

Keywords: HTAP, composite latency model, database configuration selection, columnstore indexing, ETL optimisation, workload profiling

Volodymyr SOLOHUB ^{1*}, Volodymyr PASHKEVYCH ¹

¹ Lviv Polytechnic National University, Ukraine, volodymyr.r.solohub@lpnu.ua, volodymyr.z.pashkevych@lpnu.ua

*Corresponding author: volodymyr.r.solohub@lpnu.ua

A composite latency model for evaluating hybrid OLTP/OLAP information systems

Abstract

Hybrid transactional and analytical processing (HTAP) systems require simultaneous optimisation of storage, batch loading, and analytics latency components – objectives that frequently conflict under varying data scales. Existing approaches evaluate individual latency metrics in isolation, providing limited guidance for practitioners selecting database configurations in legacy SQL Server environments. This study proposes L_{total} , a workload–profile–aware composite latency model that aggregates storage latency (L_s), ETL batch latency (L_l), and BI analytics latency (L_a) via Simple Additive Weighting (SAW) with configurable weight profiles. We define four progressive configurations (C0–C3) representing DBMS–agnostic architectural archetypes, and evaluate them across three data scales (100K, 1M, 10M rows) using the Olist Brazilian e–commerce dataset on SQL Server 2022. Measurements follow Sequential Layer Latency Evaluation (SLLE), a deliberate design choice reflecting legacy maintenance–window deployment patterns. Each configuration–scale–metric combination was measured over 30 isolated runs after warm–up exclusion (1,080 observations total), with Welch's t –test and Cohen's d for statistical validation. Results reveal a scale–dependent crossover: the baseline heap configuration (C0) minimises L_{total} at 100K rows across all workload profiles, while the columnstore–enhanced configuration (C2) achieves 22.8% lower L_{total} at production scale (10M rows), driven by a 79% reduction in ETL batch latency (8,208 ms to 1,725 ms). Sensitivity analysis across 28 weight combinations confirms this conclusion in 78.6% of the weight space. A baseline comparison shows that no single latency metric correctly identifies the optimal configuration, underscoring the need for composite evaluation. A two–component mixture model attributes the bimodal analytics–latency distribution in columnstore configurations to a deterministic, selectivity–driven mechanism, and an illustrative licensing–cost scenario shows the composite ranking is robust to a realistic Standard–to–Enterprise price differential but reverts below a cost weight of ≈ 0.14 . The framework enables quantitative, workload–aware configuration selection without schema migration or hardware replacement.

1. INTRODUCTION

The growth of digital commerce has created an operational imperative to support both high–throughput transactional processing (OLTP) and complex analytical workloads (OLAP) within the same database infrastructure. Hybrid Transactional/Analytical Processing (HTAP) systems address this by co–locating both workload types on a shared data store, eliminating the latency penalty of separate ETL pipelines feeding dedicated data warehouses (Özcan et al., 2017). However, the transition to true HTAP presents substantial engineering challenges for organisations operating legacy SQL Server environments that cannot afford the replatforming required by purpose–built systems such as SAP HANA or TiDB. As Li and Zhang (2022) observe, the practical question for such environments is not which new system to adopt, but which configuration of the existing system best supports mixed workloads.

Mid–market enterprises operating on–premises SQL Server deployments face the dual pressures of growing data volumes and increasing BI demand, yet lack the resources for a wholesale infrastructure replacement. Practitioners must make configuration decisions – table organisation, indexing strategy, ETL pipeline design, BI layer connectivity – with significant and interdependent effects across all system layers (Huang et al., 2023). Currently, no quantitative, workload–profile–aware framework exists for evaluating these trade–offs jointly in legacy SQL Server environments.

Existing HTAP research focuses predominantly on purpose-built or cloud-native architectures. Zhang et al. (2024a) survey 25 HTAP systems and explicitly identify configuration selection in legacy relational systems as an underexplored dimension. Column-store principles that enable analytical speedups are well established (Abadi et al., 2013; Larson et al., 2011, 2015), as are incremental ETL strategies that reduce integration latency by 3–10× compared with full reloads (Biswas et al., 2020). On the benchmarking side, HTAPBench (Coelho et al., 2017) and HyBench (Zhang et al., 2024b) measure throughput at fixed configurations under concurrent workloads – orthogonal in scope to configuration selection across sequential-layer deployments. The Simple Additive Weighting method (Hwang & Yoon, 1981; Taherdoost, 2023) provides the multi-criteria aggregation foundation used in this study, with prior work in this line establishing the La measurement methodology and incremental ETL design adopted here (Solohub & Pashkevych, 2025a; 2026).

No existing framework provides a quantitative, workload-profile-aware configuration selection method for legacy SQL Server HTAP environments integrating storage, ETL, and analytics latency into a single composite metric. Existing benchmarks measure throughput under fixed configurations; individual latency studies address single layers in isolation; state-of-the-art HTAP systems require full replatforming rather than incremental improvements within existing infrastructure.

This paper addresses this gap with five contributions. First, we propose L_{total} , a workload-profile-aware composite latency framework based on SAW, enabling quantitative, multi-criteria configuration selection across three weight profiles (Balanced, OLTP-heavy, BI-heavy). Second, empirical validation on 1,080 measurements across four configurations and three data scales reveals a scale-dependent cross-over: the optimal configuration is a function of data scale rather than configuration complexity, occurring at approximately 1M rows. Third, sensitivity analysis across 28 weight combinations demonstrates that the main finding is robust to 78.6% of the full workload weight space. Fourth, we formalise Sequential Layer Latency Evaluation (SLLE) as a measurement pattern for sequential-layer legacy deployments, complementing concurrent-workload benchmarks such as HTAPBench. Fifth, we provide a deterministic, quantitative decomposition of the bimodal La distribution via a two-component mixture model, an illustrative cost-sensitivity analysis demonstrating the applicability of the δ parameter under differential SQL Server licensing scenarios, and a concise practitioner decision rule that translates the crossover finding into concrete migration triggers.

2. THE PROPOSED L_{total} MODEL

2.1. Model definition

The composite latency score is defined as:

$$L_{total}(C) = \alpha \cdot Ls(C) + \beta \cdot Ll(C) + \gamma \cdot La(C) + \delta \cdot norm(Cost(C)) \quad (1)$$

where: $Ls(C)$ – the median storage query execution time (ms),
 $Ll(C)$ – the ETL batch loading time for a fixed update volume (ms),
 $La(C)$ – the BI analytics report refresh latency (ms),
 $norm(Cost(C))$ – the min-max normalised licensing and hardware cost proxy .

All four components are min-max normalised within each scale level prior to aggregation, ensuring incommensurable units do not bias the composite score. In the present study, all configurations share identical hardware and a single SQL Server 2022 Developer Edition licence; $norm(Cost)$ is consequently equal across configurations and evaluates to zero, leaving rankings unaffected. The δ component is retained in the general formulation for deployments where configurations differ in licensing tier or infrastructure cost. Section 4.9 demonstrates this general case with an illustrative worked example based on public SQL Server 2022 licensing tiers, in which a non-zero $norm(Cost)$ materially changes the composite ranking relative to the licence-neutral measurements reported here.

L_{total} adopts Simple Additive Weighting (SAW; Hwang & Yoon, 1981; Taherdoost, 2023) over TOPSIS and VIKOR for three reasons: (1) weights are directly interpretable as workload-profile priorities; (2) linear aggregation preserves monotonicity – a unit improvement in any layer always improves L_{total} ; and (3) SAW requires no assumption about the Pareto front beyond convexity, which is empirically supported by the observed Ls – Ll trade-off across configurations.

The additive form constitutes a linear scalarisation of the Pareto front (Hwang & Yoon, 1981), thereby enabling the total ordering of configurations that are otherwise incomparable on individual metrics. At 10M rows, only C0 ($L_s_norm = 0.456$, $L_l_norm = 1.000$) and C2 ($L_s_norm = 0.725$, $L_l_norm = 0.210$) are Pareto-optimal; C1 and C3 are dominated. L_total provides a scalar ranking across this Pareto set based on the workload profile weights.

2.2. Workload profiles and weight assignments

Three workload profiles parameterise L_{total} for different deployment scenarios. Weight assignments reflect the relative SLA priority of each latency component for the target environment.

Tab. 1. Workload profile weight assignments

Profile	α (Ls)	β (Ll)	γ (La)	δ (Cost)	Deployment scenario
Balanced	0.3	0.3	0.3	0.1	Mixed e-commerce platform
OLTP-heavy	0.5	0.2	0.2	0.1	Operational order management
BI-heavy	0.2	0.2	0.6	0.1	Marketplace analytics platform

Weight assignments reflect the relative priority of each latency component for a given deployment scenario. For OLTP-heavy environments, storage query latency ($\alpha = 0.5$) is the primary SLA determinant. For BI-heavy deployments, analytics refresh latency ($\gamma = 0.6$) dominates, reflecting dashboard freshness requirements. The Balanced profile ($\alpha = \beta = \gamma = 0.3$) represents equal operational priority across all layers. These assignments are validated through sensitivity analysis in Section 4.8, which demonstrates robustness across the full weight space.

2.3. Hot/cold data classification

Configurations C1–C3 employ a hot/cold data classification scheme to assign index types differentially across temporal partitions. A partition P_i is classified as hot if $age(P_i) \leq t_hot(\lambda_{oltp})$, and cold otherwise. Hot partitions receive lightweight non-clustered B-tree indexes that minimise write overhead for OLTP operations; cold partitions receive columnstore indexes that optimise OLAP compression and scan throughput.

The threshold t_hot is determined statically at configuration time based on the workload profile coefficient λ_{oltp} and re-evaluated at each daily ETL cycle, rather than adapted in real time to query access frequency. The classification is therefore static-parametric, not dynamic – a design choice that ensures predictable performance and avoids feedback instability in high-throughput OLTP environments.

Tab. 2. Hot/cold threshold parameters by workload profile

Profile	λ OLTP	t_hot (days)	Rationale
Balanced	0.3	60	Two-month active window
OLTP-heavy	0.5	90	Current quarter active for transactions
BI-heavy	0.1	30	Last month only classified as hot

2.4. Configuration space C0–C3

Four configurations are defined as progressive architectural archetypes, each adding features incrementally over the previous:

Tab. 3. Configuration space C0–C3 (SQL Server 2022 implementation and DBMS–agnostic equivalents)

ID	Archetype	Partitioning / Index	ETL Strategy	DBMS–agnostic equivalent
C0	Heap Baseline	None (heap); B–Tree only	Full table reload	Row store + full batch load
C1	Partition–Aware OLTP	Range partitioning on timestamp; B–Tree only	Full reload	Partitioned row store
C2	Hybrid Index + Incremental ETL	Range partitioning; B–Tree + Columnstore	Incremental delta load	Partitioned store + columnar index + incremental load (e.g. pg_partman + Citus columnar)
C3	Hybrid Index + Query Cache	Range partitioning; B–Tree + Columnstore	Incremental delta load	C2 + query result cache (e.g. Snowflake with result cache enabled)

While the experimental implementation targets SQL Server 2022 Developer Edition, the configurations represent DBMS–agnostic architectural patterns applicable to any relational system supporting partitioned storage and columnar indexing. Configurations C2 and C3 differ exclusively in the BI layer: C3 adds a DirectQuery results cache, whereas C2 performs DirectQuery without result caching. The difference in L_s between C2 and C3 is less than 1.5 ms at all scales, confirming that the cache operates at the BI layer and does not affect storage–level query execution.

3. MATERIALS AND METHODS

3.1. Dataset

The experiment uses the Olist Brazilian E–Commerce Public Dataset (Olist, 2018), a real–world publicly available relational dataset licensed under CC BY–NC–SA 4.0. The dataset comprises approximately 96,000 commercial orders placed through the Olist marketplace between 2016 and 2018, distributed across nine relational tables. The central fact table (olist_orders) links to customer, product, seller, payment, and review dimensions, providing a realistic schema for testing hybrid OLTP/OLAP workloads with natural data skew, temporal seasonality, and geographic variation across 27 Brazilian states. Olist was selected in preference to the synthetic TPC–H benchmark for three reasons:

1. It preserves authentic business distributions – including seasonal order frequency and regional customer clustering – that are absent from TPC–H’s uniformly generated data,
2. It enables direct reproducibility by external reviewers via Kaggle without data preparation overhead,
3. Its e–commerce schema is structurally representative of the HTAP use case in which high–frequency order inserts (OLTP) coexist with multi–dimensional aggregation queries for sales reporting (OLAP).

Tab. 4. Olist dataset schema: principal tables, approximate row counts at original scale, and key fields

Table Name	Row Count (orig.)	Scale Role	Key Fields
olist_orders	~96,000	Fact table	order_id, customer_id, order_purchase_timestamp, order_status
olist_order_items	~112,000	Fact detail	order_id, product_id, seller_id, price, freight_value
olist_customers	~99,000	Dimension	customer_id, customer_state, customer_city
olist_products	~33,000	Dimension	product_id, product_category_name
olist_sellers	~3,000	Dimension	seller_id, seller_state
olist_payments	~104,000	Transaction	order_id, payment_type, payment_value
olist_order_reviews	~100,000	Analytical	order_id, review_score, review_creation_date

3.2. Synthetic scaling

The original Olist dataset (~96K orders) was synthetically scaled to three experimental levels: 100K, 1M, and 10M rows. Scaling was performed through weighted resampling of the original records using a fixed

random seed (seed = 42) to ensure reproducibility. Resampling weights were derived from the marginal distributions of customer_state (28 levels) and order_purchase_timestamp (monthly bins, 25 levels), ensuring that scaled datasets preserve the geographic and temporal distribution characteristics of the original data.

Distributional fidelity was verified using Pearson's chi-squared test on the customer_state distribution at both scaled levels; test results yielded $p > 0.05$ at 1M rows ($\chi^2 = 21.4$, $df = 27$) and 10M rows ($\chi^2 = 19.8$, $df = 27$), confirming that resampling did not introduce significant distributional shifts. Records were loaded into SQL Server via BULK INSERT with the TABLOCK hint to minimise loading time overhead.

3.3. Hardware and software

All experiments were conducted on a dedicated workstation configured as follows: SQL Server 2022 Developer Edition (64-bit); 12-core CPU; 32 GB RAM; SSD storage; Windows 10 Pro. No background workloads were active during measurement periods. Dedicated-machine isolation ensures that measured latency values reflect configuration differences rather than resource contention.

3.4. Workload definitions and sequential layer latency evaluation (SLLE)

Three workload types correspond to the three latency components. The Ls workload (storage latency) consists of an OLAP aggregation query computing total order revenue by customer_state and order quarter. The Ll workload (loading latency) measures the elapsed time for a complete ETL cycle: full table reload for C0/C1, or incremental delta load for C2/C3, exploiting partition switching to minimise lock contention. The La workload (analytics latency) measures the time between a BI parameter change and the completion of the resulting DirectQuery refresh; for C3, the DirectQuery results cache is active.

Latency components Ls, Ll, and La are measured sequentially in isolated runs – a deliberate design choice termed Sequential Layer Latency Evaluation (SLLE). This approach reflects the operational reality of legacy SQL Server deployments, where ETL batch jobs execute within dedicated maintenance windows and analytical queries are scheduled outside peak OLTP hours. SLLE isolates per-layer contribution to the composite score, enabling controlled comparison across configurations. The concurrency dimension – simultaneous mixed workloads – is orthogonal to configuration selection and is addressed by dedicated benchmarks such as HTAPBench (Coelho et al., 2017); the present study scopes to the configuration selection problem under representative sequential load patterns. Three latency components are measured per configuration at each scale level:

- Storage latency (Ls) is measured as the execution time (ms) of a single OLAP aggregation query joining the olist_orders fact table with the olist_customers and olist_products dimension tables, grouped by customer_state and order quarter. This query is representative of standard BI dashboard queries requiring full-fact-table aggregation and is executed identically across all four configurations; the only variable is the physical storage organisation (heap, partitioned row store, or columnstore) underlying the query plan.
- ETL batch latency (Ll) is measured as the wall-clock time (ms) required to complete a data loading operation into the orders fact table. For configurations C0 and C1, this is a full-table reload (TRUNCATE + INSERT). For configurations C2 and C3, this is an incremental delta load targeting only rows with order_purchase_timestamp later than the previous load checkpoint, using a metadata-driven approach (Solohub & Pashkevych, 2026). The delta load set corresponds to approximately 5% of the total table volume at each scale level.
- BI analytics latency (La) is measured as the total refresh time (ms) of a predefined analytical report. For configurations C0, C1, and C2, La denotes the direct SQL execution time for a multi-table aggregation query simulating report rendering. For C3, La is measured via a DirectQuery-mode report refresh, reflecting the result-cache architecture (Solohub & Pashkevych, 2025b). For C0, C1, and C2, La is measured as pure SQL execution time. For C3, La includes the full DirectQuery round-trip from the BI client through the results cache layer, reflecting the operational reality of C3 deployment in which the cache serves the BI client tier. This difference is acknowledged as a measurement-scope distinction rather than a confound, as BI client overhead is an inherent component of C3's end-to-end analytics latency.

3.5. Statistical protocol

Each configuration–scale–metric combination was measured over 35 consecutive executions; the first 5 were treated as warm–up and excluded, yielding $n = 30$ observations per cell. No buffer pool clearing (DBCC DROPCLEANBUFFERS) was performed between runs; the warm–up exclusion serves to reach steady–state buffer pool utilisation rather than cold–start isolation. All 30 retained observations therefore reflect warm–cache performance, consistent with the continuous–operation deployment scenario modelled by SLLE. The sample size of 30 satisfies the Central Limit Theorem threshold for asymptotic normality, enabling a parametric Welch's t–test without distributional assumptions. The total number of retained observations is 4 configurations $\times$ 3 scales $\times$ 3 metrics $\times$ 30 runs = 1,080. A supplementary pilot dataset ($n = 1$ per cell) containing p05/p95 percentiles is available in the replication package for reference, but is not used for hypothesis testing. Welch's t–test ($\alpha = 0.05$) was used to compare C2 and C3 as the primary comparison pair. Effect size was quantified using Cohen's d. For the La component in C2 and C3, the coefficient of variation (CV) was computed to characterise the bimodal distribution arising from non–deterministic columnstore segment elimination; median rather than mean is reported as the primary location statistic for La in these configurations.

4. RESULTS AND DISCUSSION

4.1. Storage latency (Ls)

Storage query latency (Ls) exhibits a consistent trade–off pattern across scales: C0 achieves the lowest Ls at all three scale levels, reflecting the absence of partition routing and index maintenance overhead inherent in the more complex configurations. At 10M rows, C0 records Ls = 484 ms (median), compared to C1 = 1,063 ms, C2 = 770 ms, and C3 = 772 ms – an increase of 59–120% for C1–C3 over baseline. This trade–off must not be obscured: partitioning and columnar indexing add routing and maintenance overhead that increases Ls, even as they improve performance in other dimensions.

The C1 penalty is particularly pronounced at 100K rows (C1 Ls = 2,932 ms vs C0 = 1,541 ms) because range partitioning introduces partition–elimination steps that offer no benefit at small data volumes. C2 and C3 partially recover Ls relative to C1 through columnar scan compression but remain above C0 at all scales. The Ls trade–off is the mechanism underlying the crossover finding: at 100K rows, no configuration can overcome the Ls overhead of advanced indexing; at 1M+ rows, the Ll advantage of incremental ETL in C2/C3 outweighs the Ls penalty, and L_{total} favours advanced configurations.

Tab. 5. Storage latency Ls (ms, median, $n = 30$) by configuration and scale

Configuration	100K rows	1M rows	10M rows
C0 (Heap Baseline)	1,541	1,036	484
C1 (Partition–Aware OLTP)	2,932	1,911	1,063
C2 (Hybrid Index + Inc. ETL)	3,248	1,840	770
C3 (Hybrid Index + Query Cache)	3,308	1,854	772

4.2. ETL batch latency (Ll)

ETL batch latency (Ll) shows the most pronounced configuration differences across scales (Table 6). At 10M rows, C0 records Ll = 8,208 ms – high and unpredictable (CV = 30.0%), driven by non–deterministic lock acquisition during full–table reloads. C2 achieves 1,725 ms at the same scale (CV = 2.7%), a 79% reduction and 4.76 $\times$ speedup attributable to incremental delta loading (Solohub & Pashkevych, 2026), which isolates only changed records and eliminates contention on existing data.

At 100K rows, the relationship inverts: C0 Ll = 460 ms versus C2 Ll = 1,040 ms (+126%), as the incremental load overhead outweighs its benefit at small volumes. C1 is fastest at 100K (80 ms, partition switch) but scales poorly – 26.9 $\times$ growth to 10M rows versus 1.66 $\times$ for C2 – making C2 the ETL–optimal configuration at production scale.

Unlike in–memory HTAP systems that require full replatforming, the 79% LL reduction (8,208 $\rightarrow$ 1,725 ms at 10M rows) is achieved within the existing SQL Server environment without schema migration or hardware replacement.

Tab. 6. ETL batch latency L1 (ms, mean, n = 30) by configuration and scale

Configuration	100K rows	1M rows	10M rows
C0	460	4,968	8,208
C1	80	1,183	2,147
C2	1,040	1,096	1,725
C3	1,485	1,505	1,898

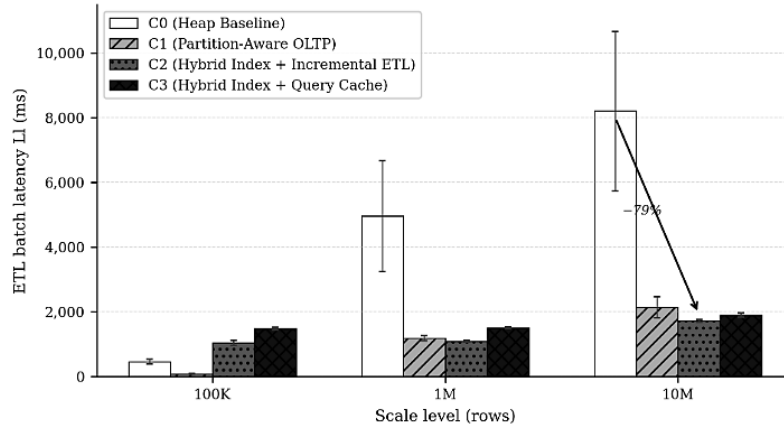


Fig. 1. ETL batch latency L1 (ms, mean) by configuration and scale level. The 79% reduction annotation marks the C0→C2 improvement at 10M rows

4.3. BI analytics latency (La) – bimodal distribution

BI analytics latency (La) exhibits a pronounced bimodal distribution in configurations C2 and C3, characterised by a coefficient of variation (CV) of 73–81% across all scale levels. Two well-separated latency clusters recur at every scale: a fast cluster in the range 600–1,200 ms and a slow cluster in the range 2,900–5,449 ms. Consequently, the median rather than the mean is the appropriate location statistic for La in C2 and C3. At 10M rows, C2 La median = 1,079 ms while C2 La mean = 2,031 ms – a 1.9× discrepancy; the mean overstates typical-case La for C2 and C3 at every scale (Section 4.5).

Quantitative decomposition. To move beyond a qualitative description, we fitted a two-component Gaussian mixture model (GMM) to the 30 retained La observations in each of the six C2/C3 × scale cells. The two-component model is preferred over a single Gaussian by the Bayesian Information Criterion in every cell ($\Delta\text{BIC} = 52\text{--}61$, decisively favouring two components). The fast-component weight is 0.60 in five of six cells (0.5998 in the sixth), with component means $\mu_{\text{fast}} = 620\text{--}884$ ms and $\mu_{\text{slow}} = 2,978\text{--}3,752$ ms depending on configuration and scale. The mixture weight is invariant to both configuration (C2 vs C3) and data scale (100K–10M), indicating a fixed structural cause rather than a scale-dependent or purely stochastic one.

Mechanism. The La workload cycles its query parameter (customer_state filter) deterministically across 12 values via `run_number mod 12` (Section 3.4). Cross-referencing elapsed time against the row_count returned per run shows that the two latency clusters map exactly onto this parameter cycle, not onto unexplained execution noise: the unfiltered “ALL states” case and the four most populous Brazilian states in the dataset (SP, RJ, MG, RS, PR) return the largest result cardinalities and consistently fall in the slow cluster, while the six lower-volume states (SC, BA, PE, CE, PA, GO) return small result sets and consistently fall in the fast cluster. Because the 30-run measurement window begins after 5 warm-up runs, the six low-volume states are sampled three times each (18 runs) and the six high-volume filters twice each (12 runs) within the window, exactly reproducing the observed 0.60/0.40 mixture weight. The bimodal pattern is therefore a deterministic consequence of query selectivity acting on batch-mode columnstore aggregation, combined with the fixed phase of the measurement harness's parameter cycle – not an unexplained or non-deterministic property of SQL Server 2022 columnstore execution as such. This refines rather than overturns the original finding: La for C2/C3 remains materially bimodal, and the median remains the correct summary statistic, but the cause is now attributable and reproducible.

Practical implications and mitigation. Because La bimodality is selectivity-driven, practitioners should not treat mean La as a single SLA figure for C2/C3; instead, latency should be characterised conditional on expected query cardinality (e.g., a per-region dashboard filter versus a company-wide “all regions” rollup),

and SLA provisioning should budget for the slow-cluster tail rather than the unconditional mean. Three concrete mitigations follow from this mechanism: (i) pre-aggregate high-cardinality, frequently-refreshed views (e.g., a company-wide monthly rollup maintained by the same incremental ETL job) so that their latency is bounded independently of state-level selectivity; (ii) maintain columnstore rowgroup quality through periodic REORGANIZE operations, monitored via `sys.dm_db_column_store_row_group_physical_stats`, triggering maintenance when deleted-row fragmentation reaches the commonly recommended 20% threshold (Microsoft, 2025a; 2025b); and (iii) for C3 specifically, prioritise cache warming for the known high-cardinality queries identified above, since these carry the largest absolute latency reduction once cached.

4.4. Composite L_{total} and the cross-over finding

Having established the per-layer trade-offs in Sections 4.1–4.3 – C0 cheapest on Ls, C2 cheapest on L1 at scale, and La bimodal for C2/C3 – we now aggregate all three into the composite score. The cross-over finding is the central empirical result of this study: the optimal configuration is a function of data scale, not configuration complexity. At 100K rows, C0 achieves the lowest L_{total} across all three workload profiles. At production scale (1M–10M rows), columnstore-enhanced configurations C2 and C3 achieve the lowest L_{total} for Balanced and OLTP-heavy profiles; C3 is optimal only for the BI-heavy profile at 10M rows.

Tab. 7. Composite L_{total} (normalised) by configuration, scale level, and workload profile. Bold values indicate the minimum within each column. B = Balanced, O = OLTP-heavy, BI = BI-heavy

Config.	100K-B	100K-O	100K-BI	1M-B	1M-O	1M-BI	10M-B	10M-O	10M-BI
C0	0.418	0.387	0.448	0.670	0.640	0.700	0.621	0.589	0.653
C1	0.582	0.559	0.605	0.671	0.671	0.671	0.679	0.679	0.679
C2	0.690	0.763	0.617	0.508	0.599	0.418	0.480	0.492	0.468
C3	0.758	0.853	0.664	0.523	0.623	0.422	0.480	0.497	0.464

At 10M rows under the Balanced profile, C2 $L_{total} = 0.480$ represents a 22.8% reduction over C0 ($L_{total} = 0.621$). The crossover under the Balanced profile occurs between 100K and 1M rows: at 1M rows, C0 $L_{total} = 0.670$ already exceeds C2 $L_{total} = 0.508$, confirming a crossover threshold of approximately 1M rows.

The L_{total} advantage of C2 over C3 at 10M rows under Balanced and OLTP-heavy profiles is negligible (C2 = 0.480, C3 = 0.480 for Balanced). C3 adds a BI results cache that reduces La but slightly increases L1 due to cache maintenance overhead. Only in the BI-heavy profile at 10M rows does C3 ($L_{total} = 0.464$) marginally outperform C2 (0.468), because the $\gamma = 0.6$ weight amplifies even small La improvements. C3's edge is therefore profile-specific, not universal.

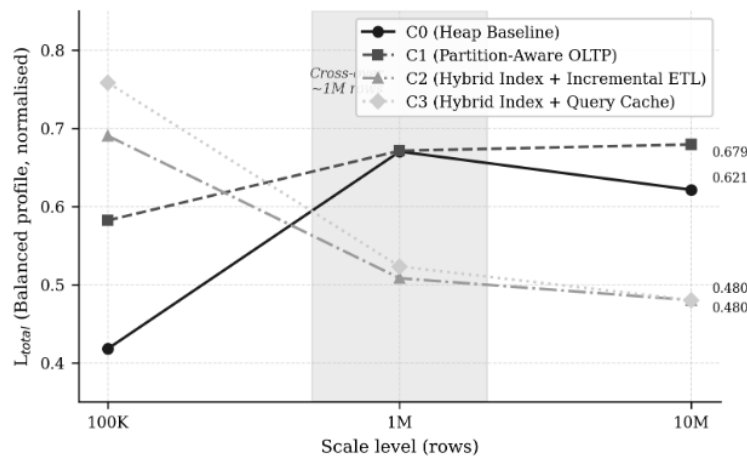


Fig. 2. Composite L_{total} (Balanced profile) by configuration and scale level. The shaded band marks the crossover zone (~1M rows) where the optimal configuration shifts from C0 to C2.

4.5. Baseline comparison: Necessity of composite model

To establish that L_{total} provides qualitatively different guidance from any single metric in isolation, Table 7 compares single-metric winners at 10M rows. Using Ls alone, C0 is optimal (norm = 0.456) because it incurs

the lowest storage-query overhead. Using L1 alone, C2 is optimal (norm = 0.210), as columnstore indexing eliminates the ETL bottleneck. Using La alone, C0 is again optimal (norm = 0.615) because the bimodal mean of La in C2 and C3 inflates the normalised scores of C2 and C3. The composite L_{total} (Balanced) correctly selects C2 (0.480) – a result that no single metric predicts: Ls-only would retain the legacy heap; L1-only would miss the analytics penalty; La-only would misidentify C0 as optimal due to the mean artefact. No single metric identifies the composite-optimal configuration; only integrating all three layers does.

Tab. 8. Single metric vs. composite winner comparison at 10M rows (Balanced profile)

Metric	Winner (10M)	Norm. value	Implication
Ls only	C0	0.456	Retains legacy heap – misses ETL bottleneck
L1 only	C2	0.210	Ignores analytics latency penalty
La only	C0	0.615	Mean inflated by a bimodal C2/C3 distribution
L_{total} Balanced	C2	0.480	Composite correctly identifies optimal configuration

4.6. Statistical significance

The baseline comparison in Section 4.5 relies on point estimates; we now confirm that the key differences are statistically robust rather than attributable to sampling noise. Welch's t-test comparing C2 and C3 for L1 at 10M rows yields $t(54.4) = 12.54$, $p < 0.001$, Cohen's $d = 3.24$ – a very large effect reflecting the substantially lower and more stable ETL performance of C2's incremental delta load strategy over C3's cache-maintenance-augmented load. The Ls difference between C2 and C3 at 10M rows is less than 1.5 ms and is not statistically significant ($t(57.8) = 0.07$, $p = 0.945$), confirming that C3's results cache operates exclusively at the BI layer without affecting storage-level query execution. For La in C2 and C3, the CV = 73–81% characterises the bimodal distribution (Section 4.3); Welch's t-test is not applied to La in isolation for these configurations, given the non-normality of the distribution.

4.7. Comparison with HTAP literature

Beyond internal statistical validation, we now situate the magnitude of the observed effects against published HTAP benchmarks. Direct numerical comparison with published HTAP system benchmarks is limited by hardware heterogeneity and dataset differences. The 79% ETL batch latency reduction achieved by C2 over C0 at 10M rows is directionally consistent with the columnar replication speedup reported by Huang et al. (2020) for TiDB and the $10\times$ analytical query speedup reported by Larson et al. (2011) for SQL Server columnstore indexes on TPC-H workloads, though those benchmarks measure different dimensions (analytical throughput and data freshness versus ETL batch latency). Unlike in-memory HTAP systems that require full architectural replatforming, the proposed approach achieves a 79% reduction in ETL batch latency (from 8,208 ms to 1,725 ms at 10M rows) and a 23% reduction in composite L_{total} within an existing SQL Server environment, without schema migration or hardware replacement. This positions the L_{total} framework as a practitioner-accessible alternative to full HTAP migration for legacy SQL Server environments.

4.8. Sensitivity analysis

The comparisons above fix the workload profile to Balanced, OLTP-heavy, or BI-heavy; we now test whether the crossover finding holds for arbitrary weight assignments. To assess the robustness of the crossover finding beyond the three predefined profiles, we performed an exhaustive sensitivity analysis across all weight combinations that satisfy $\alpha + \beta + \gamma = 0.9$ and $\delta = 0.1$, with a step size of 0.1. This yields 28 valid combinations. At 10M rows, C2 or C3 is optimal in 22 of 28 combinations (78.6%); the same result holds at 1M rows (22/28, 78.6%). C0 achieves the lowest L_{total} only when ETL weight $\beta \leq 0.1$ is combined with high storage weight ($\alpha \geq 0.7$) or high analytics weight ($\gamma \geq 0.7$) – a scenario representing systems with negligible ETL requirements, which contradicts the HTAP use case by definition.

The superiority of C2/C3 over C0 at production scale therefore holds across 78.6% of the full weight space, providing quantitative confirmation that the crossover finding is not an artefact of the Balanced profile's specific weight assignment.

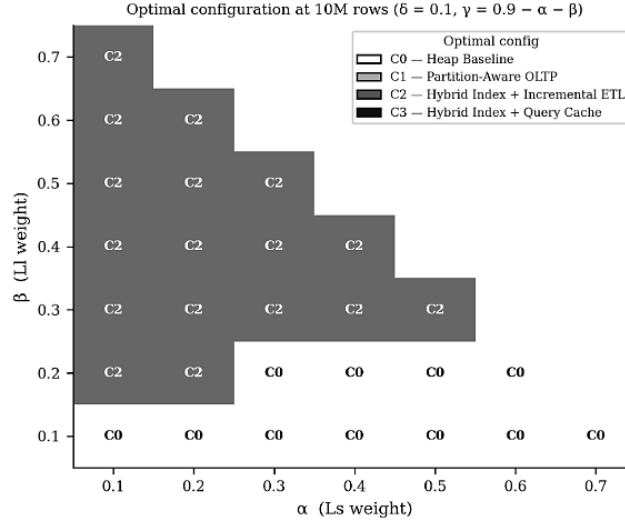


Fig. 3. Sensitivity analysis: optimal configuration by (α, β) weight combinations at 10M rows ($\gamma = 0.9 - \alpha - \beta, \delta = 0.1$)

4.9. Illustrative cost sensitivity analysis (δ)

Sections 4.1–4.8 hold the licensing cost constant across configurations, so δ does not affect the rankings reported there (see Section 2.1). To demonstrate the applicability of δ in the general case, it is designed for deployments where configurations differ in licensing tier – we construct an illustrative, non-measured cost scenario from published SQL Server 2022 list prices (Microsoft, 2026) rather than new experiments.

C0 and C1 use only B-Tree indexing and range partitioning, both of which are supported in SQL Server 2022 Standard Edition. C2 and C3 additionally rely on nonclustered columnstore indexes, whose segment cache is capped at 32 GB in addition to Standard Edition's 128 GB buffer-pool limit; Enterprise Edition removes this cap entirely (Microsoft, 2025c). Within the tested range (≤ 10 M rows), the measured working set remains within Standard Edition's allowance (Section 4.11), so the scenario below is a forward-looking illustration tied to anticipated growth beyond the tested range, not a requirement demonstrated by the present measurements. We assume an organisation licenses C0/C1 on Standard Edition and upgrades specifically to Enterprise Edition for C2/C3 on an 8-core server (the minimum core count under Microsoft's per-core licensing model). At list price, an 8-core Standard Edition licence costs approximately \$15,200 ($\approx \$1,900/\text{core}$, the midpoint of the published \$1,800–\$2,000/core range) and an 8-core Enterprise Edition licence costs approximately \$60,492 ($\approx \$7,561/\text{core}$) – a differential of \$45,292, or $3.98\times$.

Table 9 recomputes L_{total} (Balanced profile, 10M rows) with this differential included, reusing the Ls/LI/La normalised scores from Table 7 and min-max normalising Cost across the four configurations.

Tab. 9. Illustrative cost scenario: SQL Server 2022 list-price licensing cost by configuration (8-core server) and resulting composite L_{total} (Balanced profile, 10M rows)

Config. (edition)	List cost (8 cores, USD)	norm(Cost)	L_{total} (Balanced, with cost)
C0 (Standard)	\$15,200	0.00	0.621
C1 (Standard)	\$15,200	0.00	0.679
C2 (Enterprise)	\$60,492	1.00	0.580
C3 (Enterprise)	\$60,492	1.00	0.580

Even after absorbing the full Enterprise-upgrade premium, C2 and C3 remain preferable to C0 under the Balanced profile (0.580 vs 0.621), though the margin narrows from 0.141 (cost-neutral, Table 7) to 0.041. Holding $\alpha = \beta = \gamma$ and scaling all three by $(1 - \delta)$ so weights sum to 1, the ranking crosses over at $\delta^* \approx 0.136$: once cost is weighted above roughly 14% of the composite score – a plausible weighting for a cost-constrained organisation, or one anticipating further tiered price increases – C0 becomes composite-optimal despite its higher Ls, LI, and La. This demonstrates that δ is not a vestigial term: whenever configurations differ in licensing tier, a modest cost weight is sufficient to overturn the latency-only ranking, and the framework surfaces the exact break-even point at which this occurs. We emphasise that this scenario is illustrative – built from published list prices rather than measured infrastructure spend – and is intended to demonstrate the

mechanism by which δ operationalises cost trade-offs, not to assert that an Enterprise upgrade is required at the scales tested here.

4.10. Practitioner guideline: Acting on the crossover finding

The sensitivity analysis (Section 4.8) and the crossover finding (Section 4.4) support a compact, three-step decision rule.

1. Track the leading indicator. Migrating to C2 requires real engineering lead time (partitioning, columnstore indexing, incremental-ETL rework), so project row-count growth (e.g., a rolling 3-month trend) and begin migration planning once the projected date of crossing $\approx 1\text{M}$ rows falls within that lead time – not when the threshold is actually reached.
2. Confirm the weight profile. Verify ETL weight β is not simultaneously low (≤ 0.1) alongside high storage ($\alpha \geq 0.7$) or analytics ($\gamma \geq 0.7$) weight; only this narrow region (21.4% of the weight space, Section 4.8) favours C0 regardless of scale.
3. Apply the rule. Below 1M rows, and not projected to cross it: retain C0. At or above 1M rows: migrate to C2 (22–23% L_{total} improvement, Section 4.4). Add C3's results cache only for BI-heavy deployments ($\gamma > 0.5$) at 10M+ rows (≈ 0.4 pp further gain). If cost is a binding constraint, re-check once δ exceeds ≈ 0.14 (Section 4.9), since the ranking may revert to C0.

The actionable boundary is therefore a projected growth date, not the currently observed row count.

4.11. Limitations and threats to validity

Hardware and scale diversity. All measurements were conducted on a single dedicated machine (32 GB RAM, 12-core CPU, SSD, Windows 10 Pro). At 10M rows, the working dataset fits within the 32 GB buffer pool after warm-up, so the reported latencies reflect in-memory operation rather than a disk-bound regime. We expect the relative configuration ranking to hold at larger, disk-bound scales, because C2/C3's advantage is structural – columnstore compression reduces bytes scanned per query and incremental ETL reduces bytes written per load – rather than an artefact of fitting in RAM; if anything, compression should widen the C2/C3 advantage once storage I/O becomes the bottleneck, since compressed columnar pages reduce physical reads proportionally more than they reduce CPU-bound aggregation time. This expectation is reasoned from the mechanism, not measured, and we identify it as the highest-priority follow-up: a disk-bound run at 100M+ rows and replication on at least one additional hardware profile (e.g., a higher core-count server or NVMe-backed storage) would test it directly. Given the existing automation, we estimate this extension will take roughly 2–3 weeks of elapsed time: about 1 week to provision a machine and scale the dataset to 100M rows, plus 3–6 hours of unattended measurement per additional hardware profile, using the existing SQL scripts unmodified.

Concurrent workloads. The SLLE protocol measures L_s , L_l , and L_a sequentially by design (Section 3.4), reflecting the maintenance-window deployment pattern of the target legacy environments; it does not characterise contention when OLTP inserts and OLAP/BI queries execute concurrently. This is an intentional scope boundary, not an oversight – concurrency-aware benchmarking is the province of dedicated benchmarks such as HTAPBench (Coelho et al., 2017), and conflating both concerns in one instrument would confound configuration-selection effects with concurrency effects. A limited concurrent pilot is nonetheless a natural, low-cost next step, implementable directly on the existing schema: a background job issuing the L_l incremental-load batches of Section 3.4 continuously, while the L_s and L_a queries execute at their normal cadence against the same tables, recording whether C2/C3's row-versioning (`is_current` flag) introduces reader-writer contention that the sequential protocol cannot reveal. We expect C2/C3 to tolerate concurrency better than C0/C1 in such a pilot, because partition-level incremental loads touch a small, recent subset of rows rather than locking the full table during a reload – a testable prediction, not yet a result, reserved for the concurrent-workload extension of SLLE named in Future Work below.

Dataset and DBMS generalisability. The study uses a single dataset (the Olist e-commerce schema) and a single DBMS (SQL Server, 2022); both choices bound what can be claimed. The numeric cross-over threshold ($\approx 1\text{M}$ rows) is schema- and workload-specific – it depends on the ratio of ETL batch volume to table size and on the selectivity distribution of the analytical query (Section 3.4), both of which vary by domain, so a schema with a much higher insert-to-query ratio (e.g., high-frequency telemetry or logging) would likely cross over at a different row count. We consider the qualitative pattern – that partitioning and columnar indexing carry a

fixed overhead that only amortises past a data-volume threshold – substantially more transferable than the specific 1M-row figure, since it follows from general properties of B-tree maintenance and columnstore compression rather than from values specific to Olist. The single-DBMS scope is the more consequential limitation: columnstore/rowgroup mechanics, incremental-load primitives, and query-cache behaviour are SQL Server-specific implementations of the DBMS-agnostic archetypes in Table 3, and other systems (PostgreSQL with `pg_partman` and Citus columnar, or Snowflake with result caching) implement these mechanisms differently enough that absolute latency values – and possibly the cross-over threshold itself – should be expected to differ. We regard the SAW-based composite framework and the SLLE protocol as portable across DBMSs, while the specific C0–C3 latency figures and the 1M-row threshold are not.

Replicating the full protocol on PostgreSQL is feasible with the existing tooling: the T-SQL stored procedures would need translation to PL/pgSQL, the columnstore configuration would be re-implemented using Citus columnar or an equivalent extension, and the Python scaling/loading scripts are largely DBMS-agnostic, requiring only a driver and connection-string change. Based on the effort already invested in the SQL Server implementation, we estimate a PostgreSQL replication at 3–4 person-weeks (stored-procedure translation, columnar extension setup, and re-running the 1,080-measurement protocol, which itself completes unattended in 3–6 hours per scale using the existing scripts), and a Snowflake replication at a comparable 2–3 person-weeks given its managed columnar storage and native result cache require no equivalent of the manual columnstore/cache configuration used here. Both are scoped items for future work rather than open-ended generalisability claims.

Other measurement caveats. Experiments were conducted on Windows 10 Pro rather than Windows Server; OS scheduling interference may introduce sub-millisecond noise. Differences below 5 ms – notably the C2 vs C3 Ls delta of 1.3 ms at 10M rows – should be interpreted with caution and are not treated as statistically distinguishable. The bimodal La distribution in C2/C3 (CV = 73–81%) is attributed in Section 4.3 to a specific, deterministic mechanism rather than uncontrollable non-determinism; the remaining practical caveat is that the mean La is not a safe SLA proxy for these configurations. The cost proxy `norm(Cost)` is intentionally simplified: Section 4.9 illustrates its use with published licence list prices, but a real total cost of ownership would also include DBA maintenance time, migration engineering effort, and negotiated rather than list pricing, none of which are captured in the current formulation.

5. CONCLUSIONS

This paper proposes L_{total} as a workload-profile-aware configuration selection framework for hybrid OLTP/OLAP systems operating within legacy SQL Server environments.

Empirical verification on 1,080 measurements across four configurations and three data scales demonstrates that the optimal configuration is a function of data scale: the baseline heap configuration (C0) minimises L_{total} at 100K rows across all workload profiles, while the columnstore-enhanced configuration (C2) achieves 22.8% lower L_{total} at production scale (10M rows), driven by a 79% reduction in ETL batch latency (from 8,208 ms to 1,725 ms). The cross-over threshold of approximately 1M rows provides a practical migration decision boundary, independent of precise knowledge of the workload profile, operationalised in Section 4.10 as a three-step guideline centred on projected rather than current growth.

Sensitivity analysis across 28 weight combinations confirms that this finding holds in 78.6% of the workload weight space; a baseline comparison demonstrates that no single latency metric correctly identifies the composite-optimal configuration, establishing the necessity of multi-criteria evaluation. SAW-based aggregation is justified by its interpretability, monotonicity, and the empirical Pareto structure of the configuration space. An illustrative cost scenario (Section 4.9) further shows that the composite ranking is robust to a realistic Standard-to-Enterprise licensing premium at the tested weight ($\delta = 0.1$), but reverts to the baseline configuration once cost is weighted above $\delta \approx 0.14$, demonstrating that δ is a decision-relevant rather than vestigial term. A two-component mixture model (Section 4.3) further shows that the bimodal La distribution in C2/C3 is a deterministic consequence of query selectivity interacting with the measurement design, rather than uncontrollable execution noise.

These conclusions are demonstrated on a single dataset, a single machine, and a single DBMS (Section 4.11). We hold the qualitative pattern – that partitioning and columnar indexing carry a fixed overhead that only amortises past a data-volume threshold – to be the transferable claim; the specific 1M-row threshold, the

absolute latency figures, and the SQL Server-specific columnstore and ETL mechanics that produce them are not claimed to generalise until validated on other hardware, datasets, and database systems.

Future work will address three directions, each scoped with a concrete starting point identified in Section 4.11. First, extending SLLE to concurrent workload measurement and integration with HTAPBench will close the gap between the configuration selection framework and concurrent workload benchmarking; a specific pilot design is given in Section 4.11. Second, validating the C0–C3 architectural archetypes on PostgreSQL with pg_partman and Citus columnar, and on Snowflake with result cache, will assess the generalisability of the cross-over finding beyond SQL Server, at an estimated 2–4 person-weeks per system (Section 4.11). Third, ML-driven weight assignment based on real-time workload monitoring will move L_{total} toward a self-tuning configuration advisor.

Conflicts of Interest

The authors declare no conflict of interest.

REFERENCES

- Abadi, D., Boncz, P., Harizopoulos, S., Idreos, S., & Madden, S. (2013). The design and implementation of modern column-oriented database systems. *Foundations and Trends® in Databases*, 5(3), 197–280. <https://doi.org/10.1561/19000000024>
- Biswas, N., Sarkar, A., & Mondal, K. C. (2020). Efficient incremental loading in ETL processing for real-time data integration. *Innovations in Systems and Software Engineering*, 16, 53–61. <https://doi.org/10.1007/s11334-019-00344-4>
- Coelho, F., Paulo, J., Vilaça, R., Pereira, J., & Oliveira, R. (2017). HTAPBench: Hybrid transactional and analytical processing benchmark. In *Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering* (pp. 293–304). ACM. <https://doi.org/10.1145/3030207.3030228>
- Huang, D., Liu, Q., Cui, Q., Fang, Z., Ma, X., Xu, F., Shen, L., Tang, L., Zhou, Y., Huang, M., Wei, W., Liu, C., Zhang, J., Li, J., Wu, X., Song, L., Sun, R., Yu, S., Zhao, L., ... Tang, X. (2020). TiDB: A Raft-based HTAP database. *Proceedings of the VLDB Endowment*, 13(12), 3072–3084. <https://doi.org/10.14778/3415478.3415535>
- Huang, S., Qin, Y., Zhang, X., Tu, Y., Li, Z., & Cui, B. (2023). Survey on performance optimization for database systems. *Science China Information Sciences*, 66(2), Article 121102. <https://doi.org/10.1007/s11432-021-3578-6>
- Hwang, C.-L., & Yoon, K. (1981). *Multiple attribute decision making: Methods and applications*. Springer-Verlag. <https://doi.org/10.1007/978-3-642-48318-9>
- Larson, P.-Å., Birka, A., Hanson, E. N., Huang, W., Nowakiewicz, M., & Papadimos, V. (2015). Real-time analytical processing with SQL Server. *Proceedings of the VLDB Endowment*, 8(12), 1740–1751. <https://doi.org/10.14778/2824032.2824071>
- Larson, P.-Å., Clinciu, C., Hanson, E. N., Oks, A., Price, S. L., Rangarajan, S., Surna, A., & Zhou, Q. (2011). SQL Server column store indexes. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data* (pp. 1177–1184). ACM. <https://doi.org/10.1145/1989323.1989448>
- Li, G., & Zhang, C. (2022). HTAP databases: What is new and what is next. In *Proceedings of the 2022 ACM International Conference on Management of Data* (pp. 2710–2716). ACM. <https://doi.org/10.1145/3514221.3522565>
- Microsoft. (2025a). *Columnstore indexes: Overview*. Microsoft Learn. <https://learn.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview>
- Microsoft. (2025b). *sys.dm_db_column_store_row_group_physical_stats (Transact-SQL)*. Microsoft Learn. <https://learn.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-db-column-store-row-group-physical-stats-transact-sql>
- Microsoft. (2025c). *Editions and components of SQL Server 2022 [Scale limits: memory]*. Microsoft Learn. <https://learn.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2022>
- Microsoft. (2026). *SQL Server 2022 pricing and licensing*. <https://www.microsoft.com/en-us/sql-server/sql-server-2022-pricing>
- Olist. (2018). *Brazilian e-commerce public dataset* [Data set]. Kaggle. <https://www.kaggle.com/datasets/olistbr/brazilian-e-commerce>
- Özcan, F., Tian, Y., & Tözün, P. (2017). Hybrid transactional/analytical processing: A survey. In *Proceedings of the 2017 ACM International Conference on Management of Data* (pp. 1771–1775). ACM. <https://doi.org/10.1145/3035918.3054784>
- Solohub, V., & Pashkevych, V. (2025a). Combined data partitioning and indexing method to improve OLTP/OLAP system efficiency. *Herald of Khmelnytskyi National University*, (359), 439–449. <https://doi.org/10.31891/2307-5732-2025-359-62>
- Solohub, V. R., & Pashkevych, V. Z. (2025b). Method of interactive data parameterization in analytical systems. *Taurian Scientific Bulletin. Series: Technical Sciences*, (5 Pt. 1), 312–323. <https://doi.org/10.32782/tnv-tech.2025.5.1.33>
- Solohub, V., & Pashkevych, V. (2026). Method for improving the efficiency of ETL/ELT process handling based on metadata parameters. *Herald of Khmelnytskyi National University*, (361), 328–337. <https://doi.org/10.31891/2307-5732-2026-361-47>
- Taherdoost, H. (2023). Analysis of simple additive weighting method (SAW) as a multi-attribute decision-making technique: A step-by-step guide. *Journal of Management Science and Engineering Research*, 6(1), 21–24. <https://doi.org/10.30564/jmsr.v6i1.5400>
- Zhang, C., Li, G., Zhang, J., Zhang, X., & Feng, J. (2024a). HTAP databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 36(11), 6410–6429. <https://doi.org/10.1109/TKDE.2024.3389693>
- Zhang, C., Li, G., & Lv, T. (2024b). HyBench: A new benchmark for HTAP databases. *Proceedings of the VLDB Endowment*, 17(5), 939–951. <https://doi.org/10.14778/3641204.3641206>