Signature Index

Full ClickBench Real-Work Signature Query Benchmark

SI as an exact state-signature query engine, not a prediction model

99,997,497
rows

500,000
queries

5
query families

0
mismatches

After a ~19.7 min build, SI served each 100,000-query family in milliseconds to under 1 second, with ~2.42 GB peak RSS.

Public ClickBench data • full local file • external-facing summary

What Was Tested

A focused evaluation of the SI core: exact state-signature queries at scale

What SI did

- **Exact:** How many events match this exact state signature?
- **Threshold / roll-up:** How many events satisfy this broader condition or level?
- **Near-miss L1:** How many events are close to this signature, differing by one state?
- **Broad-to-exact:** Which exact sub-states sit inside a broader state condition?
- **Segment-conditioned:** How does this signature behave inside a selected segment?

What SI did NOT do

- Predict clicks
- Detect anomalies
- Classify users
- Choose which questions are interesting

Public data → Finite-state translation → SI build → Repeated exact signature queries

Goal: isolate the SI core and test whether large families of exact state-signature questions can be served quickly and verifiably.

Dataset & Benchmark Setup

Configuration and inputs used for this Signature Index benchmark

Public data source:	ClickBench hits.parquet
Rows in file:	99,997,497
Observed file size:	~14.78 GB
Columns in source:	105
Columns read in this run:	30
Literal states:	130
Selected signatures:	3,000
Queries per family:	100,000
Total queries:	500,000

Local run profile

Full local file	
Load time:	7.71 s
Build time:	1,182 s (~19.7 min)
Peak RSS:	~2.42 GB

The benchmark used the full public file locally rather than a toy sample.

Main Timing Results

Full ClickBench Real-Work Signature Query Benchmark

SI as an exact state-signature query engine, not a prediction model

Query family	Queries	SI median	SI p95	Per 10k queries	SI speedup
Exact	100,000	0.01097 s	0.01805 s	0.00110 s	**704,270x** vs reference; **439,441x** vs named baseline
Segment-conditioned	100,000	0.01127 s	0.02026 s	0.00113 s	**624,308x** vs reference; **372,104x** vs named baseline
Threshold / roll-up	100,000	0.05567 s	0.05649 s	0.00557 s	**65.4x** vs reference; **95.2x** vs named baseline
Near-miss L1	100,000	0.80927 s	0.92099 s	0.08093 s	**98.2x** vs reference; **21.4x** vs named baseline
Broad-to-exact	100,000	0.01177 s	0.01681 s	0.00118 s	**4,672x** vs reference; **591.8x** vs named baseline

All five 100,000-query families were served in milliseconds to under 1 second after build.

For Exact and Segment-conditioned, the benchmark uses sampled row-scan checks because a full 100,000-query scan workload would imply roughly 10 trillion potential row checks per family.

Why This Matters in Real Work

SI as an exact state-signature query engine, not a prediction model

Work represented by the full catalog

- **Exact:** ~9.9997 trillion potential row checks
- **Segment-conditioned:** ~9.9997 trillion potential row checks
- **Threshold / roll-up:** 300 million dictionary checks
- **Near-miss L1:** 300 million dictionary checks
- **Broad-to-exact:** 300 million dictionary checks

Resource frontier

- **Literal masks:** ~12.1 GB
- **Selected signature masks:** ~279.4 GB
- **All pair flat catalog:** ~780.9 GB
- **All degree-3 flat catalog:** ~33.3 TB
- **Observed peak RSS in this run:** ~2.42 GB

This result is not just one fast query. It turns a massive repeated-work problem into a practical indexed workload.

Public ClickBench data • full local file • external-facing summary

What This Result Does — and Does Not — Claim

This benchmark does claim

- SI gives exact, verifiable answers for a large state-signature workload.
- SI can make repeated exact / threshold / near-miss / drilldown workloads practical after build.
- SI can operate on full public clickstream data with low observed peak RSS in this run.
- SI is strongest where many related state-signature questions must be answered repeatedly.

This benchmark does not claim

- SI is a general replacement for SQL engines, column stores, or dataframes.
- SI is faster than Polars, DuckDB, or ClickHouse on every workload.
- SI predicts clicks, detects anomalies, or decides which questions are valuable by itself.
- A single simple filter is the primary target use case.