

**Memo** (immutable msg)  
Differences to ancestor memos

Projection based on known memos

Action →

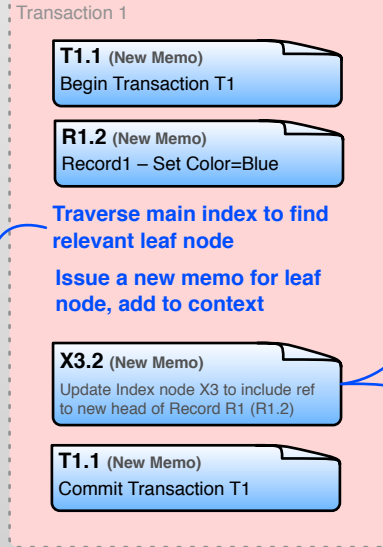
**Slab 1** (A slab is similar to what most think of as a "node", except that you may have multiple slabs corresponding to different processor cores, or different services, for instance)

**X1.1 (Existing Memo)**  
Create Index node X1 (various links)

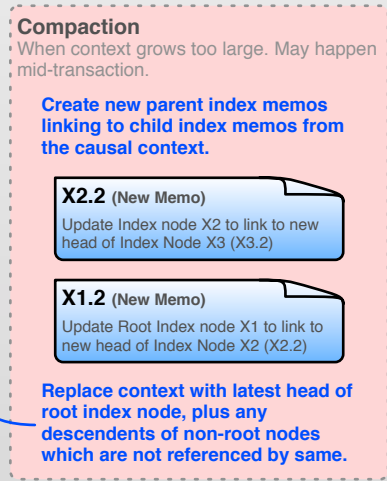
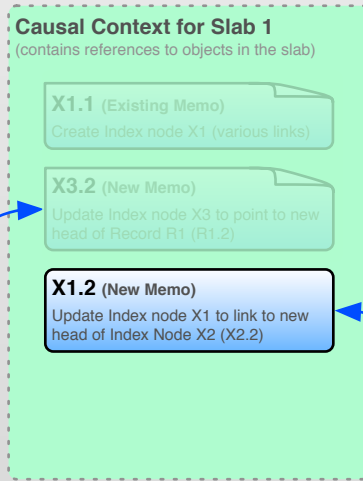
All Slabs must initialize with at least one memo from the root index node. It needn't be fresh

**R1.1 (Existing Memo)**  
Create Subject R1 – Color=Red

Last known "head" of Record R1 Gotten from previous transaction or pushed by a peer slab for durability assurance



Traverse main index to find relevant leaf node  
Issue a new memo for leaf node, add to context



TODO:

- review how index node memos are to be selectively acknowledged in order to keep traffic manageable. (Maybe merge lazily only on the basis of context swaps?)
- show how keyframes are generated
- show how projection works

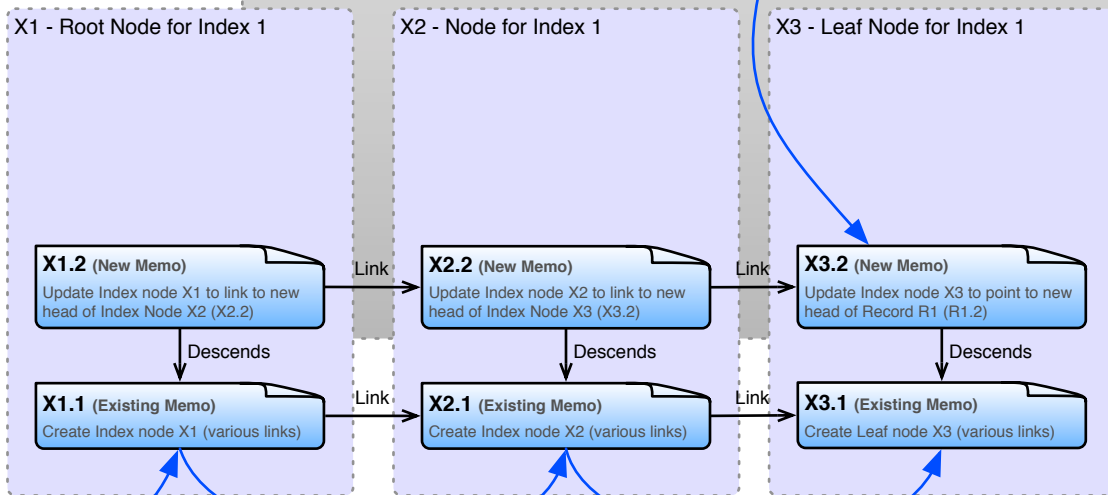
Q: how do I serialize a causal context for api users?  
A: broadcast memo hunt OR memo index

Q: how do I determine when to share causal context? ( or otherwise how do I control merging such that its layered

NOTES:

- \* Memos reference beacons
- \* index keyframes associate beacon vector to each slot
- \* All keyframes associate beacon vector to the overall frame (consider compression schemes inside each keyframe?)
- \* Beacons never get handed off, they're born, and they die
- \* Beacons are selected by their longevity, consistency, and conformance to the target interval
- \* All memo propagation can be vicarious via plumtree
- \* keyframes are merged commutatively by comparing beacon vectors for each slot
- \* each slab may be a beacon
- \* beacons which are unpopular will tend to self-terminate
- \* beacons which are popular will tend to persist
- \* each slab's probability of activating its beacon function is inversely proportionate to the health of its beacon set
- \* does plum tree allow for popularity estimation?

Upon traversal/retrieval, copies of the index memos, and any other retrieved memos are stored at least momentarily.



Assumes fixed 3-tier unbalanced index tree with 1024 per slots each node (1024^3 = 1 Billion records)  
Would likely be a 4-tier system. Balanced-tree indexing work may be possible, but is not presently included in this design