DuckDB in Science Meetup 2025 (London)

BOSTON UNIVERSITY

Elevating Bitmap Indexing in OLAP DBMSs

CUBIT, RABIT, and …

Junchang (Jason) Wang and Manos Athanassoulis

DiSC lab

Bitmap Index Basics

Column A	A=10	A=20	A=30
30	0	0	1
20	0	1	0
30	0	0	1
10	1	0	0
20	0	1	0
10	1	0	0
30	0	0	1
20	0	1	0

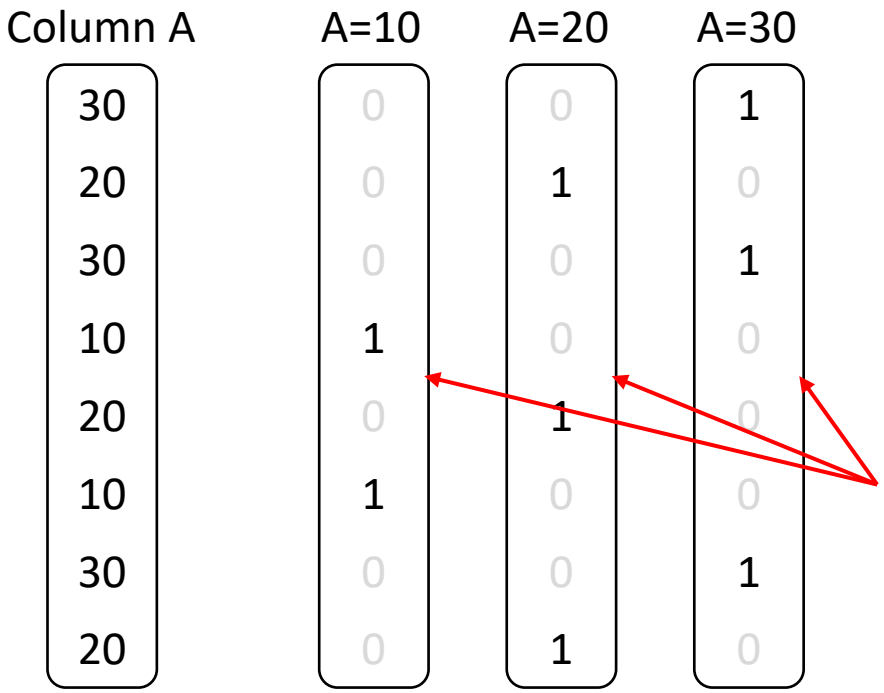
Specialized indexing

- ☑ Query result is readily available
- ☑ Compact representation of selective query result

Bitvectors

- ☑ 0s are highly compressible
- ☑ Efficient bitwise operations (e.g., OR/AND)
- ☑ Sequential access minimizes cache and TLB misses

Bitmap Indexing Limitations



Index Size

✓ Addressed by bitvector encoding/compression

core idea: *run-length encoding (RLE) to compactly store long runs of 1s or 0s*

but …

✗ Updating encoded bitvectors is **very** inefficient

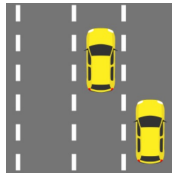
CUBIT's Goal

Update-friendly Bitmap Indexes on Multicore Systems

- **Queries are wait-free** with guaranteed completion
- **Updates are lock-free** and avoid blocking each other

Concurrent Updatable **BIT**map Indexing (CUBIT)

Out-of-place update mitigates synchronization complexity

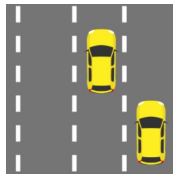


Multi-versioning enables parallel execution of queries and updates

Bitvector Segmentation exploits parallelism on multicore systems

CUBIT

Out-of-place update



Multi-versioning

Bitvector Segmentation

Goals

Mitigate synchronization

Parallel queries and updates

Parallelism on multicores

Experiments

Synthetic data

- ***n***: # tuples
- ***d***: # domain values (cardinality)
- ***q***: # queries
- ***u***: % updates in the workload

Prototype C++ implementation of CUBIT, UpBit, UCB, and In-place using FastBit

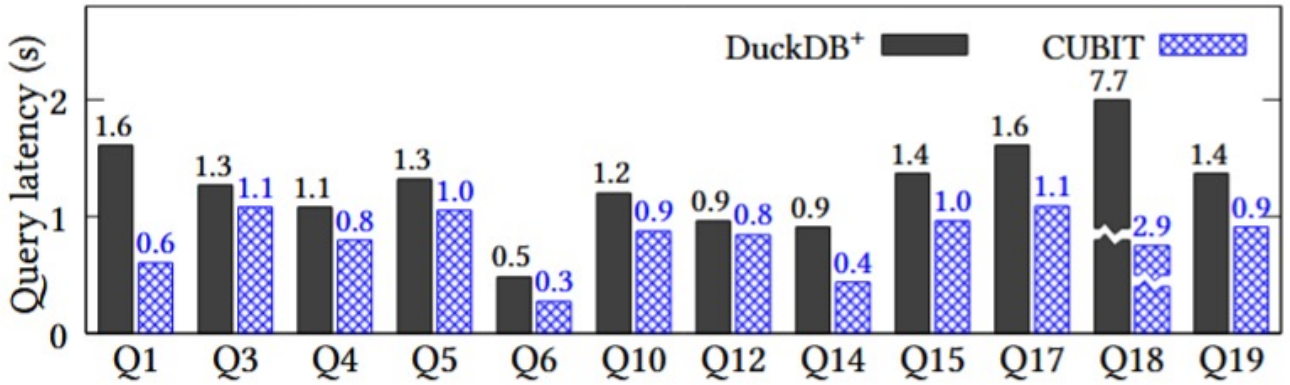
Integrated into a row-store prototype system (DBx1000) and a column-store system (DuckDB)

Integrating CUBIT into DuckDB

- TPC-H SF 10 dataset
- **DuckDB version 1.0**
- **Single-core execution to focus on executor logic**

Maintain CUBIT instances for table attributes under updates

- l_quantity (cardinality = 50)
- l_shipdate (cardinality = 2,526)
- l_orderkey (cardinality = 15M)
- …



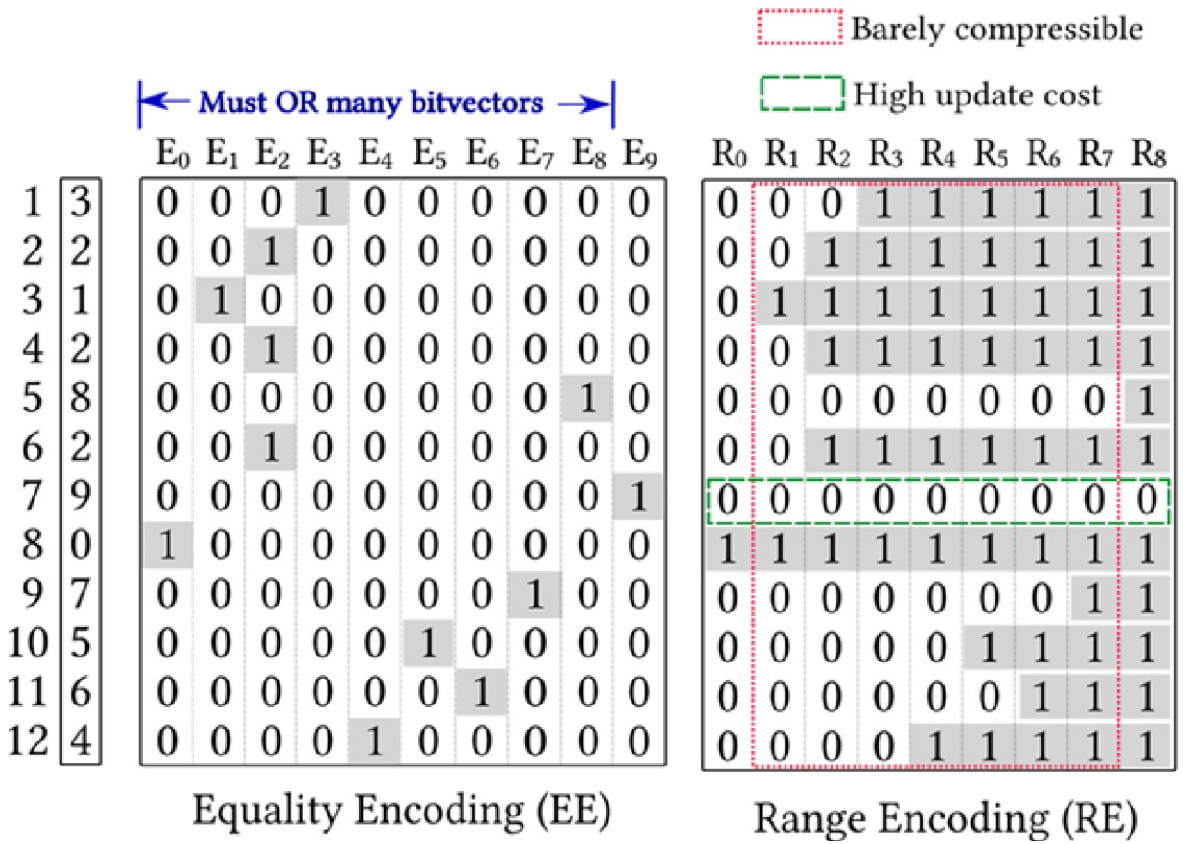
Implement CUBIT-based Scan, Aggregation, and Join executors

- See our paper for details

DuckDB makes it easy to integrate and experiment with CUBIT

CUBIT accelerates not only Scan, but also Aggregation and Join

To be continued …



CUBIT mainly focuses on point queries, meaning

- it mainly adopts equality encoding scheme
- each query reads only one bitvector

However,

to support range queries (e.g., "A <= 8"),

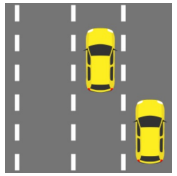
- **Extend updatability to encoding schemes beyond equality encoding**
- **Support high-cardinality attributes**

RABIT: Efficient Range Queries with Bitmap Indexing (SIGMOD'26)

CUBIT is Update-Friendly for Multicore Processors

DuckDB enables rapid integration and test with CUBIT !

Out-of-place update



Multi-versioning

Bitvector Segmentation

Thanks!

https://disc.bu.edu

We are pushing bitmap toward general secondary indexes

Please contact if you are interested

Thank you!