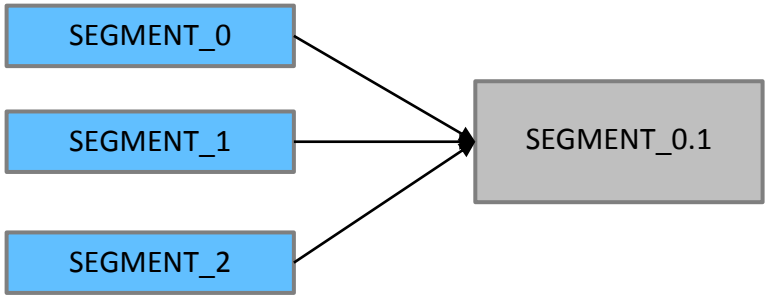
# CarbonData Compaction

# Why Compaction?

Frequent data loading into the Carbon table results in large number of Segments. As the number of segments increases, the number of B-trees also increase. This can result in the degradation of the performance of the query on the Carbon table.

# What is Compaction?

Compaction is a process in which the 2 or more segments of the Carbon table are merged into a single segment. i.e 2 or more B-trees will combine into 1 single B-tree so as to increase the query performance.

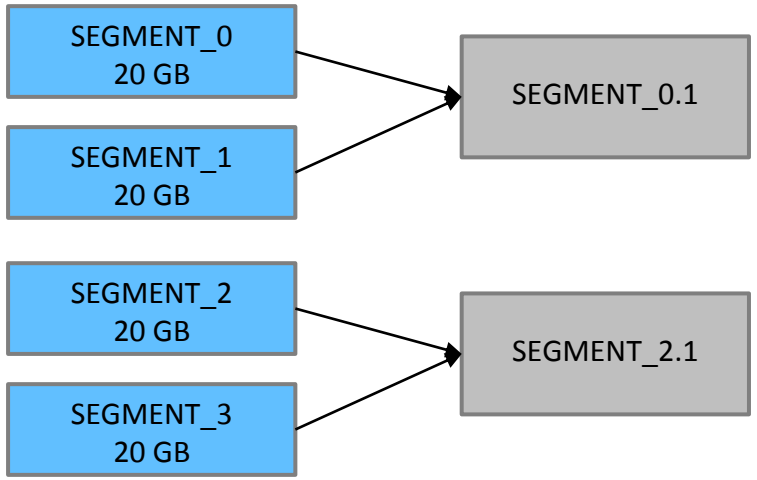



**Syntax:** ***ALTER TABLE*** *[db_name.]table_name COMPACT 'MINOR/MAJOR';*

# Types of Compaction - Major

This is a size based compaction in which, User will specify the compaction size until which segments can be merged. Major compaction is usually done during the off-peak time. Here the size means the size of the files inside the segment.

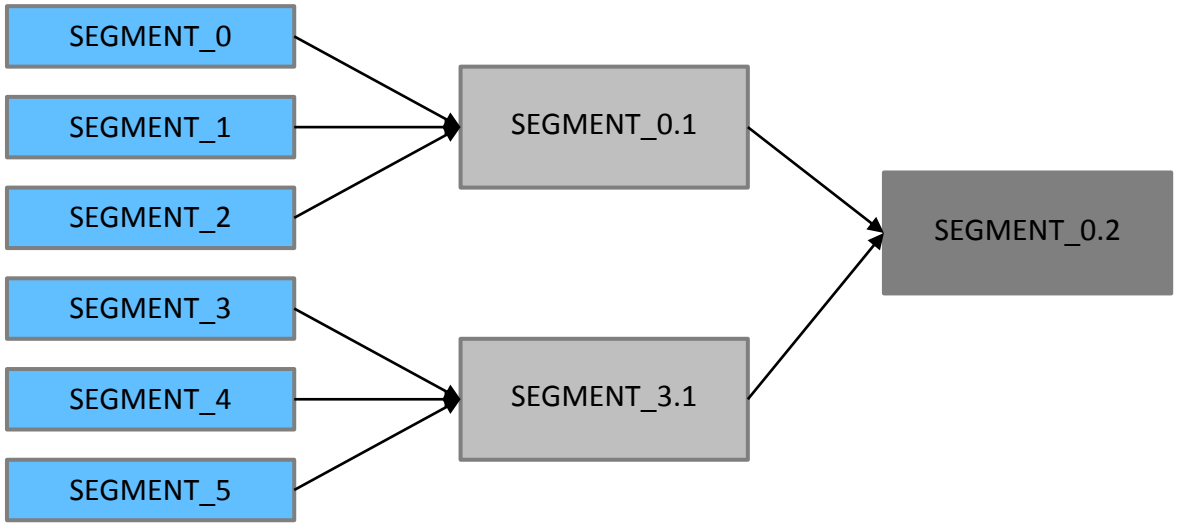
**Example:**
If the user has defined the Major compaction size as 40 GB. This means when the major compaction DDL is run then all the segments which are present will be merged into the segments of 40 GB size.



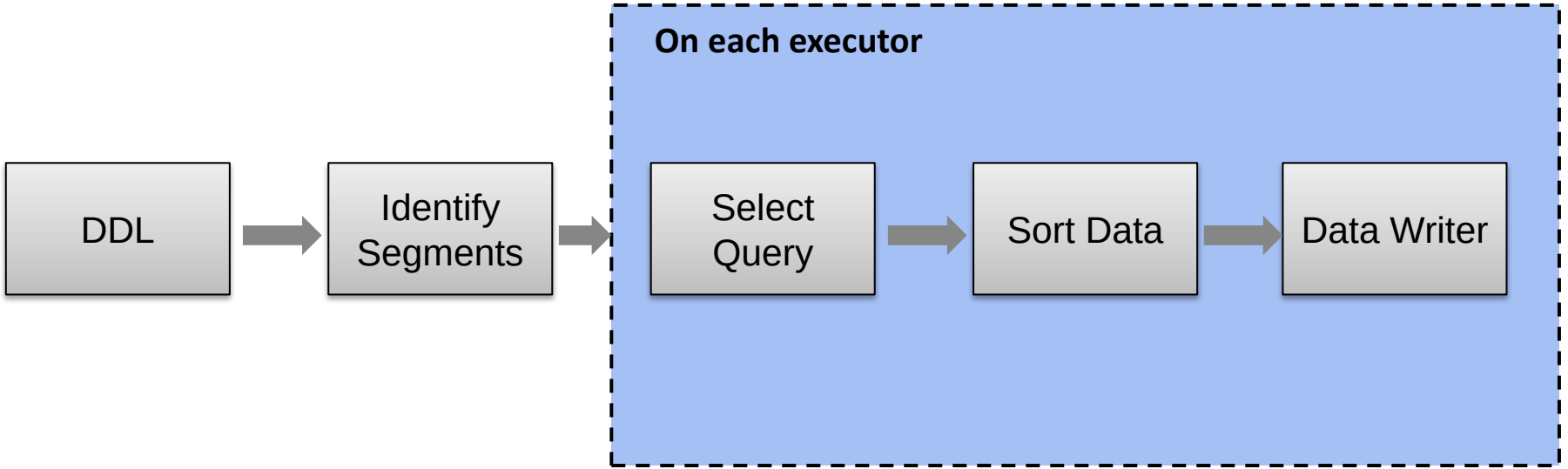
# Types of Compaction - Minor

In minor compaction, the user can give the segment count based on which compaction will take place.
Minor compaction is a 2 level process. User can configure the threshold parameter like 3,2.

**Example:**
If the threshold is 3,2 then the working of compaction will be as below.



# Working – Flow Diagram




**Note:** Auto load merge property is also present in carbon which will trigger the compaction automatically after the data load.
**carbon.enable.auto.load.merge = true**

# Working

1. Based on the user configured values for compaction the identification of segments to be compaction will happen in **DataManagementFunc.scala**
2. The task based grouping will happen, and for each executor a set of tasks will be allocated to be compacted. The executor will fire a select query on the allocated blocks of data. This is a special type of select query where the data will be in form of RawData.
   Refer to: **CarbonCompactionExecutor.java**
3. The result of the select query will be a List of RawResultIterator. This data needs to be sorted and is done using the priorityQueue. And each sorted data is sent to the next step which is CarbonDataWriter step.
   Refer to: **RowResultMerger.java**
4. Sorted data should be written to the new compacted segment folder and this will reuse the data load process writing flow.
   Refer to: **CarbonFactHandler.java**

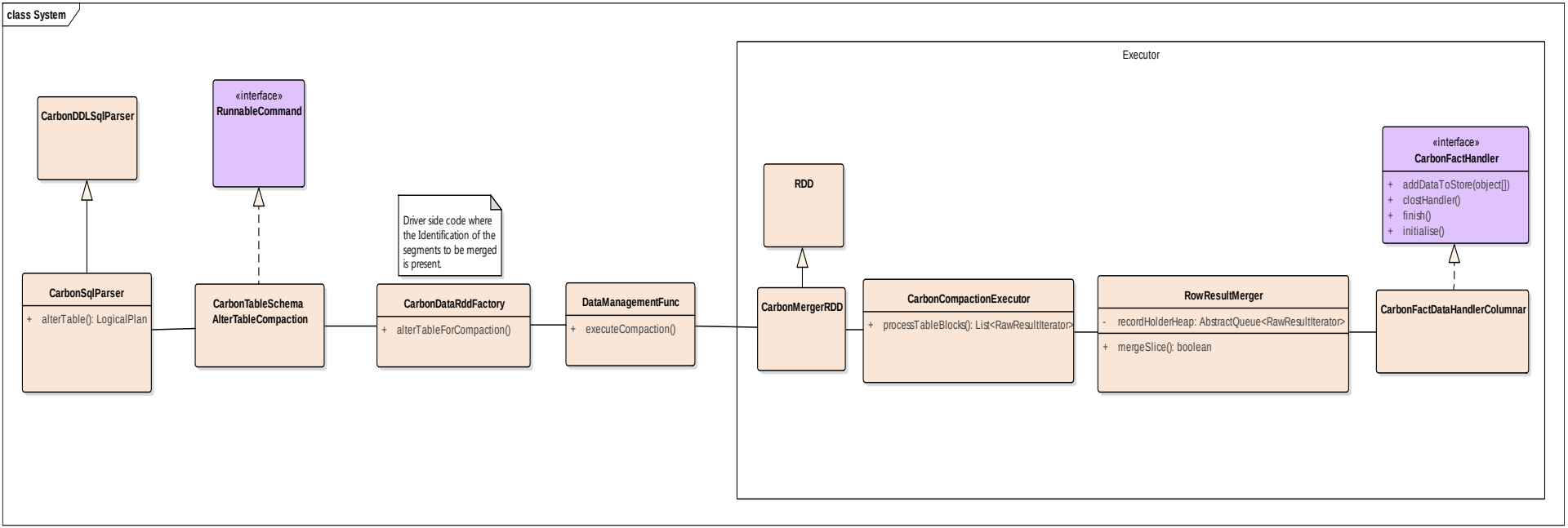
```
public interface CarbonFactHandler {
  void initialise() throws CarbonDataWriterException;

  void addDataToStore(Object[] row) throws CarbonDataWriterException;

  void finish() throws CarbonDataWriterException;

  void closeHandler() throws CarbonDataWriterException;
}
```

# Compaction – Class System



Thank you