Tombstones

Let's explore how Tombstones work in Cassandra.

We'll cover the following

What are Tombstones?#

An interesting case with Cassandra can be when we delete some data for a node that is down or unreachable, that node could miss a delete. When that node comes back online later and a repair occurs, the node could “resurrect” the data that had been previously deleted by re-sharing it with other nodes. To prevent deleted data from being reintroduced, Cassandra uses a concept called a tombstone. A tombstone is similar to the idea of a “**soft delete**” from the relational database world. When we delete data, Cassandra does not delete it right away, instead associates a tombstone with it, with a time to expiry. In other words, a tombstone is a marker that is kept to indicate data that has been deleted. When we execute a delete operation, the data is not immediately deleted. Instead, it’s treated as an update operation that places a tombstone on the value.

Each tombstone has an expiry time associated with it, representing the amount of time that nodes will wait before removing the data permanently. By default, each tombstone has an expiry of ten days. The purpose of this delay is to give a node that is unavailable time to recover. If a node is down longer than this value, then it should be treated as failed

and replaced. **Tombstones are removed as part of compaction**. During compaction, any row with an expired tombstone will not be propagated further.

Common problems associated with Tombstones#

Tombstones make Cassandra writes actions efficient because the data is not removed right away when deleted. Instead, it is removed later during compaction. Having said that, Tombstones cause the following problems:

- As a tombstone itself is a record, **it takes storage space**. Hence, it should be kept in mind that upon deletion, the application will end up increasing the data size instead of shrinking it. Furthermore, if there are a lot of tombstones, the available storage for the application could be substantially reduced.
- When a table accumulates many tombstones, read queries on that table could become slow and can cause serious performance problems like timeouts. This is because we have to read much more data until the actual compaction happens and removes the tombstones.

← Back

Compaction

Next →

Summary: Cassandra

Mark as Completed

Report an Issue