Benchmarking the Full Pipeline of Materialized-View-Based Query Rewriting

Xinjie Hu xha102@sfu.ca Zhengjie Miao | Simon Fraser University

Xinjie's Personal Space

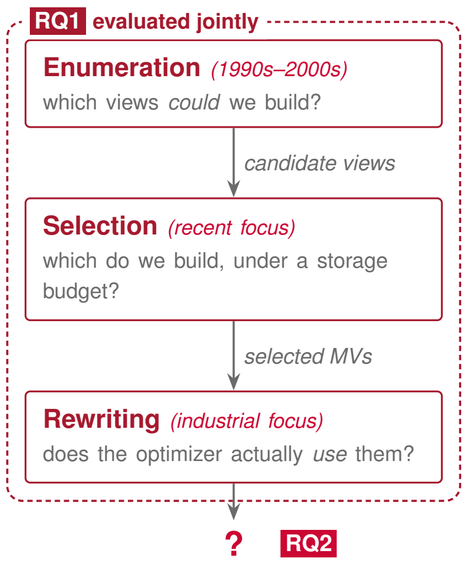
1 The promise

Analytical workloads recompute the **same joins and aggregations** over and over. A materialized view precomputes the shared subexpression once; the optimizer rewrites incoming queries to scan it instead.

Automatic MV rewrite now ships in *every* major warehouse:

Vendors promise it is invisible: "the auto rewrite feature . . . without the query needing to explicitly reference them." — AWS, 2022

2 . . . but it is a three-stage pipeline



End-to-end speedup is **not** the quality of any one stage — it is what survives all three.

RQ1 consistent savings across workloads and budgets?
RQ2 how does behaviour vary across engines that expose only a plan?
RQ3 which stage is the bottleneck — and why?

3 A modular framework

Every stage is **pluggable**. To study one stage we fix the other two over a common candidate pool, view set and engine.

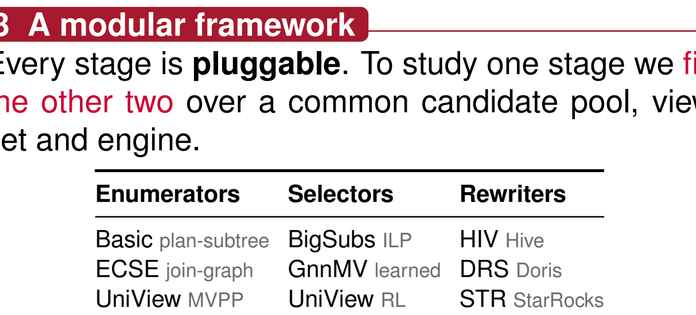
Enumerators	Selectors	Rewriters
Basic plan-subtree	BigSubs ILP	HIV Hive
ECSE join-graph	GnnMV learned	DRS Doris
UniView MVPP	UniView RL	STR StarRocks
Sys-B industrial	Sys-B industrial	CAL Calcite
		Sys-A / B / C

7 engines PostgreSQL, Hive, Doris, StarRocks, Sys-A/B/C
4 workloads JOB, SCALE, STATS, TPC-DS (SF=10)
Budget 1 GB default, swept 0.1–1 GB

Metric relative workload time saving — a regressing rewrite counts as *no* rewrite

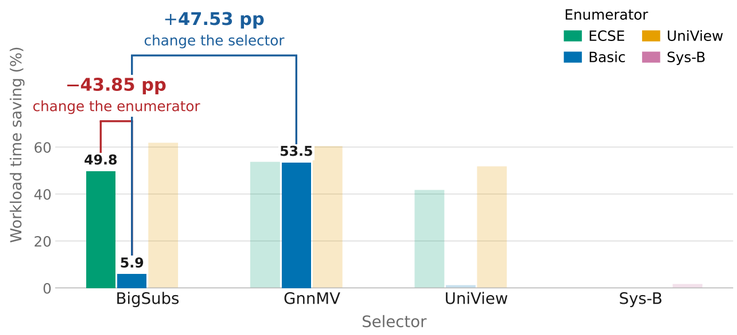
4 Cross-engine protocol RQ2

Most engines are **plan-transparent**: the plan tells you an MV was touched, not *how*. So we pin the MV set and run **both** paths on the *same* engine.



The portable rewrite is a common yardstick, not a competitor: it asks how much of the available benefit each engine's native rewriter actually captures.

5 Stage rankings flip RQ1

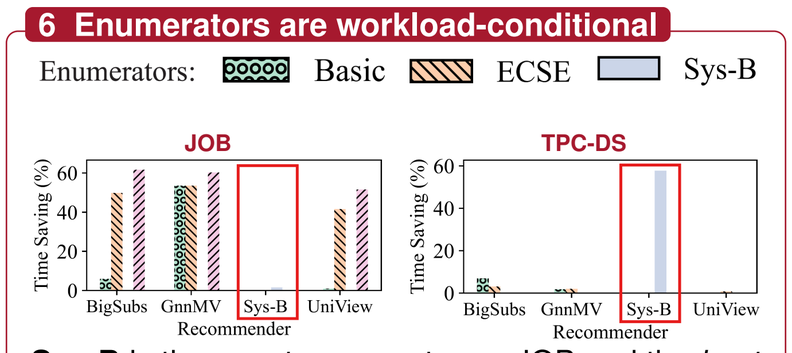


JOB · rewriter = Hive · engine = PostgreSQL · 1 GB

Swap the **enumerator** (ECSE → Basic) under BigSubs and lose **43.85 pp**. Swap only the **selector** on that same candidate set and gain **47.53 pp** back.

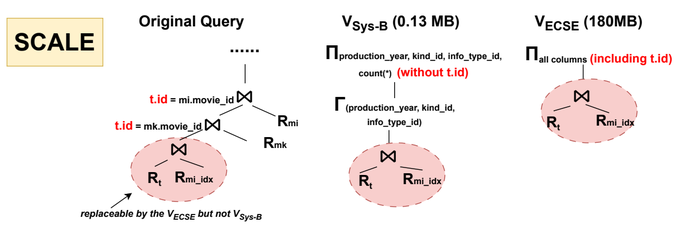
Basic is not a bad candidate set — it is one that is *unforgiving* of a particular selector. **Single-stage rankings can be globally misleading.**

6 Enumerators are workload-conditional



Sys-B is the *worst* enumerator on JOB and the *best* on TPC-DS. Its enumeration performance is strongly workload-conditional — and the TPC-DS reversal has a concrete mechanism.

7 One design choice, two outcomes RQ3

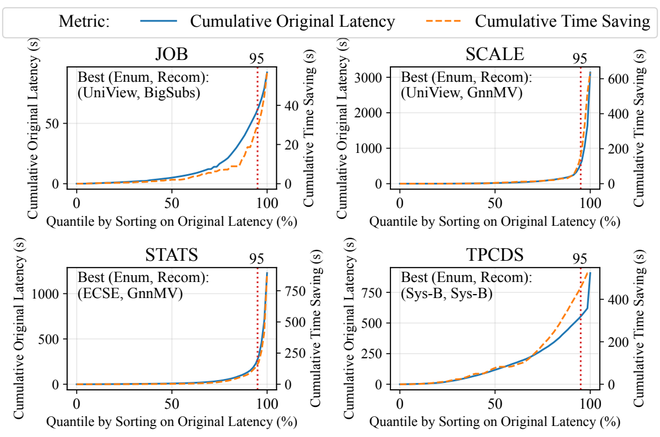


Pruning loses — SCALE. Sys-B's column pruning drops `title.id`, the query's join column. $V_{\text{Sys-B}}$ (0.13 MB) **cannot** substitute; V_{ECSE} (180 MB) can.

Pruning wins — TPC-DS. Same join pattern: Sys-B's 25 views need **30 MB**, ECSE's **2.3 GB**. Under a 1 GB budget only Sys-B fits.

Pruning **buys budget and spends rewritability** — neither direction is universally right.

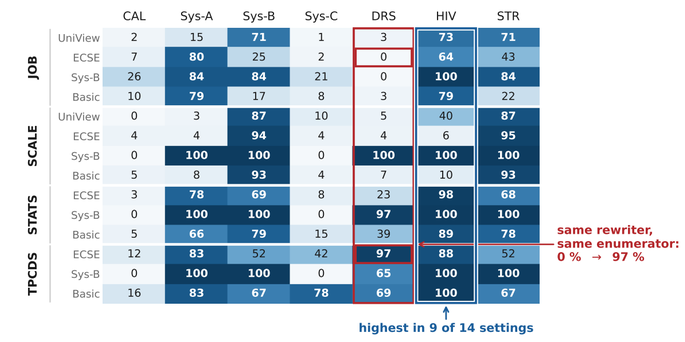
8 Where the savings sit



Cumulative original latency and cumulative time saving, queries sorted by original latency.

Savings are carried by a **small tail of slow queries**: the slowest 5% contribute 51.4% of the total saving on JOB, 71.7% on SCALE and 82.6% on STATS.

9 Rewriting carries the variance



Rewrite success rate (%) — the fraction of query–view pairs the rewriter is able to rewrite.

Same rewriter, same enumerator, different workload: Doris goes **0% → 97%**

Hive is the most consistent — highest in **9 of 14** settings — but StarRocks leads on SCALE. **No universal winner.**

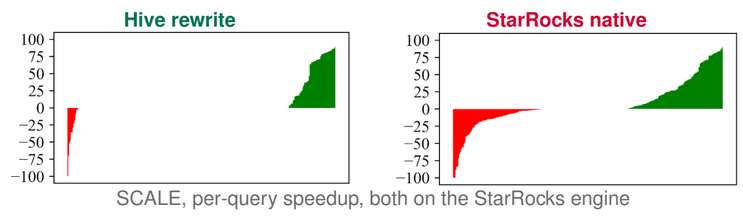
10 Success ≠ benefit

Condition. The MV holds only a *slice* of the base relation, so the rewriter *synthesises* the rest.

Outcome. The base table is read anyway, and the `UNION ALL` introduces an optimization barrier.

❶ the query needs *all* of `title` ❷ the MV covers only *part* of it ❸ so the rewriter pieces it back together:

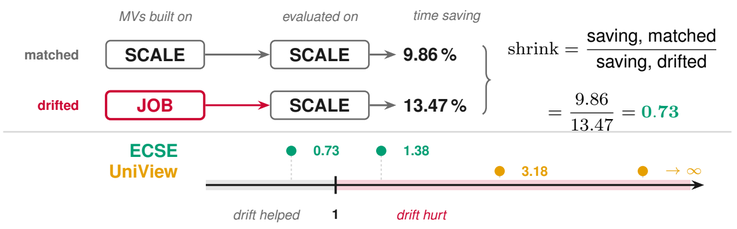
```
SELECT COUNT(*) FROM (
  SELECT id FROM title
  WHERE kind_id <= 1
     OR production_year <= 2013
     OR production_year IS NULL
  UNION ALL
  SELECT id FROM mv
) AS t JOIN ... ;
```



SCALE, per-query speedup, both on the StarRocks engine

Net time saving **−111.83%** — from rewrites that all *succeeded*.

11 Robustness: workload drift



Drift usually costs. UniView keeps a third of its saving — and on SCALE, none of it.

But not always. ECSE built on JOB *beats* ECSE built on SCALE — a richer history can beat in-distribution training.

12 Also in the paper

The poster carries the **core** results. The paper additionally reports:

- **data-distribution skew** (TPC-DS → DSB): Doris' rewritability collapses 99% → 28% while StarRocks *rises* 26% → 57%
- **hardware / memory pressure**: recommendations transfer, rewriting stays engine-sensitive
- **storage-budget sweep**, selector threshold effects, **coverage vs. materialized size**
- a diagnosis of **why Hive rewrites what Doris cannot**, and full results over all **7 engines**

Full protocol and the complete result tables are in the paper; the QR code in the header links to code, data and contact.

Conclusion

1. Optimization by MV is **far from solved**.
2. Stage-by-stage evaluation has limits: the method that looks best in isolation may be worst in context.
3. The bottleneck is set by the *context* — the workload and the engine — not by the algorithm alone.

Practical guidelines.

⇒ Join-heavy workload with heavy-tailed coverage: prefer a **cross-query enumerator**.

⇒ Budget tight against the base data: prefer **compact, column-pruned** candidates — but check the join columns survive.

⇒ Plan-transparent engine: **inspect the negative-speedup tail** before deploying — a successful rewrite is not automatically a faster one.

About Xinjie

Xinjie (Edward) Hu is a PhD candidate at **Simon Fraser University**, Canada.

He received his bachelor's degree from the **Southern University of Science and Technology** (SUSTech), China, and then worked as an engineer at **Sohu**, China, building the query engine and data-governance system for its advertising business.

Research interests
AI4DB optimization; AI task scheduling on GPUs.

Currently
Actively looking for collaboration and internship opportunities.