LSM tree is based on 3 important concepts to optimize read and writes

1. SSTables: The fastest way to write a data is to append. Appending is efficient. But then to access a data we need O(n) time. That is not good for large databases. If the data is sorted then we will need O(logn) time which is desirable. SSTable provides this efficiency. Sorted String Table aka SSTable is a data structure which is sorted according to key. It stores key value pairs and sort them with keys. It is immutable.
2. Memtable: Memtable is an in-memory data structure. It stores data in sorted fashion. It acts as a write cache. First data is written here. After reaching a certain threshold size, it is flushed as a SSTable in database i.e memory. AVL tree, Red-Black tree (self-balancing trees) can be used as memtable. Advantages of memtable:
   1. If we write directly to database, we need strong bandwidth, more io operations, repeated connection with database.
   2. By using memtable, we can overcome these requirements. We write to memtable which is very fast as it is in-memory. We sometimes need connection with database. So, it is a lot efficient. Though additional memory is needed, it is not a big overhead today.
3. Compaction: It is an algorithm running in background. It compacts SSTables and rearranges the indices. Why do we need it?

We know that SSTable is immutable. If we keep flushing as SSTables in disks there may be several same key values present. It is a wastage of space. So, we need to delete the same keys except the last inserted one. We also then need to rearrange the indices of data. Compaction comes handy here. It runs in the background merging SSTables, removing redundant keys thus creates a compacted merge tree SSTables.

It frees up space and makes performance better. Compaction occurs in every 20-30 mins in case of large databases.