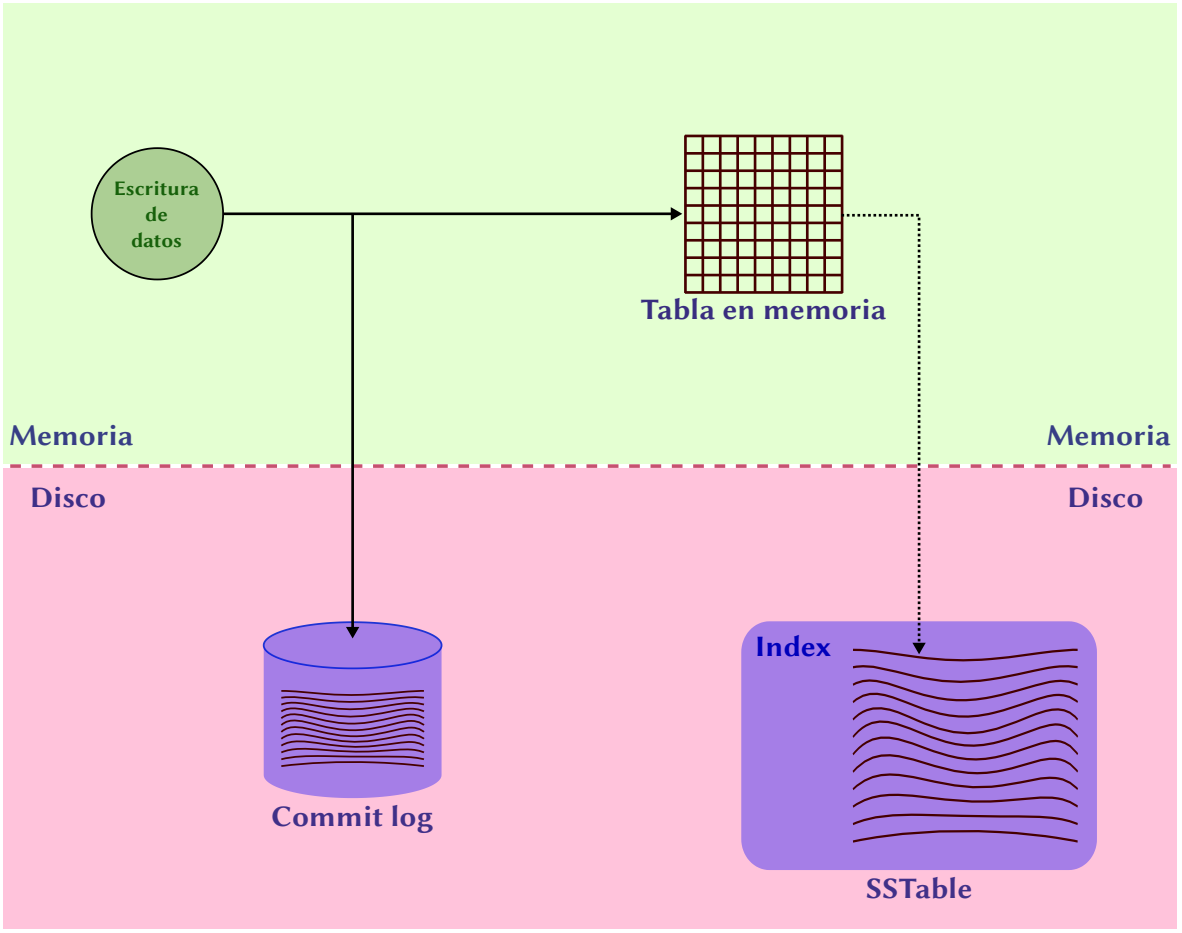
Escritura y lectura en Cassandra

Proceso de escritura en Cassandra



El dato que entra en Cassandra se divide para ir a dos destinos diferentes:

- ***memtable***: es una **tabla en memoria** que se utiliza para almacenar los datos que se van a escribir en disco. Cuando la *memtable* se llena, se vuelca en disco en forma de *SSTable*.
- ***Commit log***: es un **registro de todas las operaciones** de escritura que se han realizado en Cassandra. Se utiliza para **recuperar los datos en caso de** que se produzca un **fallo** en el sistema.

Podríamos decir que la *SSTable* es la base de datos en sí, mientras que la *memtable* y el *commit log* son mecanismos de Cassandra para mejorar el rendimiento y la disponibilidad de los datos.

(La **SS** en *SSTable* significa *Sorted String* y se refiere a que los datos se almacenan en disco de forma ordenada. Las *SSTable* se almacenan en disco en forma de archivos).

Según se van volcando los datos de la *memtable* a la *SSTable* se irán borrando entradas en el *commit log*.

memtable

Consiste en una serie de particiones en memoria. Su utilidad es la de ofrecer gran velocidad de escritura y lectura. Funciona como una caché.

Cuando alcanza un tamaño determinado (indicado en la configuración), se vuelca en disco en forma de *SSTable*.

commit log

Los *commitlogs* son logs de sólo escritura (*append only*) de todos los cambios locales a un nodo de Cassandra. Cualquier dato escrito en Cassandra se escribirá primero en un *commit log* antes de escribirse en una *memtable*. Esto proporciona seguridad frente a un apagado inesperado. Al iniciarse un nodo, cualquier cambio registrado en el *commit log* se aplicará a las *memtables*.

SSTable

Consiste en una serie de ficheros en disco que contienen los datos de las particiones. Estos ficheros serán **inmutables**. Los datos, una vez se escriben en la *SSTable* **no se podrán modificar**. Las operaciones de modificación se realizarán creando nuevos ficheros, con las modificaciones, con un nuevo *timestamp*.

Cada *SSTable* tendrá asociadas las siguientes estructuras de datos (se crean cuando se crea una *SSTable*):

- Data (`Data.db`): contiene los datos de la *SSTable*.
- Primary index (`Index.db`): contiene los índices de las claves de las columnas con punteros a sus posiciones en el fichero de datos.
- Bloom filter (`Filter.db`): estructura almacenada en memoria que sirve para comprobar si una clave existe en la *memtable* antes de acceder a la *SSTable*.
- Compression information (`CompressionInfo.db`): contiene información sobre la compresión de los datos.
- Statistics (`Statistics.db`): contiene información estadística sobre los datos de la *SSTable*.
- Secondary index (`SI_.*.db`): contiene los índices secundarios. Pueden existir varios en cada *SSTable*.
- ...