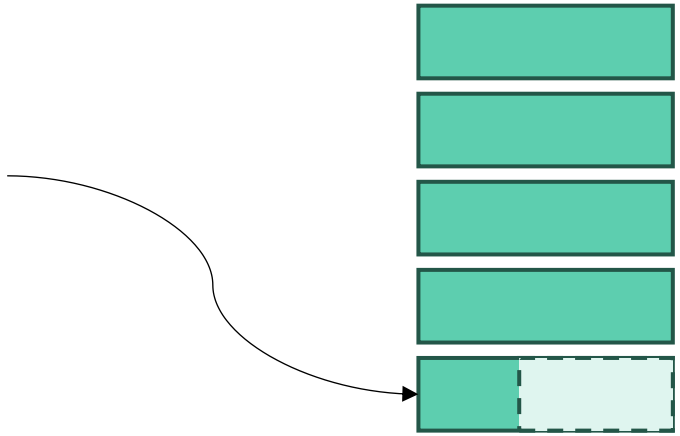
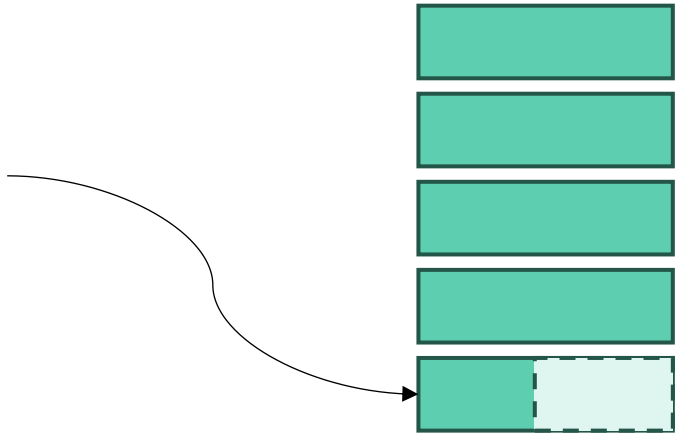
6. SEGMENTING

- Split collection to **segments**
  - Example 1M vector/seg
- Insert: append to growing segment



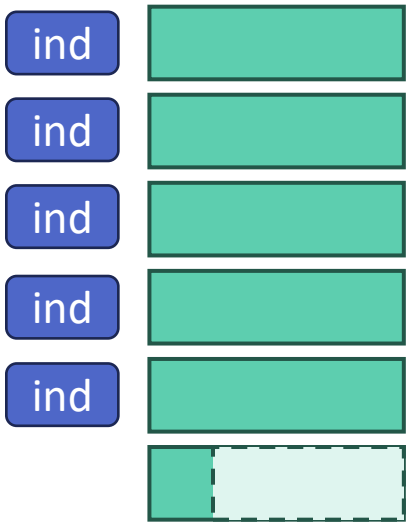
6. SEGMENTING

- Split collection to **segments**
  - Example 1M vector/seg
- Insert: append to growing segment



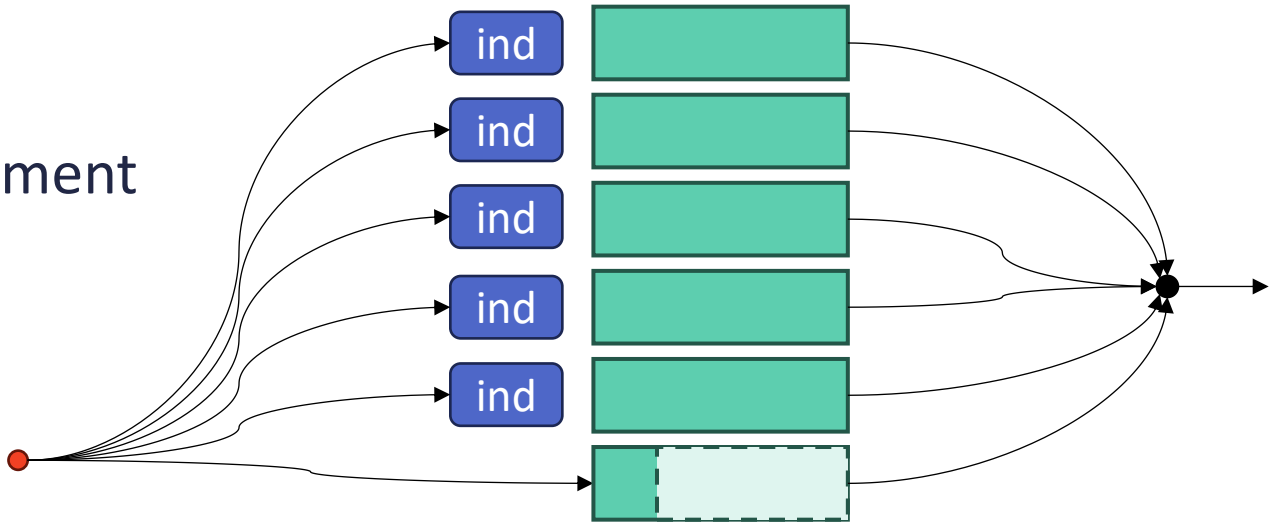
6. SEGMENTING

- Split collection to **segments**
  - Example 1M vector/seg
- Insert: append to growing segment
- Index segment when full
  - Open new growing segment



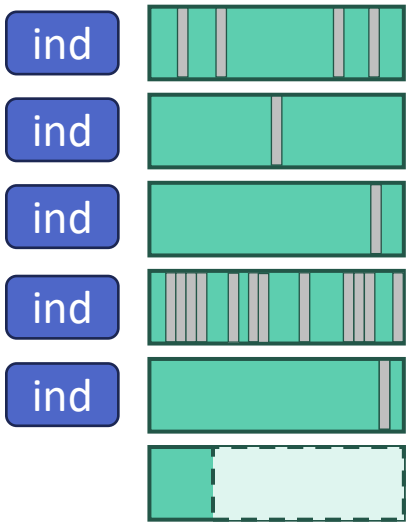
6. SEGMENTING

- Split collection to **segments**
  - Example 1M vector/seg
- Insert: append to growing segment
- Index segment when full
  - Open new growing segment
- Query all segments, combine



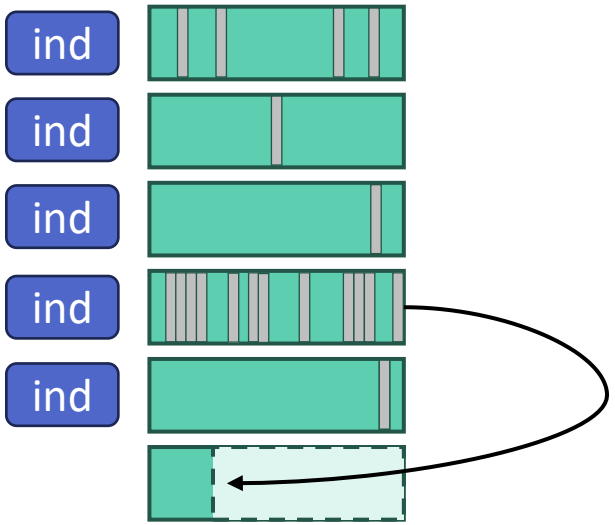
6. SEGMENTING

- Split collection to **segments**
  - Example 1M vector/seg
- Insert: append to growing segment
- Index segment when full
  - Open new growing segment
- Query all segments, combine
- Mark deleted vectors (tombstones)



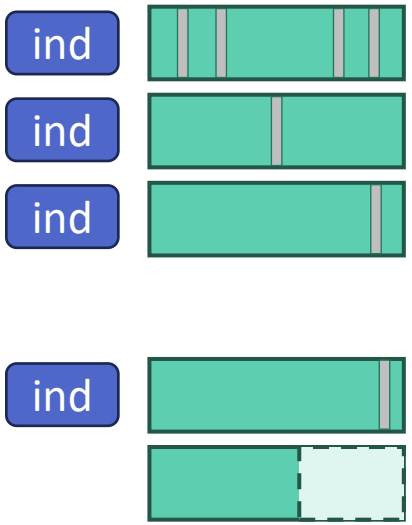
6. SEGMENTING

- Split collection to **segments**
  - Example 1M vector/seg
- Insert: append to growing segment
- Index segment when full
  - Open new growing segment
- Query all segments, combine
- Mark deleted vectors (tombstones)
  - Merge mostly-empty segments



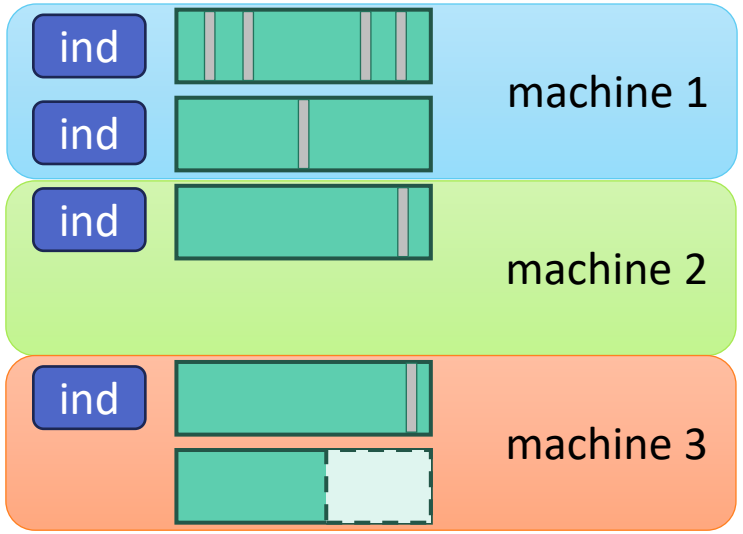
6. SEGMENTING

- Split collection to **segments**
  - Example 1M vector/seg
- Insert: append to growing segment
- Index segment when full
  - Open new growing segment
- Query all segments, combine
- Mark deleted vectors (tombstones)
  - Merge mostly-empty segments



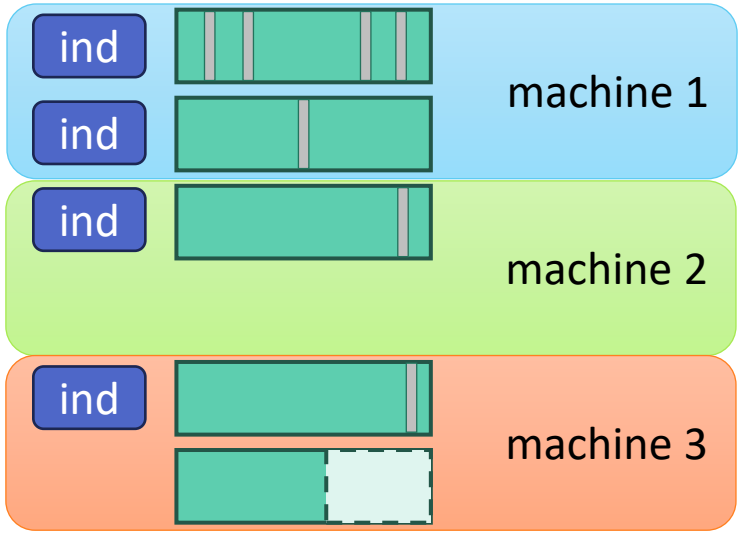
6. SEGMENTING

- Split collection to **segments**
  - Example 1M vector/seg
- Insert: append to growing segment
- Index segment when full
  - Open new growing segment
- Query all segments, combine
- Mark deleted vectors (tombstones)
  - Merge mostly-empty segments
- Distribute segments to parallelize index, querying



6. SEGMENTING BENEFITS

- ✓ **No more rebuilds**
  - Segments are static
  - Build on full segment, on merge
- ✓ Each index is small
- ✓ Growing segment = freshness layer
- ✓ Easy to distribute work
  - Example: allocate segments to shards
- Downsides:
  - Must query **all** segments
  - Write amplification if update-heavy



Used in many VecDBs!
(e.g., Milvus, Qdrant)

6. SEGMENTING THOUGHTS

- Segmenting ≠ sharding
  - Sharding: distribute data across machines
  - Segmenting: avoid reindexing, accommodate growth
- Work well together
  - Shard by key and segment each shard
  - Qdrant, Milvus
- Other perspectives:
  - Sharding – insert/write performance
    Segmenting – query performance
  - When adding data →
    num shards fixed, shards grow
    num segments grows, segments do not

good even if not indexing

good even on single machine