JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Fast Indexing for Temporal Information Retrieval

Christian Rauch and Panagiotis Bouros

Institute of Computer Science, Johannes Gutenberg University Mainz, Germany

crauch@uni-mainz.de, bouros@uni-mainz.de

ACM SIGMOD 2026 – Bengaluru, India, June 2026

Temporal IR Setting

Data $o = \left\langle id, \underbrace{[o.t_{st}, o.t_{end}]}_{\text{time interval}}, \underbrace{o.d}_{\text{description}} \right\rangle$

Query For $q = \langle [q.t_{st}, q.t_{end}], q.d \rangle$, find all o with

$$[o.t_{st}, o.t_{end}] \cap [q.t_{st}, q.t_{end}] \neq \emptyset \quad \text{and} \quad q.d \subseteq o.d$$

Example applications:

- *Web archives*: Wikipedia revisions mentioning ‘‘`COVID-19`’’, ‘‘`lockdown`’’ during 2020–2022
- *Streaming*: Listening sessions containing specific tracks during a given month
- *E-commerce*: Shopping baskets including a given set of products within the past week

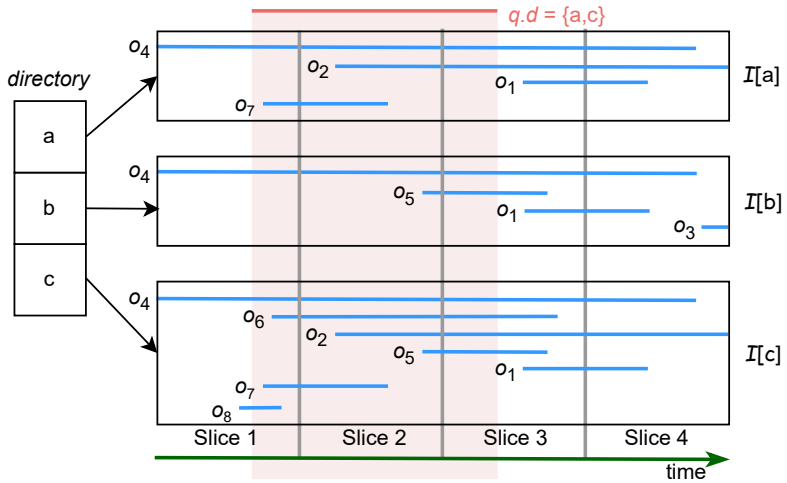
Prior Temporal IR Indices

tIF (no temporal partitioning) Posting lists with $\langle id, [t_{st}, t_{end}] \rangle$

tIF+Slicing [1]

Divides each posting list into disjoint *time slices* (1D-grid)

overlapping intervals are replicated; duplicates are eliminated via [4]

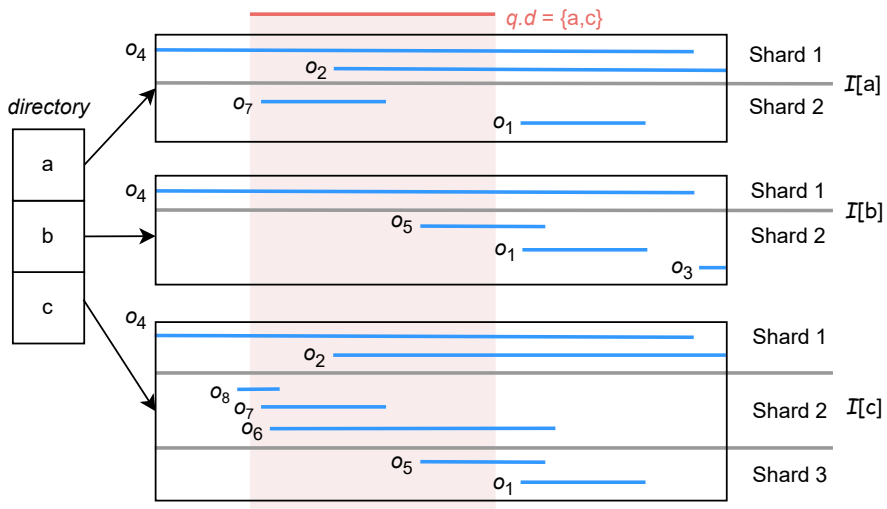


tIF+Sharding [2]

Horizontally partitioned into *shards*, each *staircase-sorted*

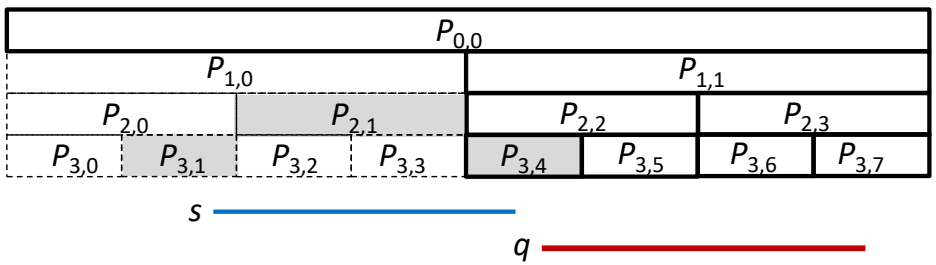
$$o_{st}^{(i)} \leq o_{st}^{(i+1)} \wedge o_{end}^{(i)} \leq o_{end}^{(i+1)}$$

no replication; relaxation of the shard property avoids over-partitioning



Background: HINT

HINT [3] is the state-of-the-art interval index and often outperforms 1D-grid by $\geq$ **1 order of magnitude**



Key idea: Hierarchical binary partitioning with m levels: level ℓ has 2^ℓ partitions, each with two classes:

P^O	*Originals*	interval starts inside the partition
P^R	*Replicas*	interval starts before the partition

Each interval is placed at the *smallest covering partition*, limiting replication. Query processing traverses the hierarchy with efficient access patterns.

Motivation & Contributions

Limitations of prior work

- Suboptimal temporal partitioning: 1D-grid causes interval replication (and deduplication overhead); both methods have limited temporal pruning power.
- IR-first bias: temporal constraint applied only *after* IR filtering. Not ideal for many workloads, especially with selective time constraints or frequent elements.

Our approach

- **IR-first** (*t*IF+HINT): organize each posting list with HINT

  querying per *standard* TIR paradigm: *IR* $\rightarrow$ *time*

- **Time-first** (irHINT): use a single HINT over the time domain; partitions interleaved with inverted indices for IR filtering

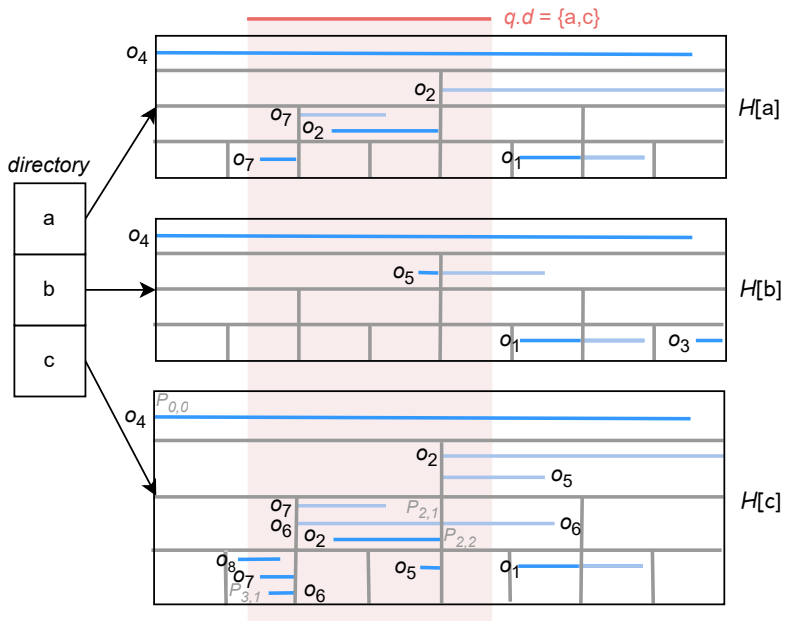
  querying per *novel* TIR paradigm: *time* $\rightarrow$ *IR*

Novel Temporal IR Indices: IR-first

tIF+HINT

Organise posting lists as HINT

limited replication; no deduplication



Variants:

- **Binary-search**: candidates from least-frequent element's HINT; remaining elements checked via binary search
- **Merge-sort**: sort divisions by object id; intersect via merge
- **Slicing hybrid**: HINT for candidates, Slicing for intersections

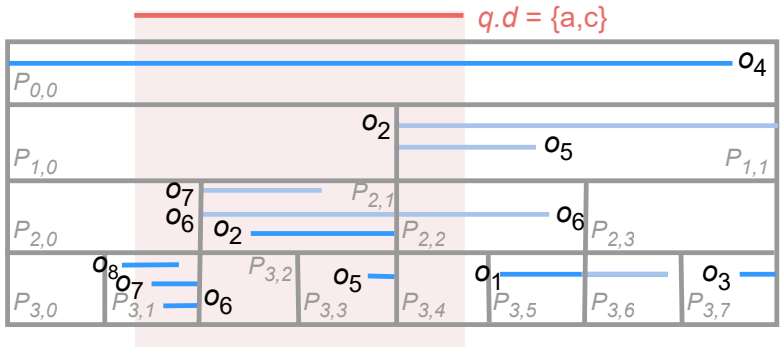
Novel Temporal IR Indices: Time-first

irHINT

Single HINT over the time domain; each partition is injected with an inverted index.

Time constraint processed first via HINT traversal; IR constraint resolved inside each relevant division.

No deduplication needed — HINT's replica principle prevents overlapping results.



element	$\mathcal{I}^O_{0,0}$	$\mathcal{I}^R_{1,1}$	$\mathcal{I}^O_{2,1}$	$\mathcal{I}^R_{2,1}$	$\mathcal{I}^R_{2,2}$
a	$\langle o_4, [\ldots]\rangle$	$\langle o_2, [\ldots]\rangle$	$\langle o_2, [\ldots]\rangle$	$\langle o_7, [\ldots]\rangle$	-
b	$\langle o_4, [\ldots]\rangle$	$\langle o_5, [\ldots]\rangle$	-	-	-
c	$\langle o_4, [\ldots]\rangle$	$\langle o_2, [\ldots]\rangle$,$\langle o_5, [\ldots]\rangle$	$\langle o_2, [\ldots]\rangle$	$\langle o_6, [\ldots]\rangle$,$\langle o_7, [\ldots]\rangle$	$\langle o_6, [\ldots]\rangle$

element	$\mathcal{I}^O_{3,1}$	$\mathcal{I}^O_{3,3}$	$\mathcal{I}^O_{3,5}$	$\mathcal{I}^R_{3,6}$	$\mathcal{I}^O_{3,7}$
a	$\langle o_7, [\ldots]\rangle$	-	$\langle o_1, [\ldots]\rangle$	$\langle o_1, [\ldots]\rangle$	$\langle o_7, [\ldots]\rangle$
b	-	$\langle o_5, [\ldots]\rangle$	$\langle o_1, [\ldots]\rangle$	$\langle o_1, [\ldots]\rangle$	-
c	$\langle o_6, [\ldots]\rangle$,$\langle o_7, [\ldots]\rangle$,$\langle o_8, [\ldots]\rangle$	$\langle o_5, [\ldots]\rangle$	$\langle o_1, [\ldots]\rangle$	$\langle o_1, [\ldots]\rangle$	$\langle o_7, [\ldots]\rangle$

Two variants differ in *how* each division is indexed: **performance** (*t*IF per division) and **size** (conventional IF and HINT temporal arrays)

irHINT: Variants

Performance variant

Each division stores a *t*IF:
$\langle id, [t_{st}, t_{end}] \rangle$ per posting-list entry

- Interval replicated per element $\Rightarrow$ higher storage
- Direct temporal pruning inside each division
- Single-pass per division – temporal and IR filtering interleaved

Size variant

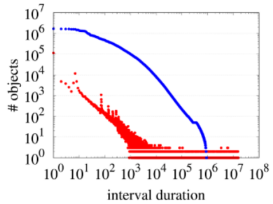
Each division stores *two* structures:
(1) HINT-style temporal array
(2) traditional IF (ids only)

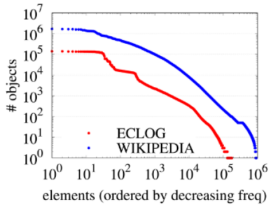
- Intervals stored once $\Rightarrow$ lower storage
- All HINT optimisations applicable (incl. sorting)
- Two-phase query: range query $\rightarrow$ candidates $\rightarrow$ sorted-merge intersect

Experimental Setup

Real Data

	ECLOG	WIKIPEDIA
Cardinality	300311	1672662
Size [MBs]	171	4715
Time domain [secs]	15807599	126230391
Min. interval duration [secs]	1	1
Max. interval duration [secs]	15802098	126169456
Avg. interval duration [secs]	1325118	6587819
Avg. interval duration [%]	8.4	5.2
Dictionary size [# elements]	178478	927283
Min. description size [# elems]	1	1
Max. description size [# elems]	14399	6982
Avg. description size [# elems]	72	367
Min. element frequency	1	1
Max. element frequency	140423	1671696
Avg. element frequency	122	675
Avg. element frequency [%]	0.04	0.05





ECLOG: 300K e-commerce HTTP sessions in $[2019_{12} - 2020_{05}]$

WIKIPEDIA: 1.67M article revisions in $[2020_{01} - 2024_{12}]$

Synthetic Data *varying:*

cardinality, time domain, interval duration (α), description size ($|d|$), element frequency skewness (ζ), and interval position (σ)

Queries *varying:*

description length ($|q.d|$),
interval extent ($q.end - q.st$),
element frequency (% of O containing $e \in q.d$),
result selectivity (% of O matching q)

Experimental Results

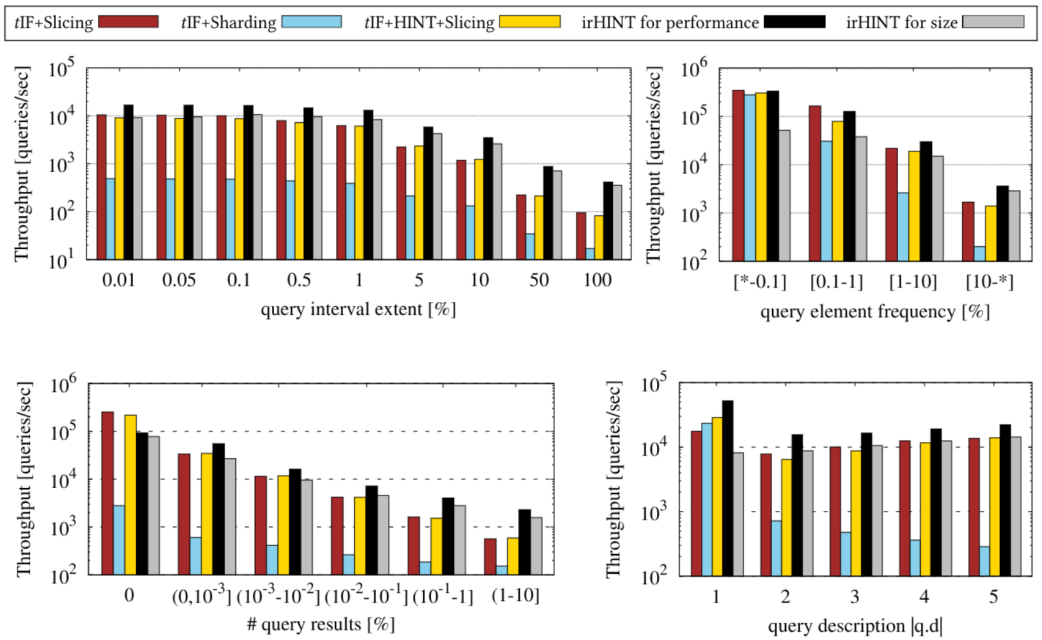
Indexing costs

index		time [secs]	size [MBs]
*t*IF+Slicing		86.8	19150
*t*IF+Sharding		288	7180
*t*IF+HINT	using binary search	529	21221
	using merge-sort	103	9740
	with Slicing	143	22507
irHINT	for performance	580	18022
	for size	594	7124

Maintenance costs: update time in secs

index		insertions			deletions		
		1%	5%	10%	1%	5%	10%
*t*IF+Slicing		1.16	6.10	11.8	6.43	32.5	65.9
*t*IF+Sharding		2.68	16.0	32.0	338	1707	3364
*t*IF+HINT	using binary search	7.23	38.5	76.2	6.62	33.8	67.7
	using merge-sort	1.85	9.97	19.7	4.77	24.2	48.3
	with Slicing	3.23	16.8	33.4	11.6	58.3	118
irHINT	for performance	3.41	17.8	34.1	9.22	46.4	96.3
	for size	5.31	28.0	54.0	15.9	78.4	156

Querying cost



- **irHINT(perf)** usually **outperforms** all IR-first competitors
- **irHINT(size)** has the **lowest size**, still outperforms others often
- **tIF+HINT+Slicing** is the fastest IR-first variant

Conclusions

Weaknesses of prior work

- Suboptimal temporal partitioning
- IR-first bias not ideal for many workloads

Contributions of this work

- Introduction of IR-first indices based on HINT
- Introduction of Time-first irHINTindices (novel paradigm)
- Performance and size advantages over competitors

Future work

- Relevance-based top-k queries
- Compression and parallelisation
- Workload-optimised hybrid indices
- Learned cost models for index synthesis

References

Code `https://github.com/chrauch/irHINT`
DOI 10.1145/3749164

K. Berberich, S. Bedathur, T. Neumann, G. Weikum. *A Time Machine for Text Search*. SIGIR 2007.

A. Anand, S. Bedathur, K. Berberich, R. Schenkel. *Temporal Index Sharding for Space-Time Efficiency in Archive Search*. SIGIR 2011.

G. Christodoulou, P. Bouros, N. Mamoulis. *HINT: A Hierarchical Index for Intervals in Main Memory*. SIGMOD 2022.

J.-P. Dittrich, B. Seeger. *Data Redundancy and Duplicate Detection in Spatial Join Processing*. ICDE 2000.