Achieving Near Real-Time Data Freshness in Fraud Detection: An HTAP Approach

Matteo G. Giorgino
Department of Computer Information Systems
Faculty of ICT, University of Malta
Msida, Malta
e-mail: matteo.giorgino.18@um.edu.mt

Joseph G. Vella
Department of Computer Information Systems
Faculty of ICT, University of Malta
Msida, Malta
e-mail: joseph.g.vella@um.edu.mt

Abstract—**The rise of complex financial fraud in banking demands sophisticated detection solutions capable of near real-time operations, delivering rapid responses on fresh data. Traditional architectures that connect Online Transactional Processing (OLTP) and Online Analytical Processing (OLAP) through Extract-Transform-Load (ETL) pipelines often fail to satisfy these requirements, particularly when both data consistency and rapid response times are critical. This paper examines how Hybrid Transactional/Analytical Processing (HTAP) architectures can address these limitations by consolidating transactional and analytical workloads within a single system. To assess HTAP's suitability for Fraud Detection in near real-time scenarios, the paper employs HyBench, a benchmarking framework that measures data freshness in centralised HTAP systems, augmented with a custom-made external harness. This setup allows for systematic scenario exploration and detailed performance tracking under realistic banking workloads. Across 48 hours, the evaluation executes 96 parameterised runs at multiple data volumes, with database configurations optimised for the available hardware. Results indicate that an HTAP platform can sustain continuous access to fresh data, achieving sub-20 ms freshness, even under mixed OLTP and OLAP loads, while maintaining high transactional throughput. Although there are efficiency trade-offs compared to standalone OLTP or OLAP deployments, proper system configuration and tuning prove critical for balancing performance and freshness. Furthermore, the flexible benchmarking harness developed here enables practitioners to define custom metrics and integrate additional processing logic into the pipeline, extending beyond HyBench's capabilities.**

Keywords–HTAP System; Near Real-Time Fraud Detection; Database Architecture; Data Freshness; DBMS Benchmarking.

I. INTRODUCTION

Effective Fraud Detection in the financial sector hinges on the availability of relevant, timely and high-quality data [1]. By scrutinising transaction records for distinctive patterns and anomalies, modern systems can distinguish legitimate activities from suspicious or malicious behaviour. The efficacy of these systems depends on computational capabilities that guarantee near real-time data freshness, enabling rapid anomaly detection, and swift pattern recognition across vast datasets. The ultimate objective is to classify each transaction accurately and immediately, preventing fraud without impeding business operations.

In practice, Fraud Detection platforms must ingest data from a variety of heterogeneous sources. Each source contributes unique attributes that, when integrated, offer a comprehensive view of user behaviour. However, differences in data formats, transmission frequencies and endpoint capabilities can introduce synchronisation challenges. Data arrival is often asynchronous, leading to temporary mismatches between sources that must be reconciled to preserve analytical accuracy. Ensuring consistent data quality and low-latency access under such dynamic conditions requires robust strategies for data ingestion, harmonisation and error handling. Without adaptive synchronisation mechanisms, systems risk suffering latency spikes, incomplete data views or compliance gaps that undermine detection performance.

Many organisations have adopted architectures that separate Online Transactional Processing (OLTP) from Online Analytical Processing (OLAP), which are linked by Extract-Transform-Load (ETL) workflows. While this paradigm maintains a clear boundary between transactional consistency and analytical throughput, it introduces inherent latency and operational overheads. Batch-oriented ETL processes can struggle to satisfy the stringent low-latency requirements of fraud detection and prevents OLTP systems from directly leveraging the most recent analytical insights during transaction authorisation.

This paper explores Hybrid Transactional/Analytical Processing (HTAP) systems as a unified architecture capable of addressing these limitations. HTAP platforms merge transactional and analytical workloads, offering continuous, low-latency access to up-to-date data without the need for discrete ETL cycles. By running analytics directly on live transactional feeds, HTAP can enhance both responsiveness and detection accuracy, reducing the window of vulnerability to fraud. We present a detailed evaluation of HTAP performance in a banking context, measuring data freshness, throughput and anomaly-detection latency under realistic workload scenarios. To evaluate HTAP's effectiveness for near real-time Fraud Detection, we select a benchmarking suite that defines and measures freshness on a centralised HTAP system and extends it with a custom-built external harness. This setup allows for systematic scenario enumeration and detailed performance tracking across realistic banking workloads. Our findings demonstrate that, with careful configuration and tuning, HTAP systems can sustain sub-20 ms data freshness while processing high transaction volumes, outperforming OLTP/OLAP deployments in near real-time fraud detection.

Despite their advantages, HTAP systems also come with limitations. Running transactional and analytical workloads concurrently can cause resource contention, especially under heavy loads. Tuning performance requires expertise, and some platforms may lack support for strong consistency or scalability. Upgrading from traditional architectures also involves significant cost and complexity.

This paper begins with Section I, which introduces the problem and motivation. Section II sets out the research aims and objectives. Section III provides background on fraud detection in banking, system development, the role of Database Management Systems (DBMSs), database architectures, HTAP technologies, and benchmarking. Section IV details the schema, benchmarking suite, and experimental setup. Section V presents the results, focusing on the main Key Performance Indicators (KPIs). Finally, Section VI summarises the findings and outlines directions for future work.

II. Aims and objectives

The Fraud Detection System (FDS) aims to enable near real-time classification of fraudulent transactions by ensuring data freshness, and facilitating ad hoc pattern recognition, thereby addressing the operational limitations of traditional FDSs. In achieving this aim, a set of objectives was laid down:

1) Choosing and adapting an HTAP benchmarking suite, including setting up both separate OLTP and OLAP components for testing.
2) Seamlessly running both operational and analytical operations on an HTAP computational set-up.
3) Evaluating the performance of various runs using the chosen HTAP configuration and different data movement scenarios.

III. Literature review

A. *Fraud Detection in Banking*

Fraud is defined as any activity that relies on deception to achieve a gain. Banks are among the most obvious targets for fraudsters seeking financial gain. Given the complexity of modern fraud schemes and the sophistication of fraudulent behaviour, the financial industry faces some of the toughest detection and prevention challenges, necessitating advanced systems that monitor multiple sources of data.

A recent study revealed that fraud in the banking industry has become a matter of grave concern for almost all countries across the globe, causing significant financial and non-financial damages to banks, customers, other stakeholders and economies [2]. Fraud in banking also results in reputational damage and compliance challenges. Regulatory compliance further necessitates robust Fraud Detection frameworks to protect sensitive data and ensure adherence to legal standards and frameworks.

B. *Fraud Detection Systems Development*

Traditional FDS primarily rely on rule-based approaches, which flag suspicious activities based on predefined criteria, such as unusually high transaction amounts, frequent transactions, or geographical discrepancies [3]. These static rules struggle to adapt to the continuously evolving tactics employed by fraudsters.

Machine Learning (ML) techniques are increasingly being integrated into traditional FDS to address some of these limitations, as discussed by Minastireanu [4]. These models typically employ supervised learning models trained on historical data [5]. They exhibit reduced efficacy against novel fraud schemes that deviate significantly from historical fraudulent patterns. Additionally, these ML approaches often require substantial time and resources for periodic retraining and validation to maintain their relevance.

A further notable limitation of traditional systems is the separation of the transactional data, which is used in day-to-day activities, from the analytical data utilised in reporting and analysis. These distinct operational environments are often optimised differently, as discussed by Camilleri et al. [16], impeding the integration of real-time transaction analysis with historical data evaluation. Lastly, the broader advancement of Fraud Detection methodologies suffers due to restricted knowledge-sharing within the public domain.

C. *Database Management Systems for Fraud Detection*

DBMSs are critical components in Fraud Detection architectures, serving as the backbone for processing large volumes of transactional and analytical data. As banks grapple with increasing transaction volumes and sophisticated fraud schemes, traditional DBMSs face limitations, prompting the exploration of more advanced DBMS architectures that can provide near-real-time responses, robust scalability, and comprehensive data integration.

A DBMS ensures that data remains available to users and applications, handles increasing volumes of data without loss of data consistency, and can scale to meet rising demand. DBMSs play an increasingly crucial role in supporting Fraud Detection mechanisms in banking. The ability of a DBMS to process, store, and retrieve data efficiently in a structured manner can significantly improve the quality of Fraud Detection efforts.

At its core, an FDS must rapidly ingest large volumes of heterogeneous data, enforce stringent data quality and integrity constraints, and support ad hoc analytical queries. Nevertheless, relational DBMSs have encountered bottlenecks in performance and scalability, especially in transactional processing [6]. Detecting patterns of suspicious transactions requires the ability to query and analyse historical data alongside real-time transactions.

To address these challenges, the DBMS landscape has diversified into multiple directions. Noticeably, columnar with in-memory analytical platforms are enabling HTAP by integrating OLTP and OLAP workloads within a single engine. By maintaining online materialised views or employing Multi-Version Concurrency Control (MVCC) optimised for mixed workloads, HTAP systems permit on-the-fly computation of fraud scores without the latency penalties of ETL pipelines [7].

Practical performance depends heavily on real-world data patterns, which are driven not just by the underlying schema

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

but by unpredictable end-user behaviour, which make static optimisation insufficient. This unpredictability creates additional challenges in scaling transactional throughput while preserving data consistency guarantees.

D. Database Architecture for Fraud Detection

In many architectures, OLTP and OLAP systems operate and are set-up in isolation. OLTP databases are optimised for light transactions with few tuples in its scope, handling inserts, updates and deletes with ACID (Atomicity, Consistency, Isolation and Duration) guarantees, whereas OLAP engines are tuned for complex, read-heavy queries against large datasets. To bridge these worlds, ETL pipelines are constructed to periodically move data from OLTP systems into purpose-built Data Warehouses (DWH) to support OLAP. This also introduces latency: data freshness is bounded by the ETL cadence, and the overhead of maintaining dual schemas with the pipeline code can be substantial [8].

There are two major challenges organisations face to maintain a data infrastructure that caters for their data driven decision making, namely; (i) performance in terms of query response time, transactional throughput, and resilience to computational failures, and (ii) moving data from internal and external data sources to build an integrated, synchronised, business-process focused, and time-variant data repository, i.e., a Data Warehouse [9].

If data freshness becomes a requirement, as in a banking FDS, then the architecture with ETL does not meet it. HTAP architectures seek to collapse the boundary between OLTP and OLAP by introducing a computational set-up that supports both workloads within a single, integrated engine and instance.

This unified approach can dramatically reduce data movement overheads, eliminate ETL-induced data staleness, and simplify system maintenance. However, it does demand advanced engine optimisations and comes at a loss of the decoupling and staging flexibility offered by ETL pipelines. DWHs and OLAP asynchronous updates are typically not suitable for real-time Fraud Detection, as they are designed to process large datasets in batches, resulting in delays that could allow fraudsters to act before fraud is detected, as OLAP is querying stale (i.e., not fresh) data [10].

E. HTAP Architectures and Techniques

HTAP, attributed to Gartner in 2014 by Zhang et al. [8], describes a database architecture that unifies transaction processing and analytics in near real-time, allowing OLTP and OLAP workloads to run side by side without undue interference, as seen in Figure 1. By collapsing separate systems and eliminating complex ETL pipelines [11], HTAP simplifies data management. However, supporting both transactional and analytical demands simultaneously remains challenging, given their inherently different performance and resource requirements.

Overall, supporting both transactional and analytical queries on the same dataset increases the risk of data contention in HTAP systems, necessitates careful management of isolation levels, concurrency controls, and workload balancing strategies is essential.

HTAP remains a conceptual model rather than a formally standardised architecture, and there is currently no broad consensus on the best way to implement it. One common architecture is the primary row store with an in-memory column store, where the primary storage is a row-oriented database optimised for transactional operations, and an in-memory column store used to handle analytical queries. This design facilitates real-time analytics without compromising transactional performance.

The choice of architecture depends on specific application requirements, workload characteristics and resources available. For instance, high-volume transactional systems favour OLTP for fast, consistent writes, while analytics-heavy workloads benefit from OLAP's read-optimized design. Hybrid systems like HTAP are more suitable when near real-time insights are needed without sacrificing transactional integrity, such as in fraud detection scenarios. Additionally, resource constraints (e.g., limited memory or compute power) may dictate the use of simpler architectures, whereas larger organizations with more capacity can adopt in-memory or distributed architectures for greater responsiveness and scale.

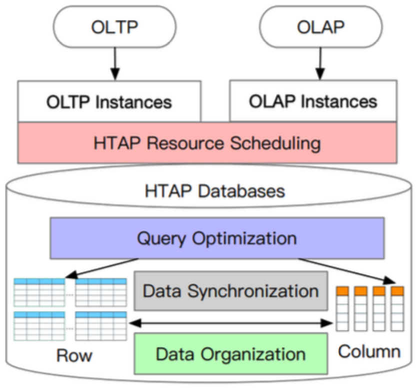

Figure 1. HTAP System Overview [15].

The primary challenge in an HTAP system is to facilitate the smooth flow of data from all these different sources into a unified platform where both transactional and analytical operations can occur. Ensuring that transactional updates and analytical views remain tightly consistent is critical for applications that demand ACID guarantees, such as Fraud Detection in banking. Thus, HTAP platforms in banking must implement strong consistency mechanisms to synchronise updates from OLTP to OLAP stores with minimal latency and without sacrificing throughput.

F. Benchmarking

Benchmarking database systems underpins objective performance assessment, offering repeatable tests that measure metrics, such as throughput, latency or completion time against standardised workloads [11]. A typical benchmark defines a data schema, data volume, workloads (queries and transactions) and an evaluation criterion. These tests serve multiple purposes: identifying raw horsepower through intensive tuning; comparing enhancements or configuration changes; and contrasting architectures or hardware choices under controlled conditions. Fairness, consistency and conservatism in benchmark design are essential to yield meaningful insights.

Several HTAP-specific benchmarks have emerged [13]. Early efforts, such as CH-benCHmark extend the Transaction Processing Performance Council's (TPC) benchmarks TPC-C and TPC-H by executing transactional and analytical workloads on the same dataset, thereby revealing workload contention effects [14]. However, these benchmarks still lack the depth required for domains like banking, where anomaly detection and time-sensitive analytics are paramount.

HyBench [15] was developed to address these precise challenges. It integrates realistic banking transactions alongside analytical tasks like trend analysis. Its workload comprises 18 transactional operations and 13 analytical queries executed under mixed scenarios, mirroring real-world HTAP use cases more closely than its predecessors. By focusing on financial workloads and anomaly-driven scenarios, HyBench offers a comprehensive evaluation framework that blends transactional throughput, analytical query performance and data freshness. Its extensible design allows practitioners to adjust parameters, incorporate new metrics or inject domain-specific logic, making it a versatile tool for advancing HTAP research and guiding production deployments in latency-sensitive environments.

IV. Design, Implementation and Testing

G. System Design

HyBench forms the foundation of our evaluation framework, simulating both transactional and analytical workloads on a unified schema mimicking typical banking entity, such as customers, accounts, transactions, loans and companies, with SFs of 1x, 10x and 100x factors to approximate 1 GB, 10 GB and 100 GB of base data, respectively. A harness orchestrates the workflow: it restores a clean database snapshot, vacuums and analyses tables to eliminate fragmentation, pre-warms the cache using a representative query, then launches concurrent clients for:

- Transactional Processing (TP) workloads, with high-volume, short-lived OLTP operations
- Analytical Processing (AP) workloads, with long-running, scan-intensive OLAP-style queries
- Hybrid Processing (XP) mixed workloads, combining both TP and AP workloads

All tests were conducted on a standalone machine (macOS, M3 Pro, 16 GB RAM, 500 GB SSD), using PostgreSQL 16, with pg_stat_statements enabled to record detailed query metrics. The Python 3.13.2 harness performed orchestration and monitoring, while Java components (via OpenJDK 21) drove the HyBench workload. Key PostgreSQL configurations were tuned, allowing up to 100 concurrent connections, allocating 4 GB each to shared_buffers and effective_cache_size, disabling autovacuum, and setting work_mem to 64 MB. These configurations were chosen based on the available hardware capacity and industry best practices.

During each benchmark, we gathered metrics from both HyBench logs and the Python harness, e.g., query performance data—including total and mean execution times (ms), execution counts, rows returned, shared block hits and reads, and the query text—table activity counts for inserts, updates and deletes, lock information detailing lock types and their frequencies, transactional throughput figures for committed transactions per table, as well as total transactions (committed plus rolled back), and storage statistics listing schema and table names alongside total table and index sizes.

Together, these KPIs offer a comprehensive view on resource use and contention under concurrent OLTP/OLAP conditions. To unify these dimensions, HyBench's creators introduced the H-Score. This unified metric incorporates Transactions Per Second (TPS), Queries Per Second (QPS), mixed workload throughput (XPS = TPS + QPS for mixed workloads), data freshness (f_s) and Scale Factor (SF) to yield an overall performance rating. The H-Score is defined as the geometric mean of all throughputs, multiplied by the SF and divided by the freshness metric, as in (1).

$$H\ Score\ =\ SF\ \times \frac{\sqrt[3]{TPS \times QPS \times XPS}}{f_s + 1} \qquad (1)$$

H-Score is beneficial as solely relying on one aspect cannot reflect the true HTAP performance. The five components in H-Score are widely recognised by benchmarking suites as the most important factors for quantifying the HTAP performance [10].

H. Implementation

We defined 11 core parameter dimensions to explore various workload mixes and data volumes, producing 144 unique configurations, as seen in Table 1, but we ultimately retained only 96 combinations (i.e., the 1x and 10x SF ones) after observing inconsistent KPI behaviour at 100x, namely scalability limits in HyBench's threading and PostgreSQL's I/O performance on given computer set-up.

By adjusting these parameters, we could evaluate scenarios with standard transactional-heavy loads (3:1 TP:AP) and more analytics-intensive mixes (3:2 TP:AP), as well as stress-test at 100x SF.

A Python harness was built to minimize manual intervention. The script handled the below process:

1) Generate the HyBench '.props' files programmatically for every parameter combination.
2) Restore and reset the database from a known backup via 'pg_restore' utility, then execute VACUUM FULL ANALYZE on all tables to rebuild storage and refresh Data Dictionary (DD) statistics.
3) Prewarm the cache using a realistic query that joins key tables and exercises index and sequential scans in parallel, ensuring relevant pages are memory-resident.
4) Launch HyBench with a '.props' configuration file, while a background process queries the DD views, pg_stat_statements, pg_locks, pg_stat_user_tables and pg_stat_database every 60s for live metrics.
5) Parse the logs and plot key performance indicators.

TABLE I. HYBENCH PARAMETER CHOICES

Parameter	Set Value	Description
sf	1x, 10x, 100x	Scale factors for the table data
at_percentages	(35,25,15,15,7,3), (3,7,15,15,25,35), (10,10,20,20,20,20)	AT Ratio (sum = 100%)
apclient	10	AP concurrency
tpclient	15, 30	TP concurrency
fresh_interval	150	Freshness evaluation is done every (xpRunMins/150) s
apRunMins	5, 10	AP evaluation time
tpRunMins	5, 10	TP evaluation time
xpRunMins	5, 10	XP evaluation time
xapclient	10	XP-ATS concurrency
xtpclient	15, 30	XP-IQS concurrency
distribution	Uniform	Data distribution at generation phase

I. *Testing*

We began with unit tests of individual harness modules (e.g., DB housekeeping, monitoring). Next, integration tests ran end-to-end workflows using varied '.props' files to validate the sequence: restore → housekeeping → prewarm → workload → monitoring. Logs from PostgreSQL and HyBench were cross verified to ensure consistency, accounting for differences in granularity (e.g., DBMS internal vs. atomic operations launched by HyBench). All runs were performed in a controlled environment with server resource prioritisation to avoid external interference. Three main scenarios were executed – each taking 24 hours:

1) 1x: Establish expected performance and tune PostgreSQL parameters to avoid configuration-induced artefacts.
2) 10x: Stress test concurrency, observe degradation in freshness and throughput, and validate that HTAP sustains desired freshness under mixed workloads.
3) 100x: Identify limits of the harness and the hardware set-up available for the DBMS, revealing thread-management issues, Java Database Connectivity termination overheads, and I/O saturation that rendered metrics unreliable (consequently it was discarded for evaluation).

For each run, we generated data (via HyBench's gendata module), created indexes and tables, and ran the full benchmark cycles. Post-run validation included checking table and index sizes, transaction counts and comparing relative throughput trends against published HyBench with PostgreSQL baselines to confirm that our findings aligned qualitatively with prior results.

V. EVALUATION

Across both 1x and 10x data volumes, the HTAP setup consistently delivered sub-20 ms data freshness, demonstrating its ability to service the latest transactional changes to analytical queries almost instantaneously. At 1x scale, the F-Score typically ranged from 2 ms to 12 ms, with occasional peaks near 12 ms under high contention; at 10x, peaks rose only slightly—up to around 15 ms—confirming robust freshness even under heavier mixed loads. This encouraging performance underpins real-time use cases where even small delays can blindside FDS.

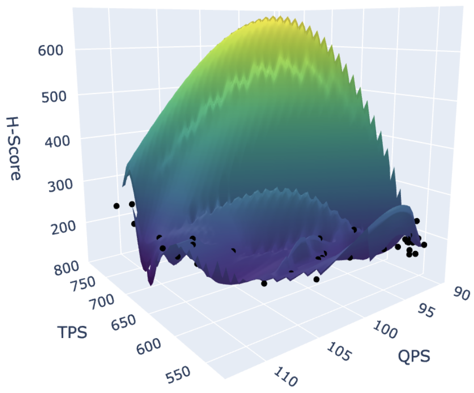

Figure 2. TPS vs QPS vs H-Score over all 1x Runs.

The composite H-Score further highlights the trade-offs inherent in unified HTAP processing. At 1x, H-Scores were higher but volatile, spanning 180 to 260, reflecting bursts of transactional and analytical contention (see the sharp ridges and valleys in Figure 2). Conversely, 10x runs yielded lower but much more stable H-Scores (about 18–28), indicating that increased data volume smooths performance variability through more effective MVCC snapshot isolation and adaptive resource scheduling (as shown by the more compressed surface in Figure 3).

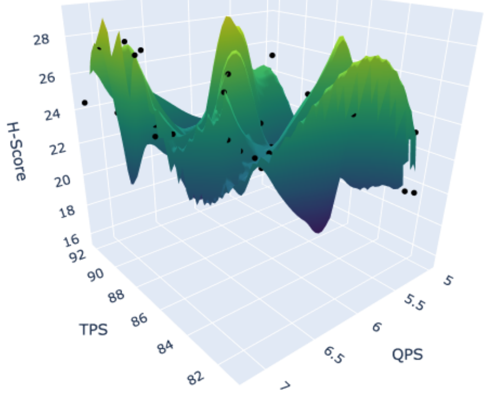

Figure 3. TPS vs QPS vs H-Score over all 10x Runs.

Although running OLTP and OLAP workloads independently on the same system can yield higher raw TPS or QPS in isolation, this approach still lacks the integration needed for real-time analytics and introduces delays when used in sequence. In contrast, the HTAP unified architecture eliminates these delays by supporting concurrent transactional and analytical workloads within a single engine.

Mixed-mode throughput in our HTAP configuration was approximately 3.5x lower than the sum of isolated workloads at SF 1x, and around 4x lower at SF 10x, due to shared resource contention under mixed loads. However, these throughput trade-offs are more than offset by the removal of data staleness, pipeline complexity, and maintenance overhead, resulting in a streamlined, low-latency platform ideally suited to modern, mission-critical analytics.

VI. Conclusion

The overall objective of our evaluation was to investigate the feasibility of employing an HTAP architecture capable of handling mixed workloads, while also delivering near real-time data insights within a fraud detection scenario. The technique to evaluate this feasibility was by using an HTAP benchmarking suite, while simultaneously building a harness in order to be able to orchestrate the entire process, keeping reproducibility and fairness across all runs.

Our runs show that, with proper configuration, HTAP systems can consistently uphold demanding Service Level Agreements by keeping sub-second data freshness while handling heavy mixed workloads at scale. The benchmarking methodology offers a practical guide for rolling out HTAP in live environments. However, one must take into consideration contention between long-running analytical queries and high-frequency transactions which led to latency spikes–highlighting the need for fine-grained concurrency control and adaptive resource scheduling. Early attempts using default database configurations significantly underperformed, reinforcing the necessity of tailored tuning and proactive monitoring. To capitalise on these advantages, data architects should start incorporating HTAP-aware access patterns and concurrency controls starting from the application design phase.

Future research could expand in several directions. Pushing tests to HyBench's higher scale factors would shed light on I/O behaviour, buffer management and concurrency under extreme loads and therefore better sizing of the instance. Exploring in-memory databases with persistent Non-Volatile RAM logging, as well as evaluating DBMS auto-tuning features for adaptive query optimisation, could further improve low-latency analytics. Standardising HyBench and adding support for varied data distributions, refining freshness metrics, and harmonising threading models is a critical need. Finally, extending the harness presented here for domain-specific microbenchmarks would create a unified framework for HTAP evaluation across sectors.

Looking ahead, the full adoption of HTAP architectures holds significant promise, but not without challenges. On the one hand, HTAP offers a path to simpler architectures, fresher data, and faster insights extraction, aligning closely with modern regulatory, operational, and customer expectations. On the other hand, widespread adoption will require rethinking application patterns, retraining engineering teams, and overcoming vendor lock-in as HTAP maturity varies across platforms. Moreover, while HTAP simplifies data pipelines, it shifts complexity into query optimisation, workload isolation, and configuration management; domains that still require advanced expertise and careful management. If these challenges can be addressed, HTAP could become a cornerstone for near real-time, data-driven decision-making in finance and beyond.

References

[1] O. O. Elumilade, I. A. Ogundeji, G. O. Achumie, and H. E. Omokhoa, "Enhancing Fraud Detection and forensic auditing through data-driven techniques for financial integrity and security," Journal of Advanced Education and Sciences, vol. 1, no. 2, pp. 55–63, 2021.

[2] D. Mangala and L. Soni, "A systematic literature review on frauds in banking sector," Journal of Financial Crime, vol. 30, no. 1, pp. 285–301, 2023.

[3] N. Faisal, J. Nahar, N. Sultana, and A. A. Mintoo, "Fraud Detection in Banking Leveraging AI to Identify and Prevent Fraudulent Activities in Real-Time," Journal of Machine Learning, Data Engineering and Data Science, vol. 1, no. 01, pp. 181–197, 2024.

[4] E. A. Minastireanu and G. Mesnita, "An Analysis of the Most Used Machine Learning Algorithms for Online Fraud Detection.," Informatica Economica, vol. 23, no. 1, 2019.

[5] A. Abdallah, M. A. Maarof, and A. Zainal, "Fraud detection system: A Survey," Journal of Network and Computer Applications, vol. 68, pp. 90–113, 2016.

[6] S. A. Ionescu, V. Diaconita, and A. O. Radu, "Engineering Sustainable Data Architectures for Modern Financial Institutions," Electronics, vol. 14, no. 8, p. 1650, 2025.

[7] Z. Zhang, A. Megargel, and L. Jiang, "Performance Evaluation of NewSQL Databases in a Distributed Architecture," IEEE Access, 2025.

[8] C. Zhang, G. Li, J. Zhang, X. Zhang, and J. Feng, "HTAP Databases: A Survey," IEEE Transactions on Knowledge and Data Engineering, 2024.

[9] W. Lehner, "Merging OLTP and OLAP–Back to the Future: (Panel)," in International Workshop on Business Intelligence for the Real-Time Enterprise, Springer, 2009, pp. 171–173.

[10] R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, and B. Becker, The data warehouse lifecycle toolkit. J Wiley, 2008.

[11] D. Beyer, S. Löwe, and P. Wendler, "Reliable benchmarking: Requirements and solutions," Int. Journal on Software Tools for Technology Transfer, vol. 21, no. 1, pp. 1–29, 2019.

[12] TPC-C Homepage — tpc.org, https://www.tpc.ord/tpcc/. [Retrieved: June, 2025]

[13] S. Leutenegger "A modeling study of the TPC-C benchmark," ACM Sigmod Record, vol. 22, no. 2, pp. 22–31, 1993.

[14] R. Cole et al., "The mixed workload CH-benCHmark," in Proceedings of the Fourth International Workshop on Testing Database Systems, 2011, pp. 1–6.

[15] C. Zhang, G. Li, and T. Lv, "HyBench: A New Benchmark for HTAP Databases," Proceedings of the VLDB Endowment, vol. 17, no. 5, pp. 939–951, 2024.

[16] C. Camilleri, C. Vella and V. Nezval, "HTAP with Reactive Streaming ETL," JCIT, vol. 23, no. 4, pp. 1-9, 2021.