Compaction

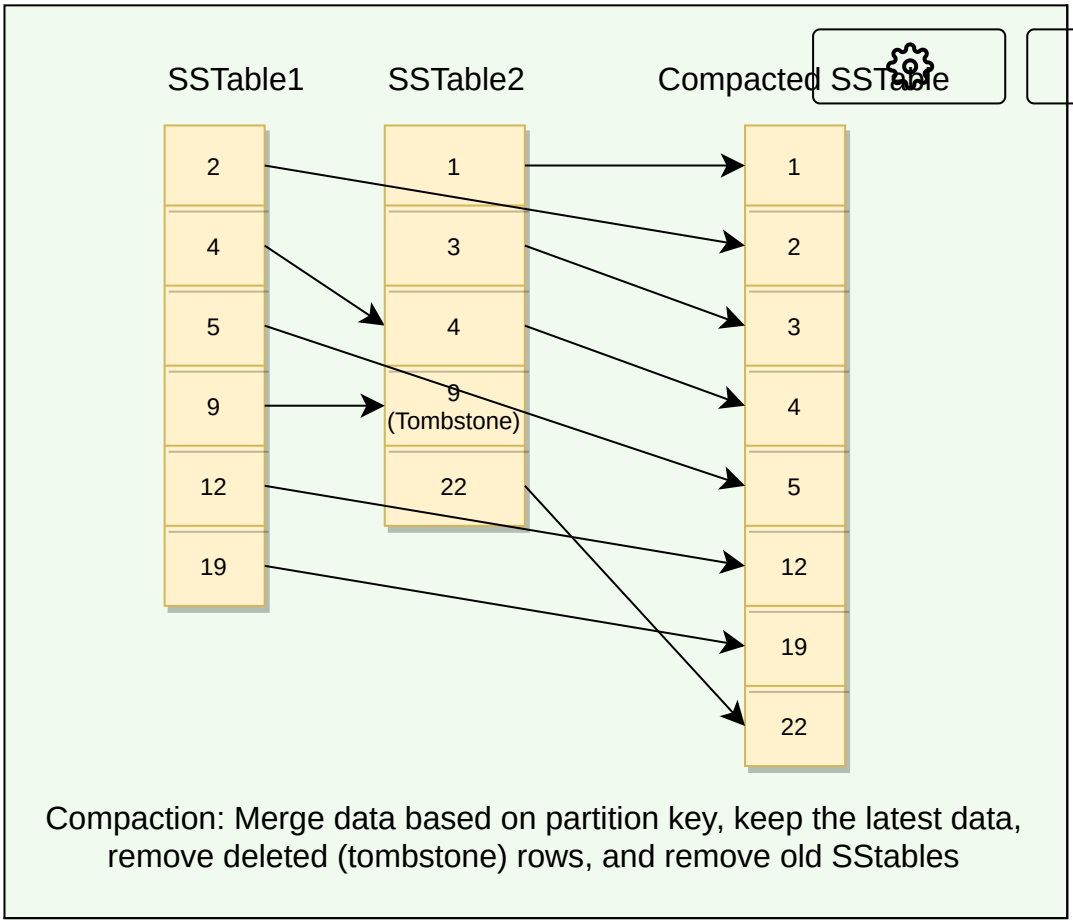
Let's explore how Cassandra handles compaction.

We'll cover the following

How does compaction work in Cassandra?#

As we have already discussed, SSTables are immutable, which helps Cassandra achieve such high write speeds. Flushing of MemTable to SStable is a continuous process. This means we can have a large number of SStables lying on the disk. While reading, it is tedious to scan all these SStables. So, to improve the read performance, we need compaction. Compaction in Cassandra refers to the operation of merging multiple related SSTables into a single new one. During compaction, the data in SSTables is merged: the keys are merged, columns are combined, obsolete values are discarded, and a new index is created.

On compaction, the merged data is sorted, a new index is created over the sorted data, and this freshly merged, sorted, and indexed data is written to a single new SSTable.



Compacting two SSTables into one

- Compaction will reduce the number of SSTables to consult and therefore improve read performance.
- Compaction will also reclaim space taken by obsolete data in SSTable.

Compaction strategies#

SizeTiered Compaction Strategy: This compaction strategy is suitable for insert-heavy and general workloads. This is the default compaction strategy and is triggered when multiple SSTables of a similar size are present.

Leveled Compaction Strategy: This strategy is used to optimize read performance. This strategy groups SSTables into levels, each of which has a fixed size limit which is ten times larger than the previous level.

Time Window Compaction Strategy: The Time Window Compaction Strategy is designed to work on time series data. It compacts SSTables within a configured time window. This strategy is ideal for time series data which is immutable after a fixed time interval.

Sequential writes#

Sequential writes are the primary reason that writes perform so well in Cassandra. No reads or seeks of any kind are required for writing a value to Cassandra because all writes are ‘append’ operations. This makes the speed of the disk one key limitation on performance. Compaction is intended to amortize the reorganization of data, but it uses sequential I/O to do so, which makes it efficient. If Cassandra naively inserted values where they ultimately belonged, writing clients would pay for seeks upfront.

← Back

Next →

Anatomy of Cassandra's Read Operati...

Tombstones

Mark as Completed

Report an Issue