# TIDE: Indexing Time Intervals by Duration and Endpoint

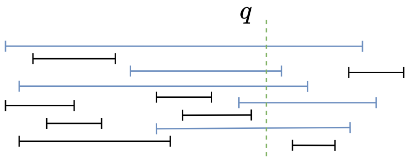
Kai Wang, Moin Hussain Moti, Dimitris Papadias

Hong Kong University of Science and Technology

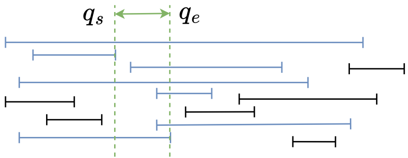
Aug 27, 2025

# Interval Queries

- In temporal databases each record has a lifespan $[t_s, t_e)$, where $t_s < t_e$. A record is *alive* if $t_e$ equals the current time; otherwise, it is *dead*.
- Stabbing query: returns intervals intersecting timestamp $q$.

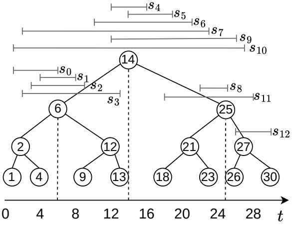



- Range query: returns intervals intersecting range $[q_s, q_e]$.



# Interval Tree [Edelsbrunner(1980)]

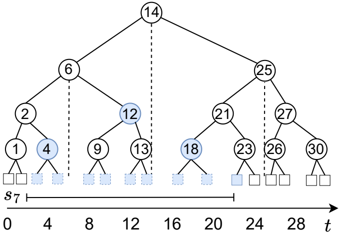
- Construct a BST (Binary Search Tree) by sorting endpoints of intervals.
- Pick the *median* endpoint as the root node recursively.
- An interval is stored at the *highest* node that intersects it.
  - E.g., interval $s_7 = [2, 22)$ is stored in $n_{14}$.




- Secondary structure for each node: two lists of intervals sorted on the two endpoints.

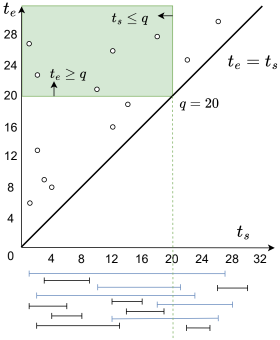
# Segment Tree [De Berg(2000)]

- At the bottom level, each box is a conceptual leaf.
- Each interval is partitioned and stored in *multiple* nodes.
  - An interval is first stored at leaf nodes that intersect it.
  - Consecutive partitions are merged at the upper level recursively, if they cover their parent node.
  - E.g., $s_7 = [2, 22)$ is stored in $n_4$, $n_{12}$, $n_{18}$ and $n^L_{23}$.

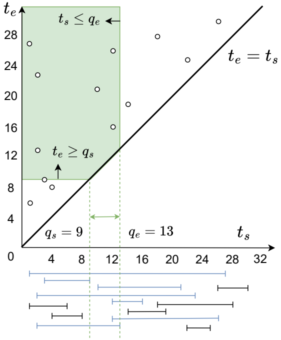


# Diagonal Corner Structure [Kanellakis et al.(1993)]

- Represent intervals as points in a 2D space.
- $t_s$ is the horizontal axis, and $t_e$ is the vertical axis.



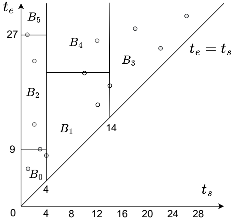

stabbing query



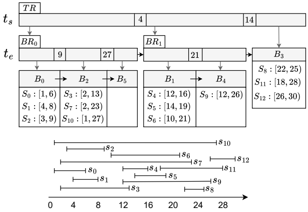

range query

# Start/End timestamp B-tree (SEB) [Song and Roussopoulos(2003)]

- A disk-based corner index
- Increasing Ending Time (IET) assumption: intervals are inserted in increasing order of $t_e$.



Corner structure

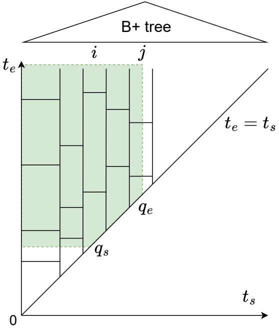


index

- Two layers of *append-only* B+-trees:
  - Top tree: ordering intervals by $t_s$
  - Bottom tree: ordering intervals by $t_e$
- Append-only insertions: at the last data node of some bottom tree.

# Queries in SEB

- SEB must search all bottom trees with $t_s \leq q_e$.
  (Any interval that starts before $q_e$ may die after $q_s$.)

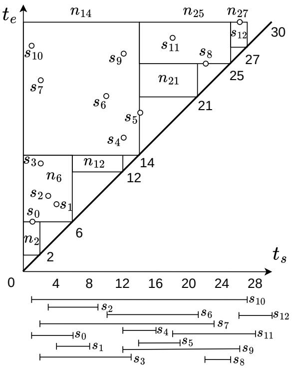


# A Unified Representation for Interval Indexes

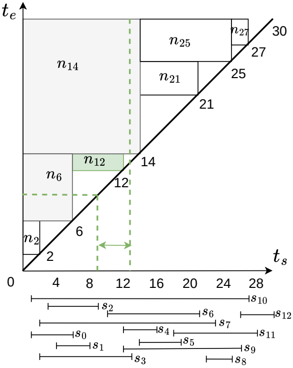
- Interval indexes can be captured by some corner structure in a 2D space defined by $t_s$, $t_e$, or duration $d$.
- Benefit for regular queries
  - Identify nodes that must contain query results (i.e., all intervals can be directly reported).
- Benefit for *aggregate* queries
  - To compute the count of intervals intersecting a range, the number of intervals within each node inside the range can be aggregated directly, without visiting the node.

# Interval Tree in the Corner Structure

- Each node is mapped to a rectangular area. E.g.,
  - $n_{14}$ is mapped to the space of $t_s \leq 14 \leq t_e$.
  - $n_6$ is mapped to the space of $t_s \leq 6 \leq t_e < 14$.



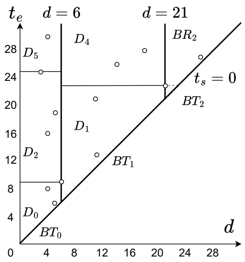

the mapped structure



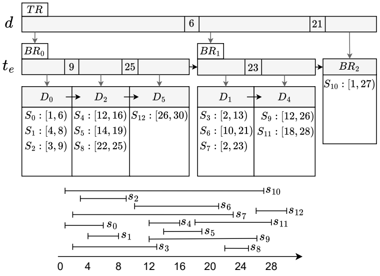

query with range [9, 13]

# TIDE (time intervals by duration and endpoint)

- IET assumption: interval arrive in increasing order of $t_e$. (i.e., they are inserted in the database when they die)




corner structure

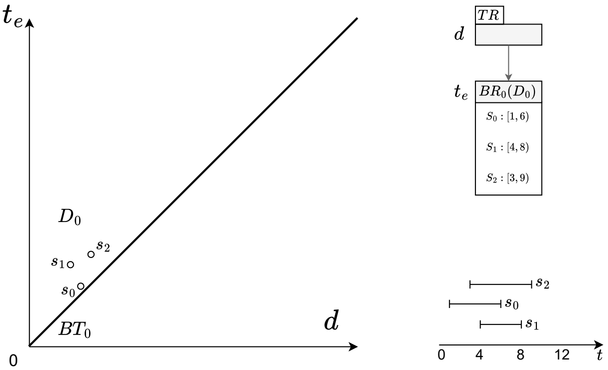



index

- Two layers of *append-only* B+-trees:
  - Top tree: ordering intervals by duration $d$
  - Bottom tree: ordering intervals by $t_e$
- Append-only insertions: at the last data node of some bottom tree.

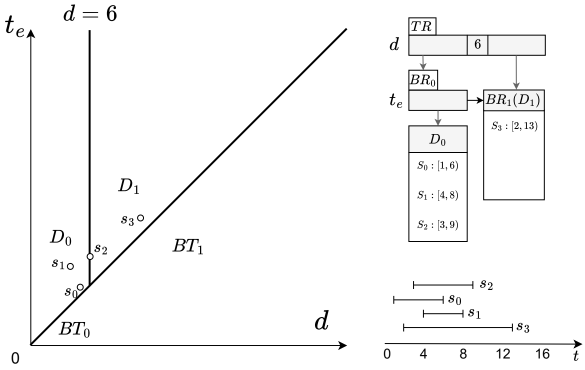
# Insertions of TIDE

- Insert $s_0$, $s_1$, $s_2$ (assuming data node capacity 3)



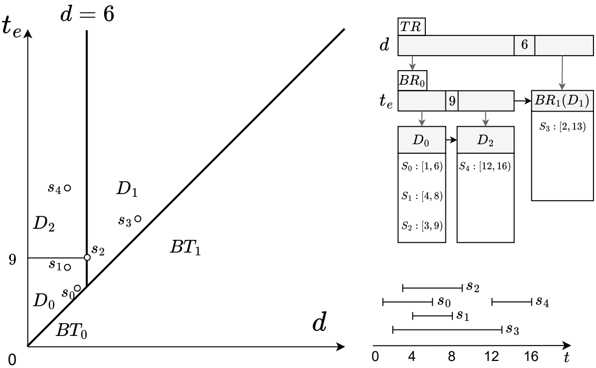
# Insertions of TIDE

- Insert $s_3$: vertical split



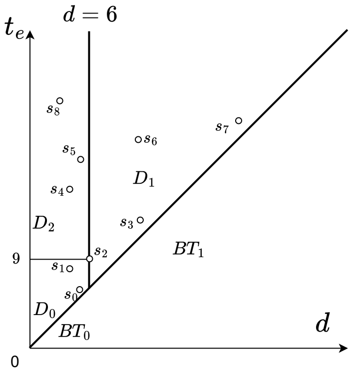
# Insertions of TIDE

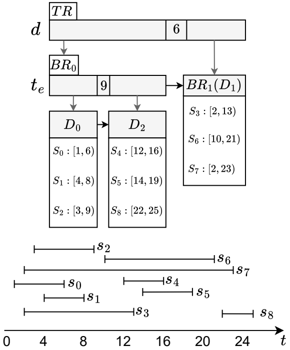
- Insert $s_4$: horizontal split



# Insertions of TIDE

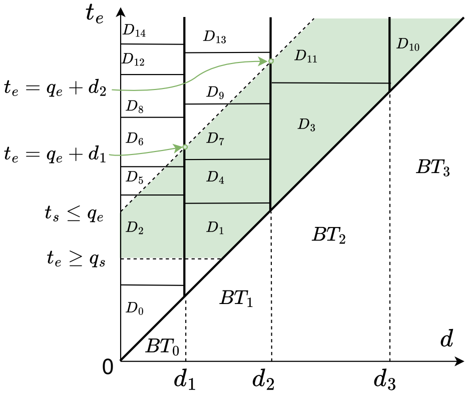
- Insert $s_5$, $s_6$, $s_7$, $s_8$ (according to the duration)



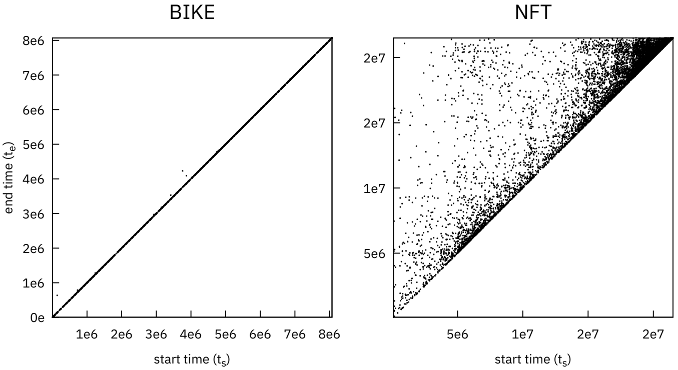
# Queries in TIDE

- Search *all* bottom trees and return intervals fulfilling $t_e \geq q_s$ (horizontal boundary) and $t_s \leq q_e$ (diagonal boundary).
- A stabbing query is a special case of a range with $q_s = q_e$.

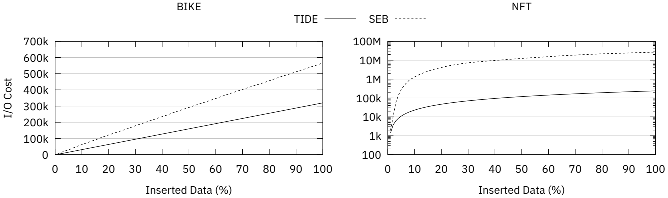


# Datasets

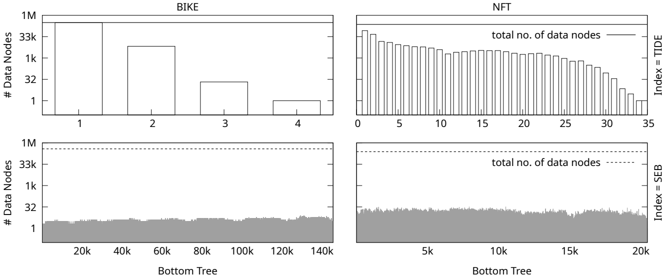
- **BIKE**: Start and end timestamps of 100M bicycle trips, during 2014-2020 in New York City.
- **NFT**: 28M intervals denoting the stable price period of non-fungible tokens transactions, during 2021-2023.



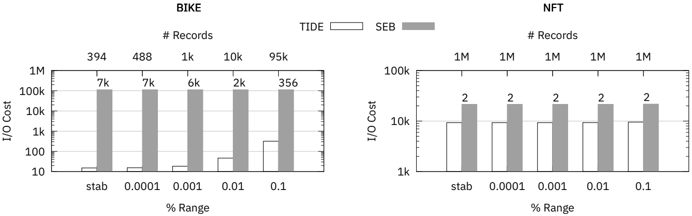
- I/O cost of sequential insertions (4 MiB buffer)




- Data nodes per bottom tree

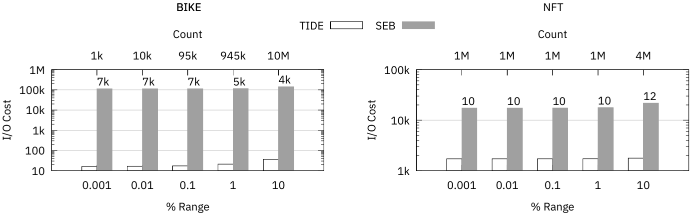


- *Range queries* return qualifying records




- SEB searches numerous bottom trees with $t_s \leq q_e$.
  - For instance, a stabbing query in the middle of the data space is expected to visit at least half of the bottom trees.
- On BIKE, bottom trees with small $t_s$ rarely contain results, although their mutable nodes intersect $q_s$.
- On NFT, even a stabbing query retrieves 1 million intervals, and this number remains almost the same for all ranges.

- *Count queries* return the number of qualifying records




  - Compared to range queries, both TIDE and SEB incur less I/O. (aggregate directly results of full nodes covered by the query)
  - TIDE has shorter nodes that may be covered by the query.
  - SEB examines numerous irrelevant mutable nodes, which cannot be covered since they are open-ended.

# Summary

- We propose a unified representation that
  - treats intervals as points in a 2D corner space
  - enables optimization opportunities for query processing
  - reveals strengths and weaknesses of interval indexes
- Following the IET assumption, TIDE outperforms SEB in
  - compactness and cache locality
    (i.e., most non-full nodes are cached)
  - insertion efficiency (up to x100)
  - query efficiency (up to x7000)

# References I


Mark De Berg. 2000.
*Computational geometry: algorithms and applications.*
Springer Science & Business Media.

H Edelsbrunner. 1980.
Dynamic rectangle intersection searching.
*Technical Report* (1980), 47.

Paris C. Kanellakis, Sridhar Ramaswamy, Darren E. Vengroff, and Jeffrey S. Vitter. 1993.
Indexing for data models with constraints and classes (extended abstract). In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (Washington, D.C., USA) *(PODS '93)*. Association for Computing Machinery, New York, NY, USA, 233–243.
doi:10.1145/153850.153884

Zhexuan Song and Nick Roussopoulos. 2003.
SEB-tree: An Approach to Index Continuously Moving Objects. In *Proceedings of the 4th International Conference on Mobile Data Management (MDM '03)*. Springer-Verlag, Berlin, Heidelberg, 340–344.