Indexes don't mean slow inserts

Anastasia Lubennikova

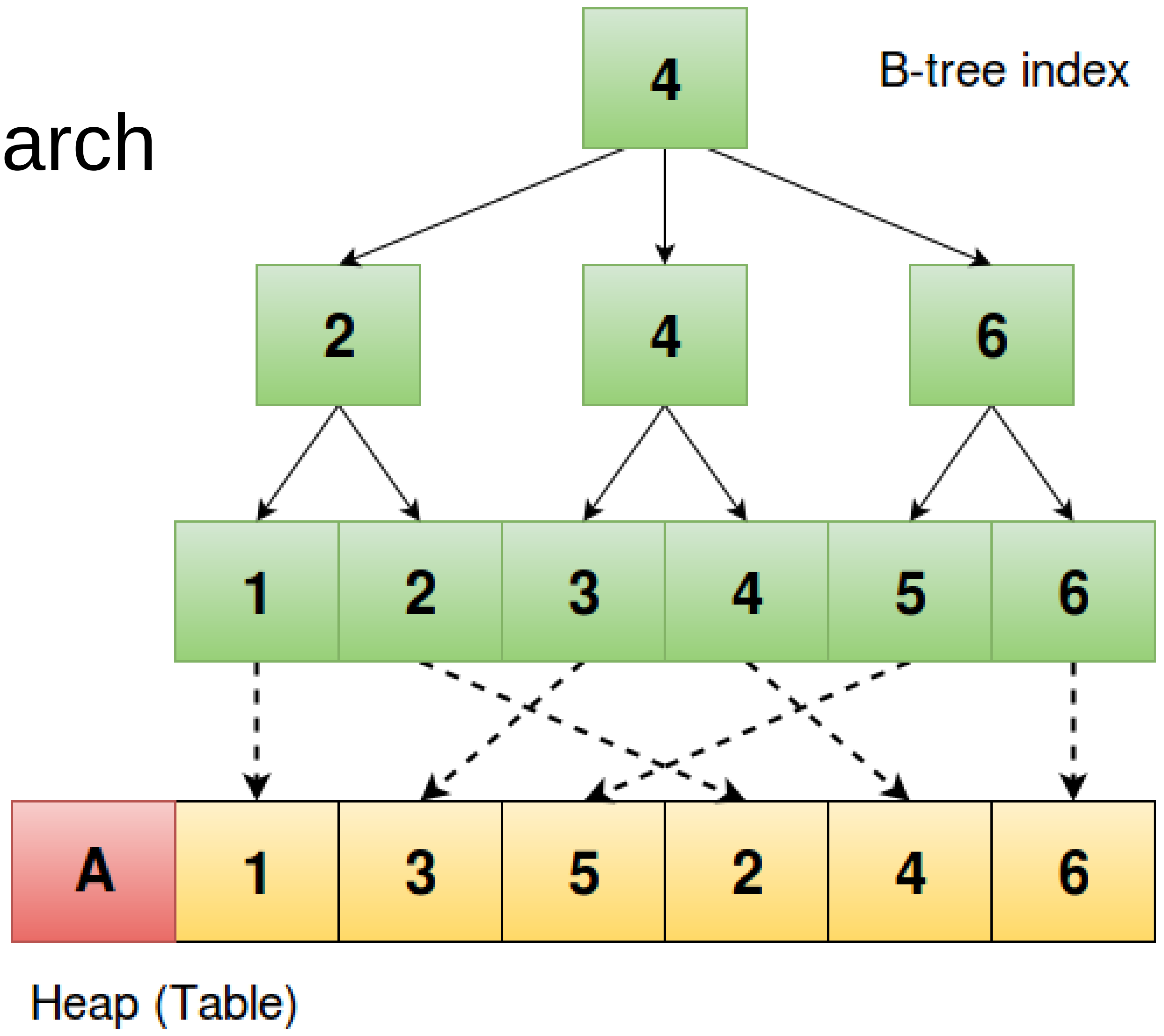
Agenda

1. Why do we need it?
2. Write-optimisation techniques
3. PostgreSQL specific
4. Advanced PostgreSQL indexes
5. Future of indexing in PostgreSQL

PostgreSQL indexes

- Speed up search
- Primary key
- Constraints

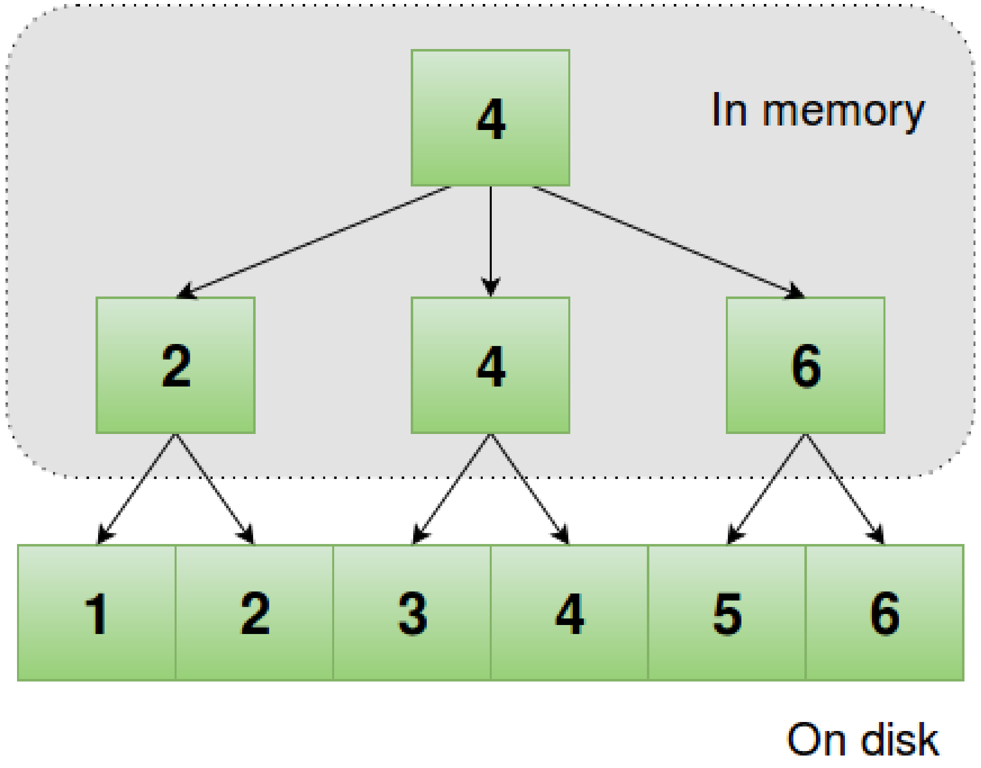
- Secondary indexes



Index maintenance overhead

- Index size
- INSERT slowdown
- Random I/O
- Index becomes fragmented

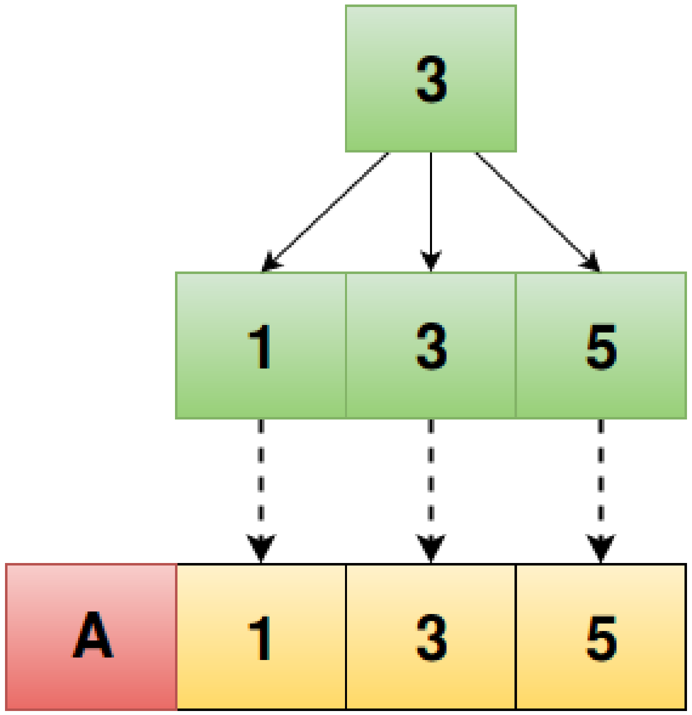
- More indexes - more overhead



Do we need write-optimisation?

- Heavy write load
- MVCC

  update = insert



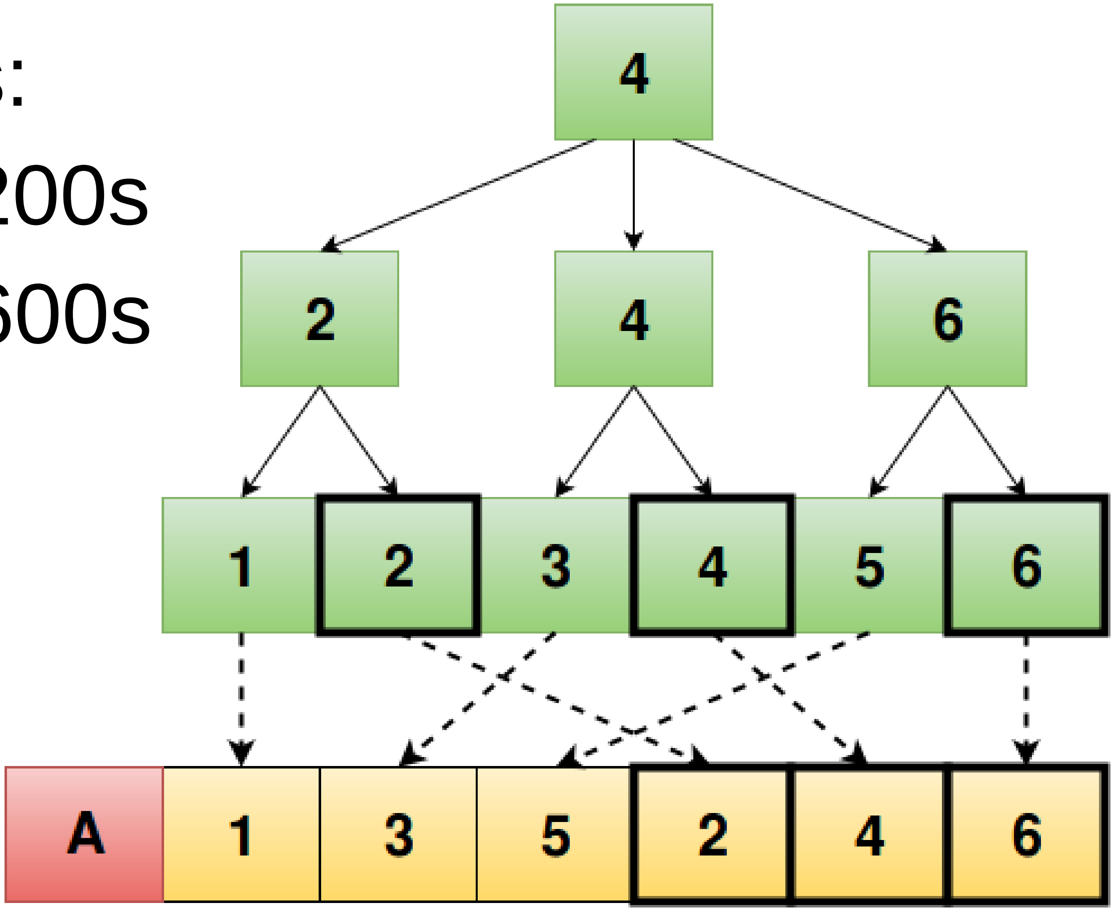
UPDATE mytable SET a = a + 1;

Do we need write-optimisation?

- 1Gb table
- Update all values:

  Without index ~ 200s

  With index ~ 600s



DBMS trade-offs

DBMS trade-offs

- CAP-theorem
- ACID vs BASE
- Lower hardware cost vs Better productivity
- Read speed vs Write speed
- Productivity vs Fault-tolerance

Write-optimisation

- Writes are faster
- Reads are good
- Storage is fault-tolerant

Insert buffer

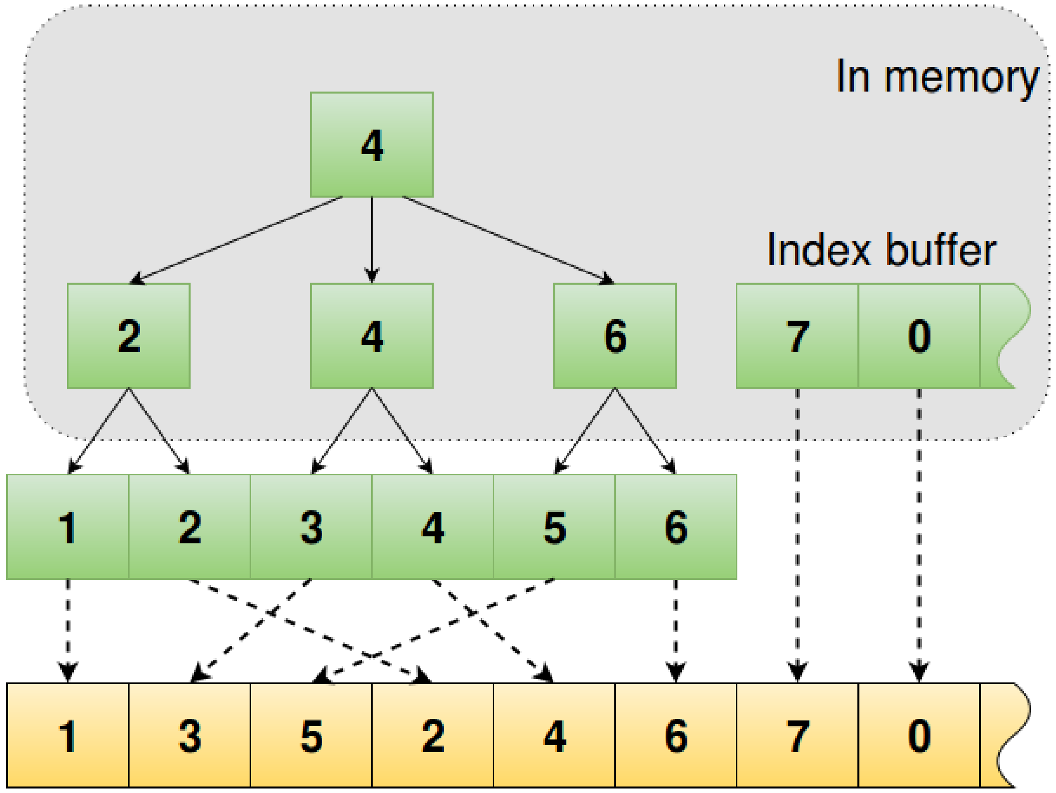
- Accumulate data. Sort. Insert at a time.

  + Avoids random I/O

  - Seqscan buffer

  - Possible data loss

  - Merge time



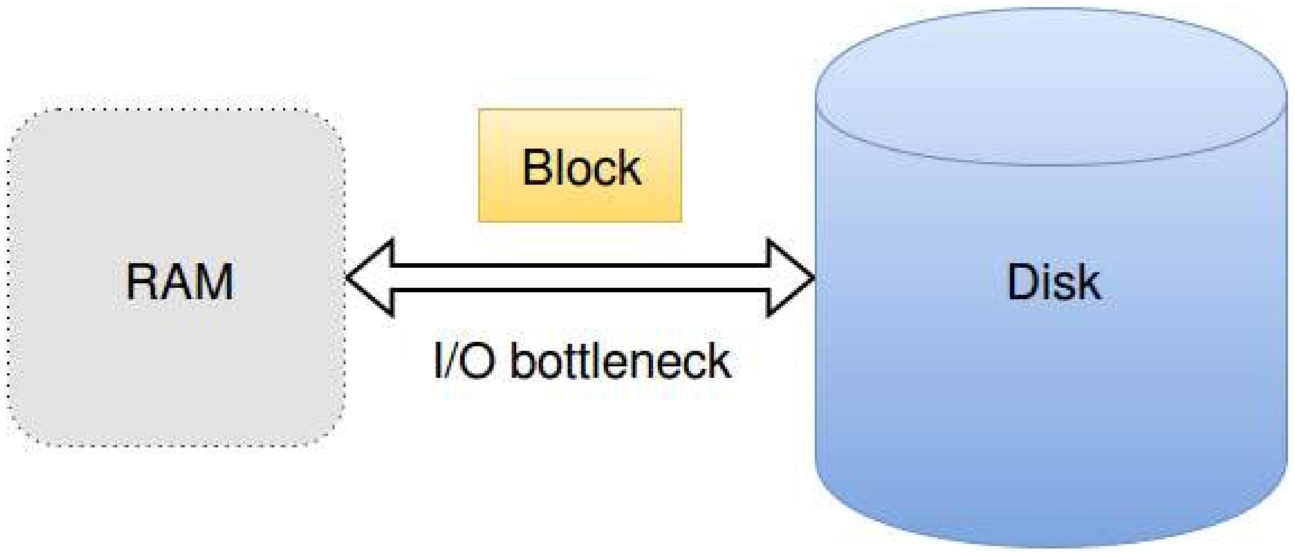
- Avoids hidden scans

  - only non-unique

- MySQL InnoDB Change Buffer

Cache-oblivious data structures

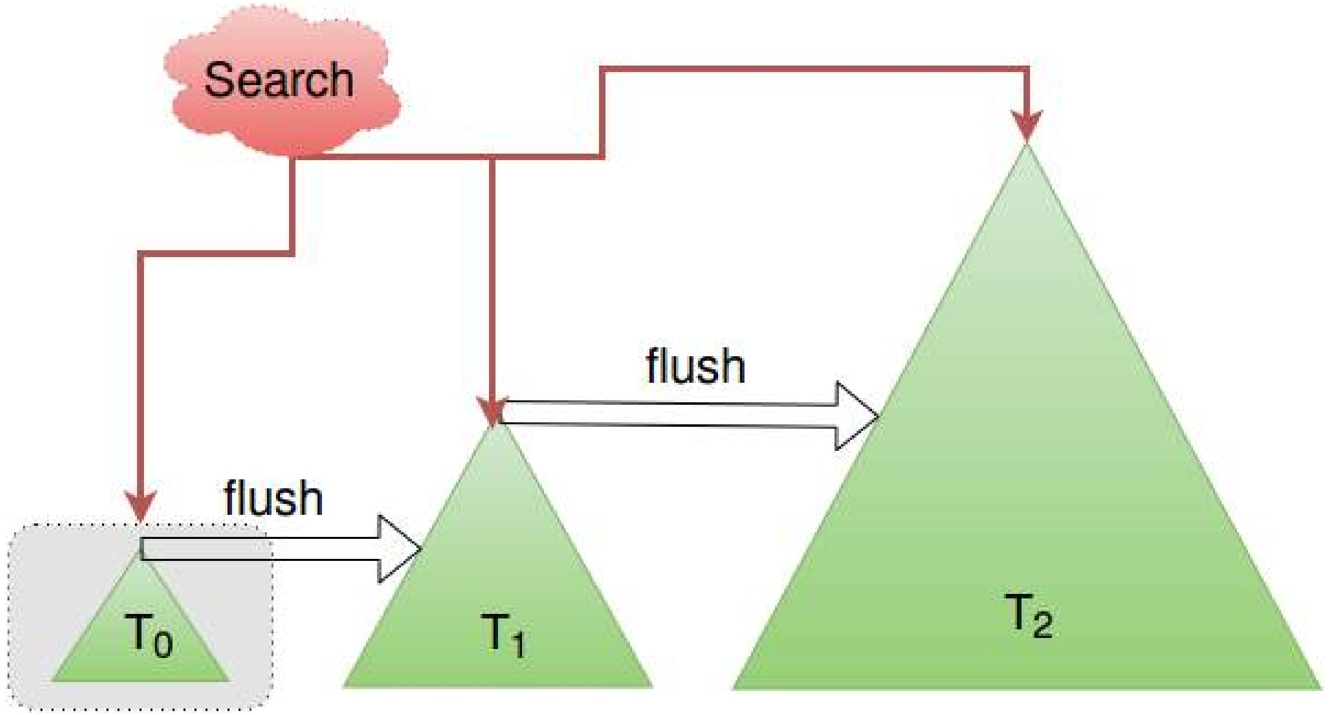
- Approximately optimal for any hardware
- Divide & Conquer



LSM trees

- Cascade of B-trees
- First tree is in memory

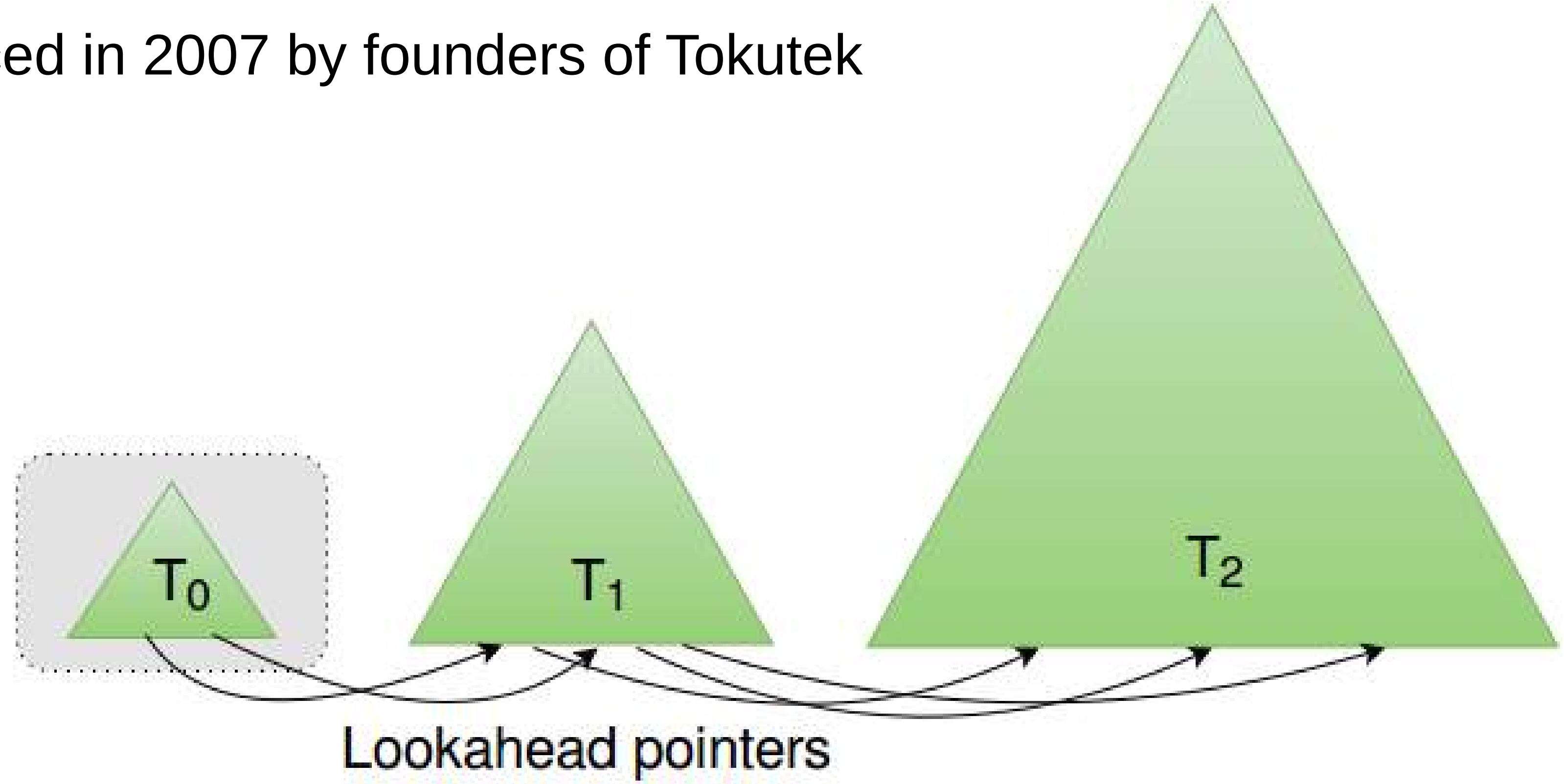
- LevelDB
- BigTable
- Cassandra
- Hbase
- SophiaDB
- other NoSQL DBs



From LSM to COLA

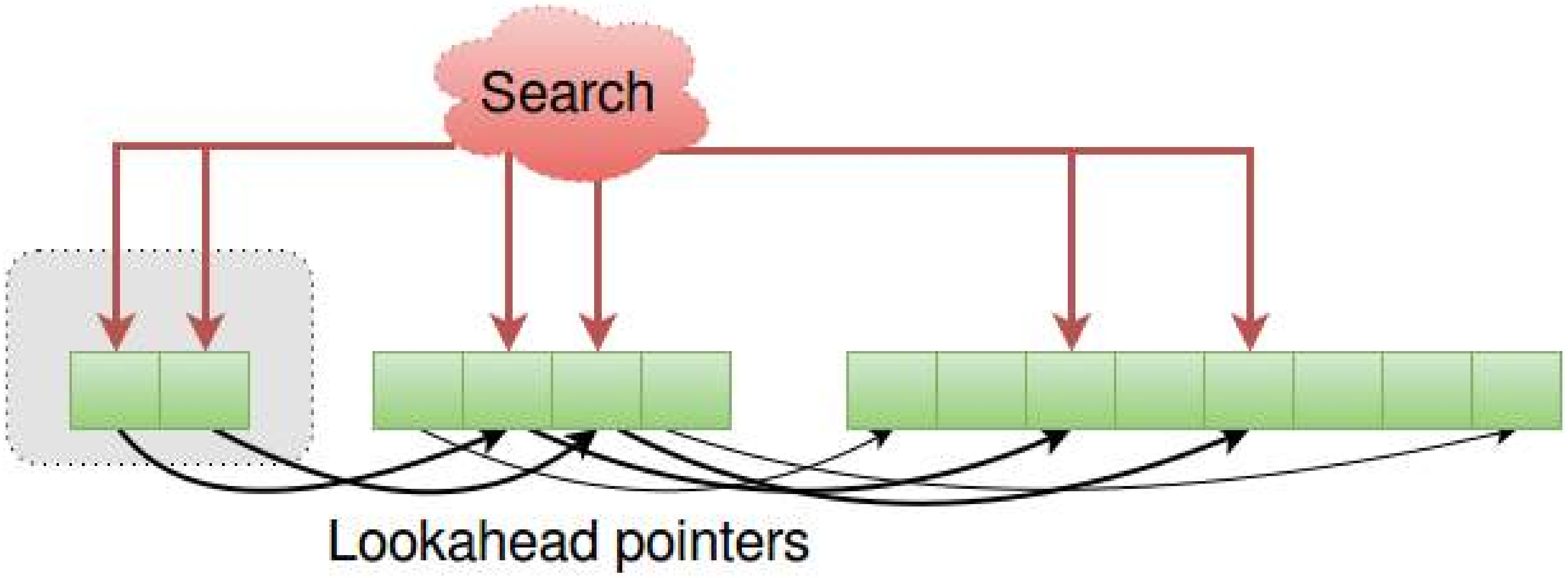
- Search optimisation for LSM
- Leaf levels are linked by lookahead pointers

- Introduced in 2007 by founders of Tokutek



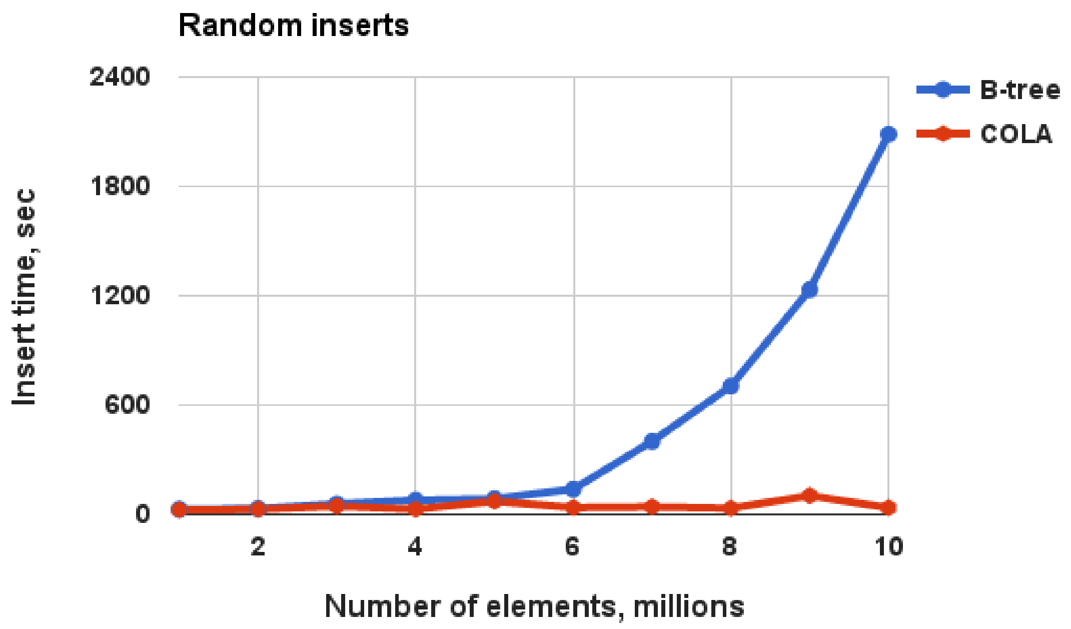
Cache-oblivious lookahead arrays

- Drop internal trees nodes
- Bound the scan area with lookahead pointers



COLA: theory and pracrice

- Prototype shows incredible results!

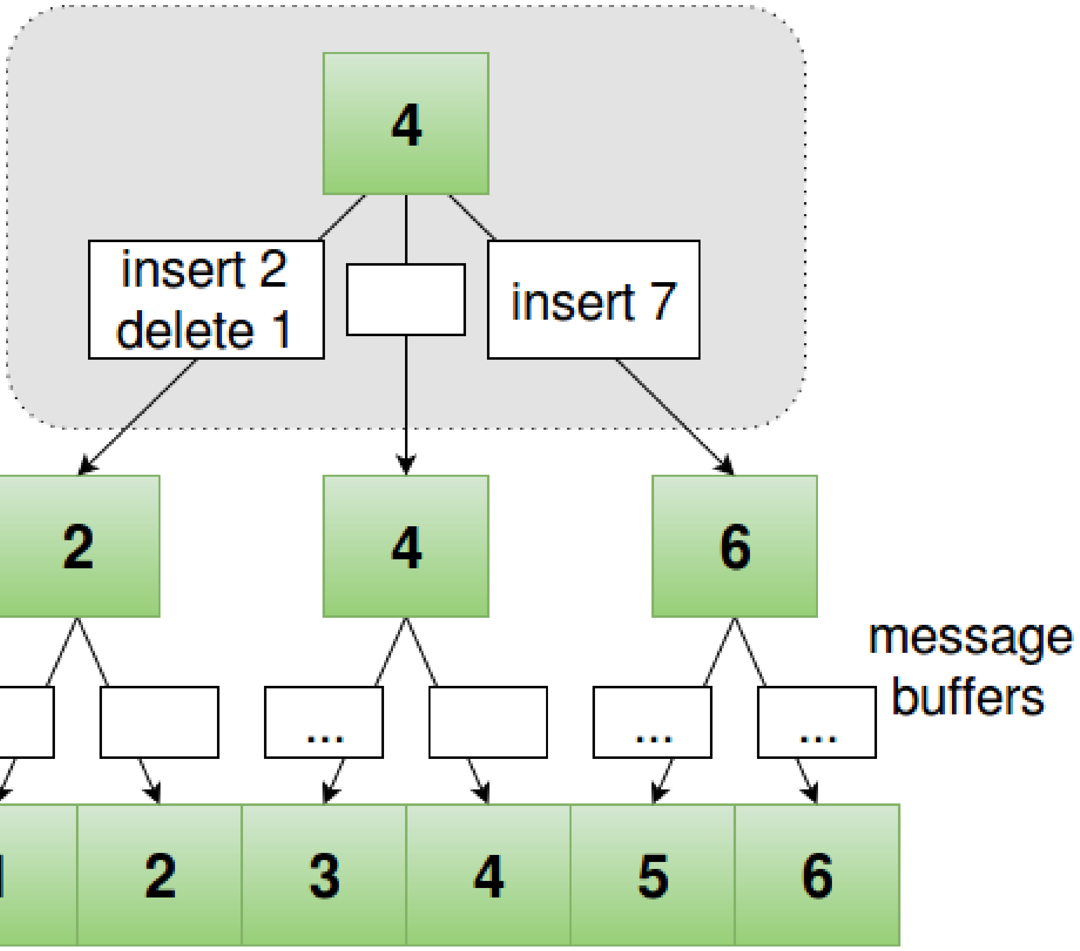


- VACUUM?
- WAL?
- Concurrency?
- Index size?

- too hard =(

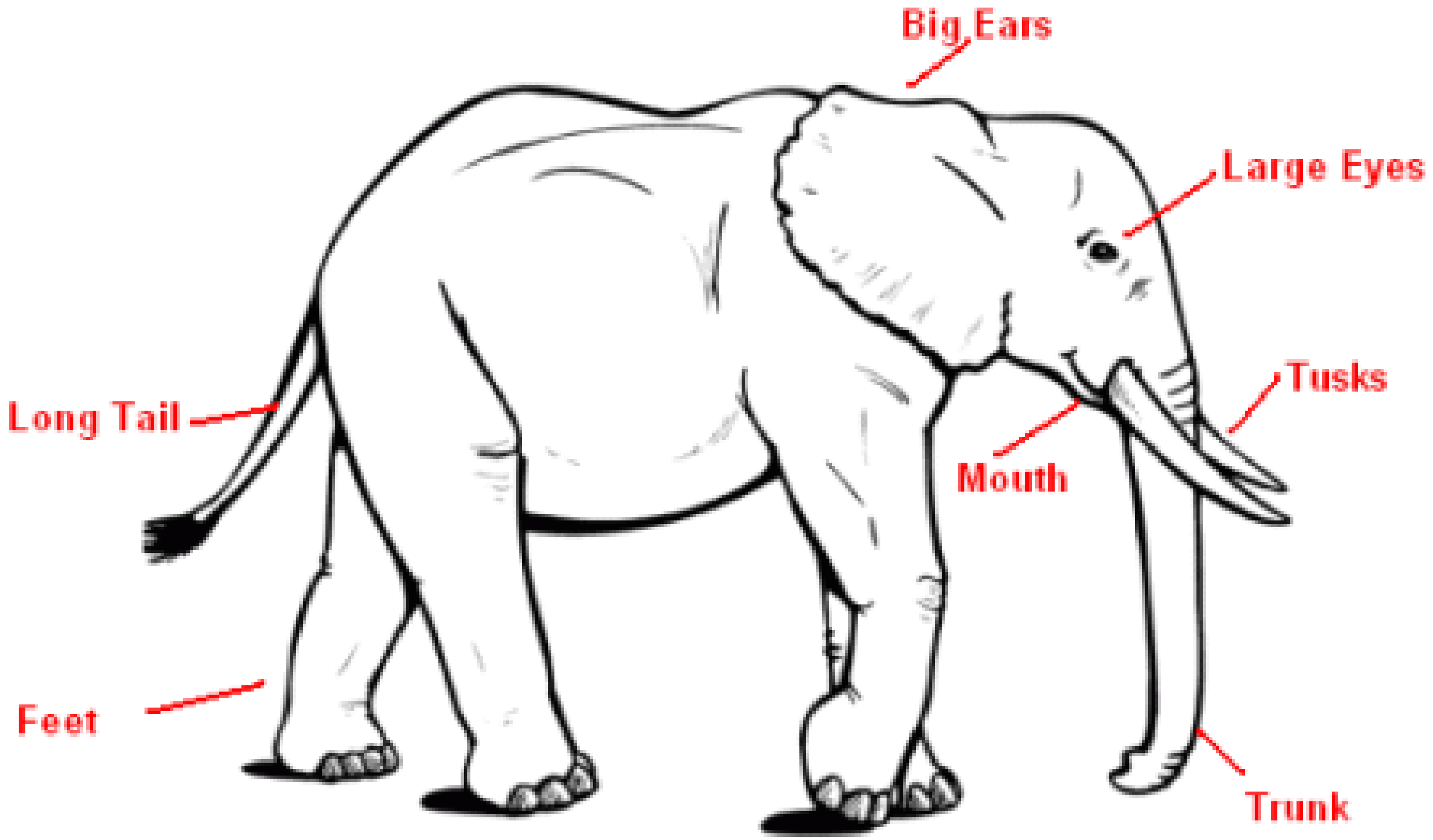
Fractal Tree

- Insert the message instead of the data
- Send it down the tree
- Apply a message
  to leaf page



- TokuDB for MySQL
- TokuMX for MongoDB

PostgreSQL specific



PostgreSQL specific (1)

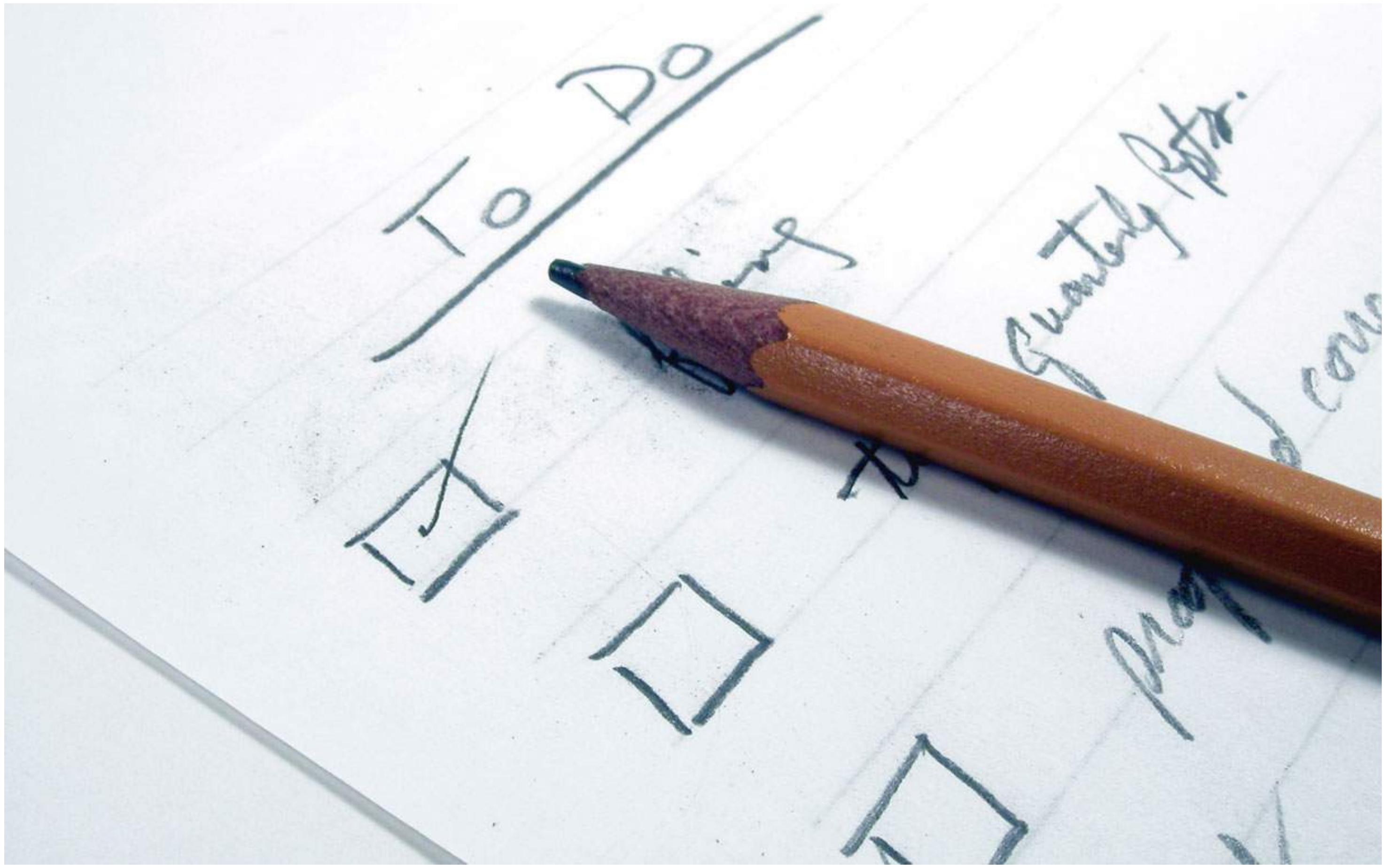
- Write-Ahead-Log
  - WAL is not extendable
- File manager
  - 1 Relation (Heap or Index) = 1 continuous file
  - Free Space Map
- Block size
  - 8 Kb

Advanced indexes

Advanced PostgreSQL indexes

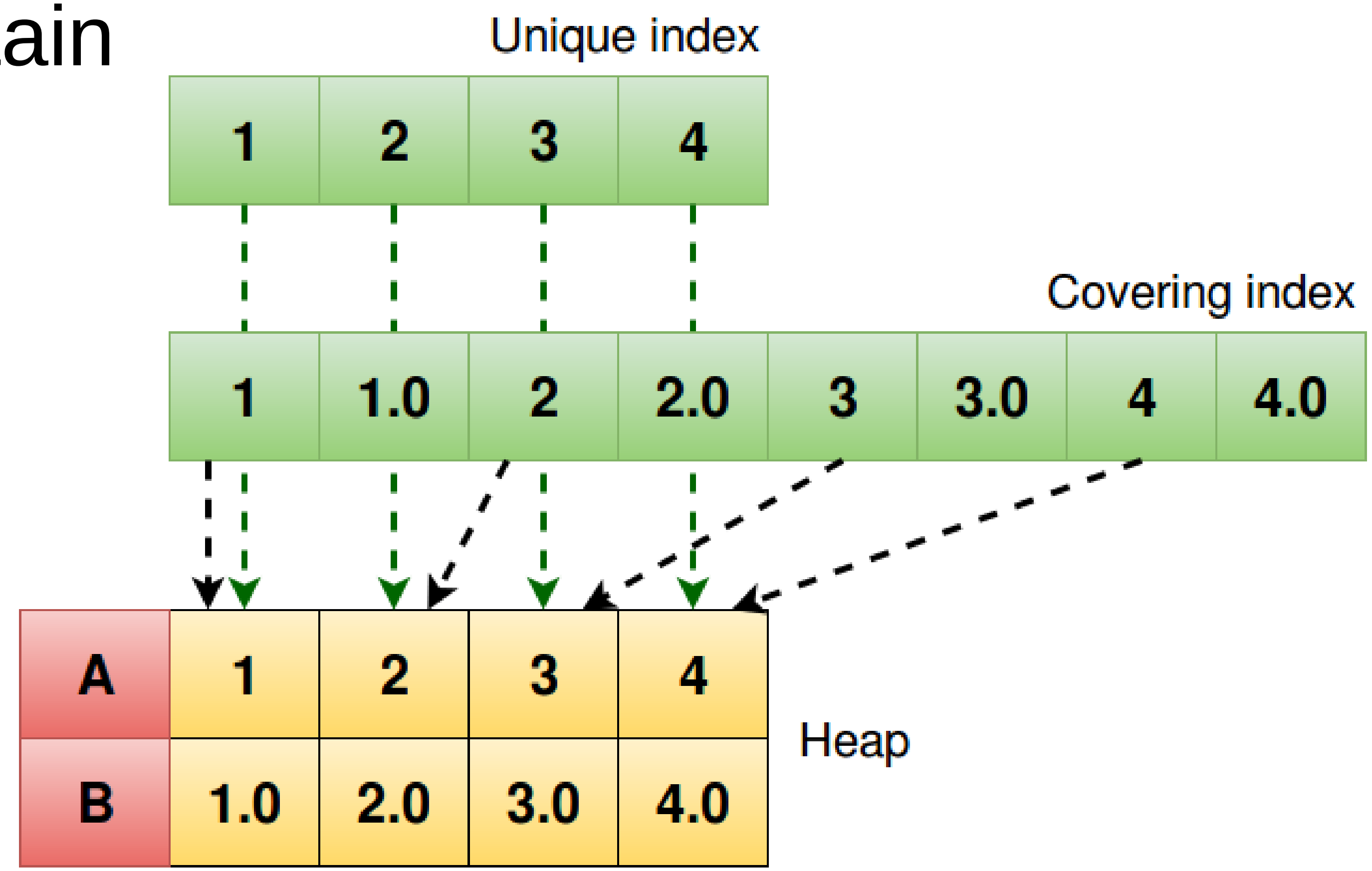
- Optimize the number of indexes
  - pg_stat_statements
- REINDEX
- CREATE INDEX CONCURRENTLY
  - rebuild bloated and fragmented indexes
- Partial indexes
- BRIN
  - tiny min/max index

Ideas?



Covering and Unique

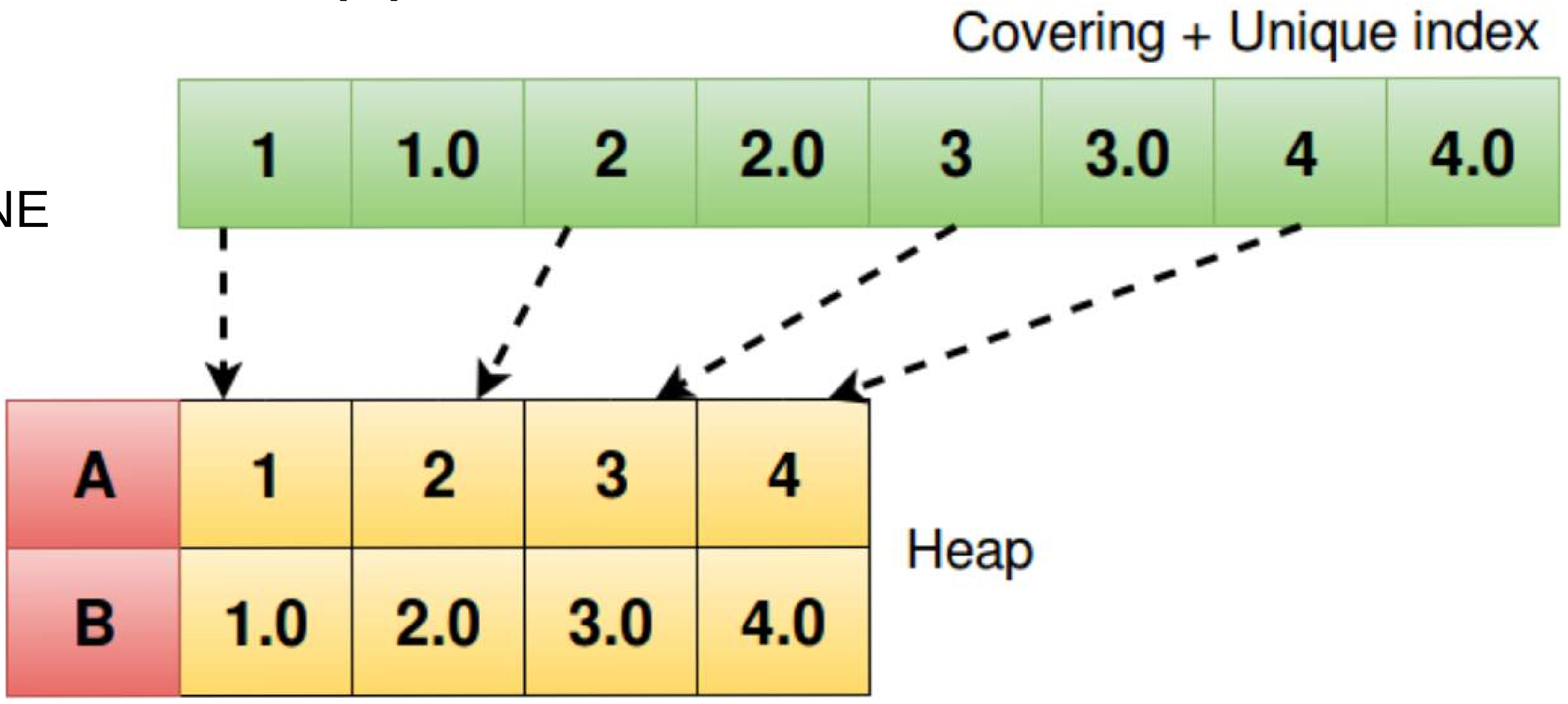
- To maintain constraint (Unique, Pimary key..) on A
- To use IndexOnlyScan on A,B
- Have to maintain 2 indexes



Covering + Unique

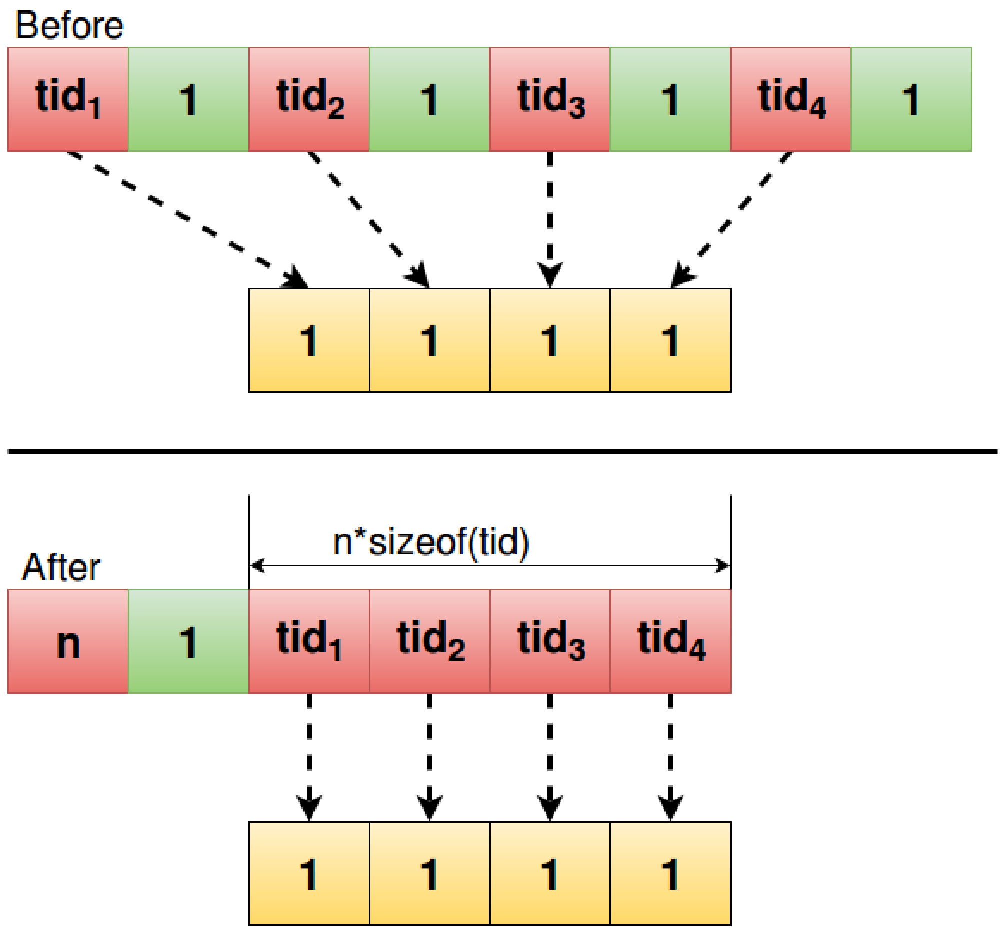
CREATE UNIQUE INDEX ON mytable
USING btree(a)
INCLUDING(b);

- DONE



Effective storage of duplicates

- Compress duplicated keys on index page
- IN PROGRESS



Bulk insert

- INSERT INTO mytable
  SELECT x
  FROM generate_series(0, 1000000) as x;

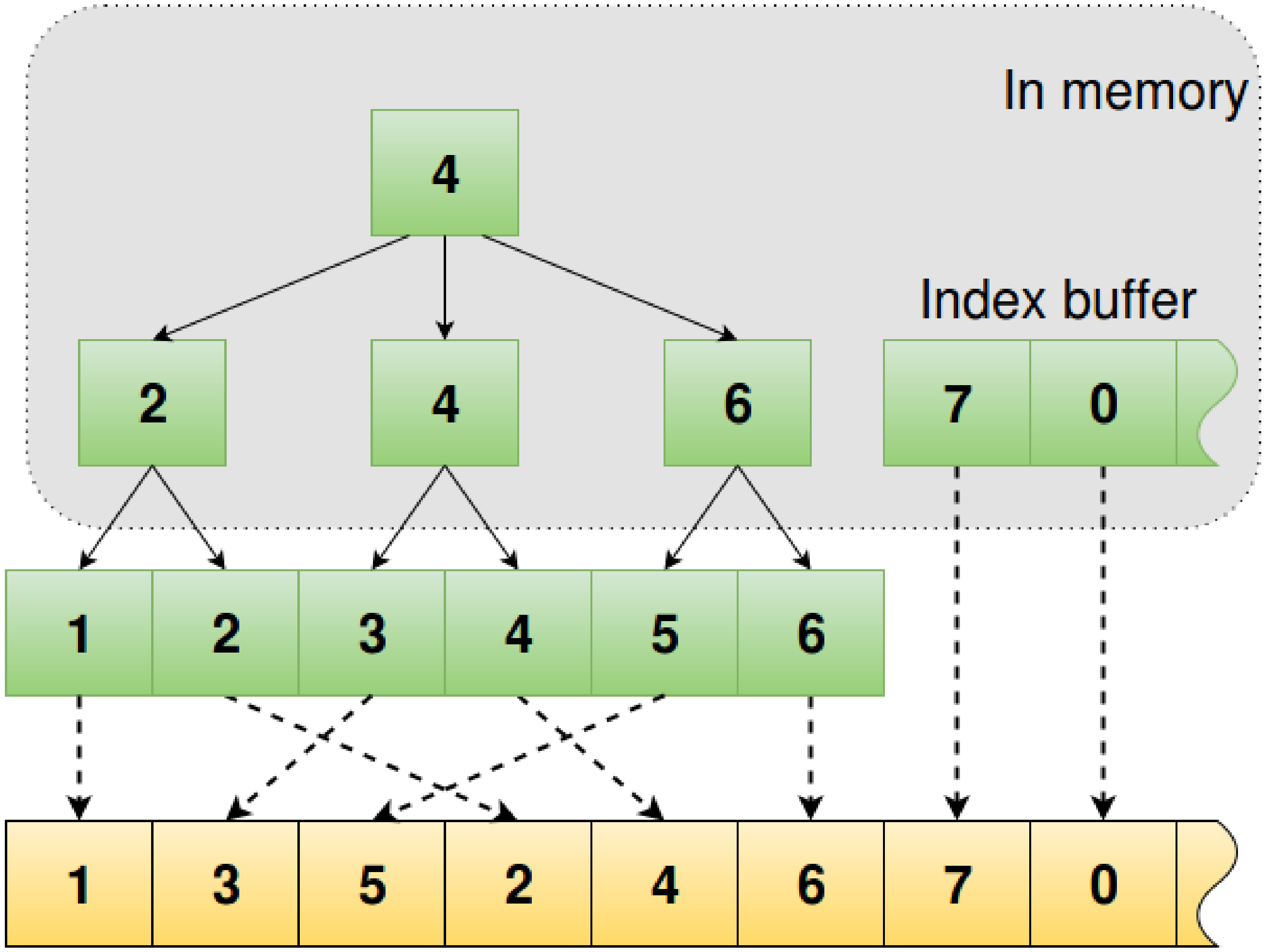
- 1.000.000 B-tree searches
- 1.000.000 WAL records

- TODO

Insert Buffer

- Flexible
- Recoverable

- TODO



Thanks for attention!
Any questions?

a.lubennikova@postgrespro.ru