/carbon.store/customer/customer_rec_2/Fact/Part0/Segment_4
```

Segment_0.1 is the compacted segment and the others are the original segments before compaction.

Types of Compaction:

There are two types of compaction that may be triggered based on the requirement-

1. Major Compaction: Here, the segments are merged based on their sizes. The default value is 1GB, therefore as long as the total size of all existing segments for the table is less than 1GB, the segments can be merged into a new single segment. This value can also be changed by the user parameter, supplied in the file carbondata.properties- "*carbondata.major-compaction-seg-size*"
2. Minor Compaction: Here, the segments are merged based on the count provided. The default value is 4, which means that the segments are merged in groups of 4. This can be changed as well based on the count provided by the user, as follows, in the file carbondata.properties- *"carbondata.minor-compaction-seg-count"*

The values can be provided as such-

```
carbondata.minor-compaction-seg-count = 6
carbondata.major-compaction-seg-size = 2
```

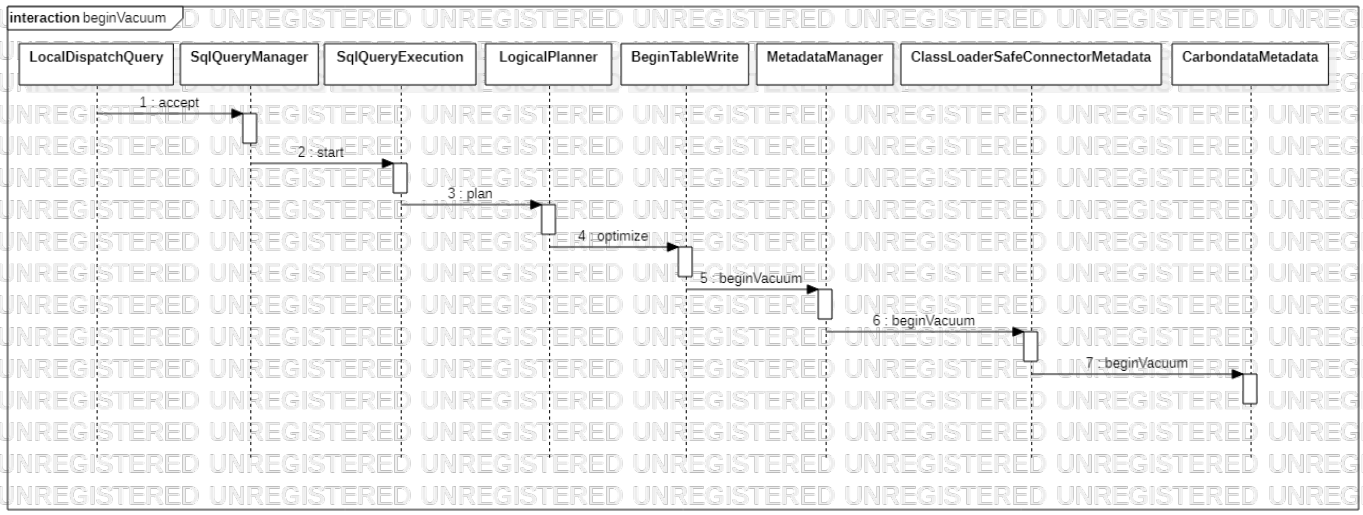
Query Details:

The following query is submitted to the client to start the compaction-

1. For Major Compaction- **VACUUM TABLE table_name FULL;**
2. For Minor Compaction- **VACUUM TABLE table_name;**

Design Diagram:

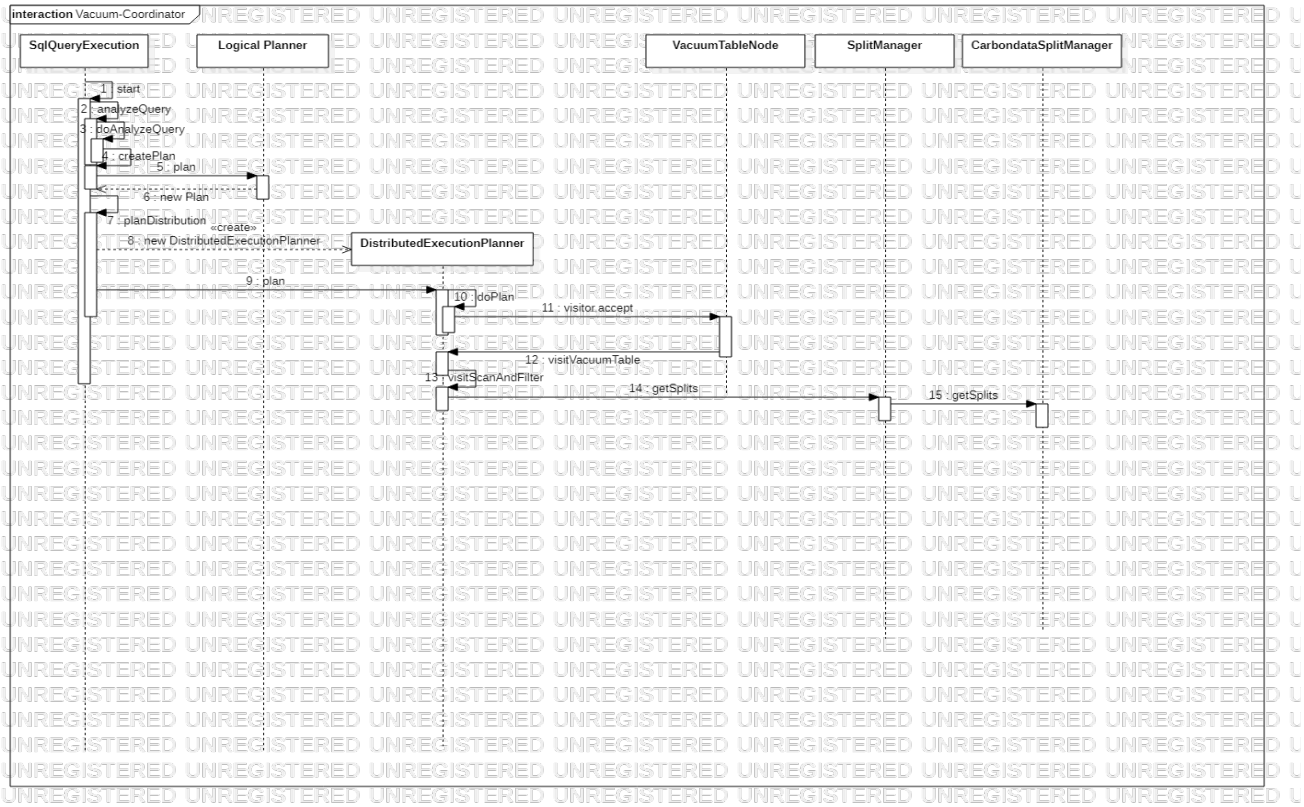
The following diagram highlights the steps followed to call the beginVacuum( ) method. Since, VACUUM is a new query type, not part of the ANSI standard, we modify the SqlBase.g4 file and ensure that the command is interpreted as a new query. Once, the query has been parsed correctly, it is sent to CarbondataMetadata which sets the context and initial bootstrapping that will be used for the query.



In terms of code, beginVacuum() has the following signature-

```
@Override
public CarbondataVacuumTableHandle beginVacuum(ConnectorSession session, ConnectorTableHandle tableHandle, boolean full, Optional<String> partition) throws PrestoException
{
    currentState = State.VACUUM;
    HiveInsertTableHandle insertTableHandle = super.beginInsert(session, tableHandle);
    return hdfsEnvironment.doAs(session.getUser(), () -> {
        SchemaTableName tableName = insertTableHandle.getSchemaTableName();
        Optional<Table> table =
                this.metastore.getTable(tableName.getSchemaName(), tableName.getTableName());
        this.table = table;
```

CarbondataSplitManager.java calls the method getSplits ( ) which collects all the splits which are to be compacted and sent to the workers for compaction. The following design diagram shows the steps involved-



Finally, the compaction is performed in the performCompaction( ) method called from vacuum( ) method inside CarbondataPageSink.java, which runs on the side of the Workers in the cluster. The signature for the method is as follows-

```
public void performCompaction(CarbondataPageSource connectorPageSource, HiveSplit split) throws PrestoException
{
    HdfsEnvironment hdfsEnvironment = connectorPageSource.getHdfsEnvironment();
    hdfsEnvironment.doAs(session.getUser(), () -> {
        try {
            // Worker part: each thread to run this code
            boolean mergeStatus = false;
            List<CarbonInputSplit> splitList = convertAndGetCarbonSplits(split);

            CarbonTable carbonTable = connectorPageSource.getCarbonTable();
            String databaseName = carbonTable.getDatabaseName();
            String factTableName = carbonTable.getTableName();
```