Versioning Data – First Proposed Design

Single System Design

Versioning

In this proposed system, there will be no compaction (LSM-Tree compaction)

Compaction: throwing away older duplicate keys and deletion marker keys in the log segment, and keeping only the most recent update for each key. Merging them in one segment and discarding older ones in the process.

- We are not throwing away duplicate keys in the log, we are versioning them.
- New keys with latest timestamps will be added to the growing collection of that particular key.
- Keys that have a deletion marker will not be discarded as well, it will just spell the end of that object's growth, but it will be kept in the database.
- An update/edit will add the object to the collection, thus marking it with a version number and its predecessor/parent.

Proposed Data Object

key : { $version_1$: [[value], timestamp, predecessor],...,$version_n$: [[value], timestamp, predecessor] }

Figure 1: Data Object Design

key : { ver_1 : [[val], ts, pre.],...,ver_n : [[val], ts, pre.] }

Figure 2: Shorthand of Data Object Design

- Each key will have a collection of versions.
- Each version within the collection will have a value, timestamp, and predecessor/parent.
  - The value might be another collection containing multiple values for that object's version, as seen below:

    val = first_name, last_name, department, home_address

  - The timestamp is simply when that version was written/persisted to the storage
  - The predecessor is the parent version that derived that specific version, since new versions can be branched from all versions and not only the latest version.

Examples

New object

- Assign latest/unused key
- Assign version number 1 to key; since it's the first entry for the key
- Accept and assign input to version 1: [value, timestamp, 0 (since it's the first version, thus no predecessor/parent)]

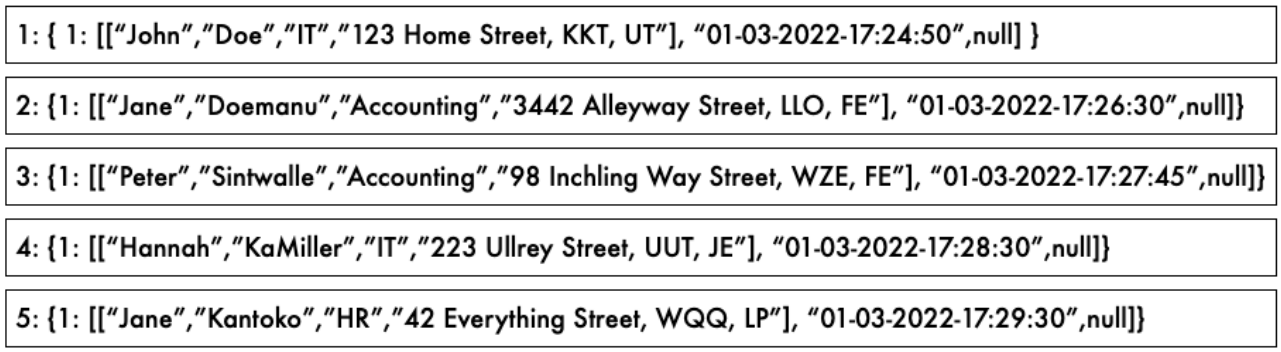

Figure 3: New objects

Update object

- Search for key first and display all versions of object once found
- User should indicate which version to view/focus on
- Then the user can choose to edit that value, thus creating a new branch from that specific version

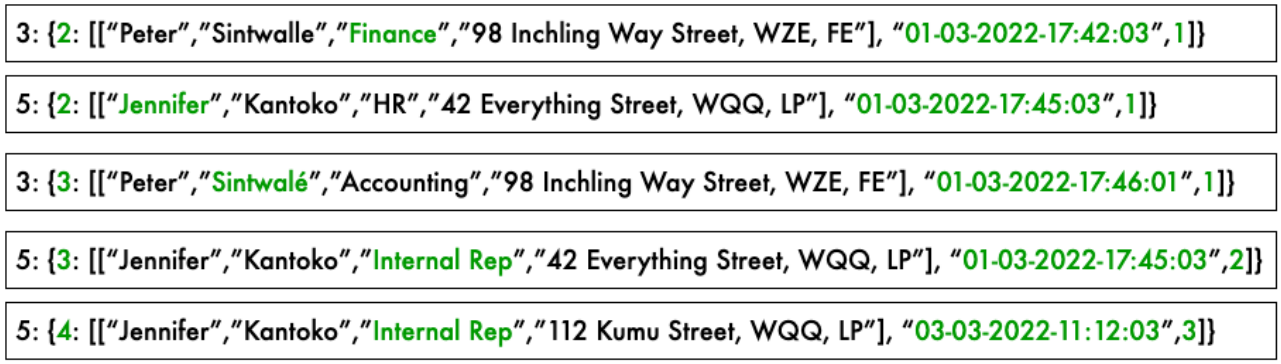

Figure 4: Update existing objects – green highlights indicate changes to objects

Delete/Discontinue/Terminate objects

- Search for key first and display all versions of object once found
- Then the user can choose to terminate the entire object, thus halting the growth of that object
- No new branches/versions can be derived from this object
- The user can still view the object and all its versions/branches, but cannot update

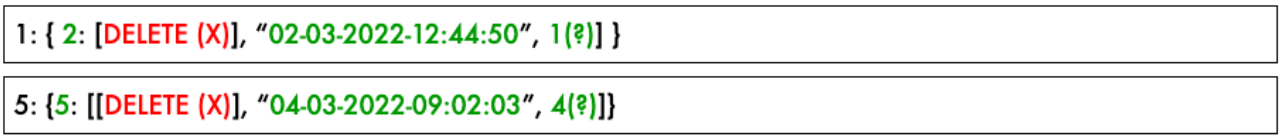

Figure 5: Discontinue objects – green highlights indicate changes to objects, red indicate deletion marker

Reconciliation/Merging Key Versions

- When merging segments, if multiple similar/duplicate keys are identified, they will not be discarded from the log, they will be added to an existing collection, or a new collection will be made.
- With the key as an identifier and the duplicate data objects will be distinguished with versions within the collection.
- Once merge segment has been made, the older segments will be discarded to reconcile storage space/capacity.
- And repeat.