NHtapDB: Native HTAP Databases

ABSTRACT

Hybrid **T**ransactional/**A**nalytical **P**rocessing (HTAP) database in a natural productive environment must leverage multimodal data to generate valuable real-time business insights, and execute OLAP queries in-between online transactions. State-of-the-art and state-of-the-practice systems adopt (1) a separate database and disparate business applications that leverage machine learning techniques to generate real-time business insights and (2) dual-format stores to guarantee the performance of different workloads—row-based storage for OLTP workloads and column-based storage for OLAP workloads. They fail to achieve the above goals because of massive data transfer overhead rooted in separate systems and dual-format stores. To this end, we propose NHtapDB, the first native HTAP database, providing business insight in real-time (within milliseconds to seconds). NHtapDB (1) provides a near-data machine learning framework to facilitate generating real-time business insight, and predefined change thresholds will trigger online training and deployment of new models, and (2) offers a mixed-format store to guarantee the performance of HTAP workloads, especially the hybrid workloads that consist of OLAP queries in-between online transactions. We make rigorous test plans for NHtapDB with an enhanced state-of-the-art HTAP benchmark.

PVLDB Artifact Availability:
NHtapDB is a vision paper, so it is not available for lacking of experiments.

1 INTRODUCTION

The goal of the traditional HTAP database (THtapDB) is to support Online Analytical Processing (OLAP) on the fresh data generated by the Online Transaction Processing (OLTP) [2, 5–7]. OLTP workloads generally read and write a small number of rows by index. OLAP workloads are read-intensive and involve complex queries on a few columns but numerous rows. Despite the abundance of THtapDB, they perform poorly in generating real-time business insights and guaranteeing the performance for HTAP workloads because of the vast data transfer overhead rooted in (1) a separate database, disparate business applications leveraging machine learning techniques, and (2) dual-format stores: row-based storage for OLTP workloads and column-based storage for OLAP workloads.

First, state-of-the-art or state-of-the-practice THtapDB workloads often run for a relatively long time (from minutes to hours),
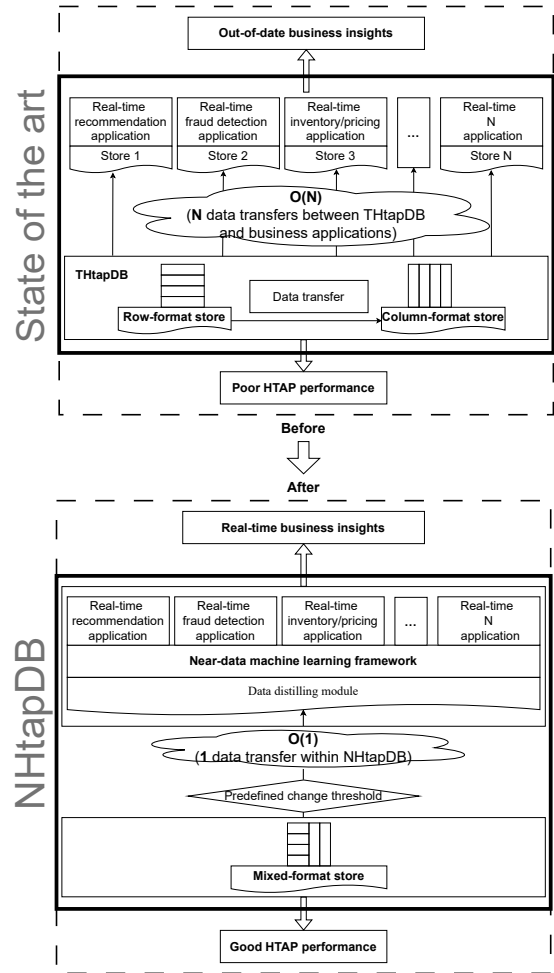
Figure 1: The motivation for the Native HTAP database: massive data transfer overhead results in the lack of capability of generating real-time business insights: the state-of-the-art systems (before) vs. NHtapDB (after).

incapable of providing real-time business insights and preventing it from satisfying the fleeting needs of customers. Consistent with the previous work [4], the implication of real-time emphasizes performing a task like data analysis or user behavior simulation interactively within milliseconds to seconds. THtapDB deployer focuses much of its effort on guaranteeing online transactions and complex query performance. They ignore the enormous benefits that real-time business data can bring to business applications [14, 17] and, as a result, fail to build the necessary infrastructure.

As shown in Figure 1, state-of-the-art architecture is separated into two parts: THtapDB and disparate business applications. Self-governed business applications leverage the data loader [12] to retrieve business data from the database, pre-process the business data into training samples, and then feed the training samples to

the business applications. As an independent OS process, each self-governed business application needs its own data loader instance to transfer data from databases. However, the long distance between THtapDB and business applications results in a long data turn-around time and O(N) data transfer, as shown in Figure 1, making it challenging to satisfy the fleeting needs of customers.

THtapDB stores a large amount of up-to-date data critical for business applications, but there is an urgent need to establish enabling facilities to generate real-time business insights from sharing business data. In E-commerce, many self-governed business applications, such as real-time recommendations, fraud detection, inventory/pricing, etc., need multiple real-time customer services based on the same business data. A typical scenario is real-time business insights in e-commerce that capture the real-time preferences of the customer to generate more accurate recommendations (predictions), which is a significant source of revenue for business applications. For example, in Amazon, the real-time recommendation can increase transactions by 35% [18]. If the recommendation is not real-time, the customer will likely leave the current session and purchase in another e-commerce application.

Besides, state-of-the-art THtapDB does not utilize multimodal data very well. Multimodal data refers to the different data types from different sources that may be complementary [13]. Multimodal data includes more details about customer behaviors that happened in business applications, such as customer clicks and customer reviews in e-commerce platforms, such as Taobao [16], so the fusion of these multimodal data could provide more accurate real-time business insights.

Second, the actual HTAP workloads consist of hybrid transactions that execute OLAP queries in-between online transactions [4, 8], rather than the separate OLTP and OLAP workloads. However, state-of-the-art or state-of-the-practice THtapDB adopts row-format, column-format, or dual-format stores that exaggerate the challenges in guaranteeing the HTAP workload performance.

For example, in e-commerce, customers prefer to purchase the best-selling commodity within their budget and other constraints, which is a typical HTAP workload. The related statements in the HTAP transaction are as follows.

```
1) SELECT MAX( ws_quantity )
   FROM web_sales
   WHERE ws_price between 64 and 64 + 16;

2) UPDATE customer SET c_balance = 1024
   WHERE c_id = 256;
```

The best-selling commodity selection calls the OLAP MAX function (ws_quantity) in structured query language (SQL), which is performed between online transactions — purchase operations involving banking and credit card activity (c_balance).

Row-based storage allows online transactions to locate the rows to update quickly. At the same time, column-based storage permits read-intensive OLAP queries to scan and aggregate operations over specific columns quickly. Nevertheless, a single row-based or column-based data organization is insufficient for HTAP workloads that have OLAP workloads in-between OLTP operations.

Using a dual-format store [3, 5] will introduce data transfer overhead between the row-based store and column-based store as shown in Figure 1. To keep data fresh, row-based data updates must be propagated to column-based storage as soon as possible. The previous work [4] revealed the peak HTAP performance using a dual-format store is lower than the peak OLTP performance by one order of magnitude. Consequently, a more suitable data organization for HTAP workloads is one of THtapDB's reform goals.

Our solution. As shown in Figure 1, to mitigate the heavy data transfer overhead O(N) between THtapDB and business applications, we built the native HTAP databases (NHtapDB), which not only makes a near-data machine learning infrastructure for acquiring real-time business insights but also handle HTAP workloads well. To provide real-time business insights, NHtapDB adopts the near-data machine learning framework that leverages predefined change thresholds to trigger online training to update the models in real-time. We implement data distilling instance that loads the real-time data, pre-processes it into training samples, and feeds training samples into models. And the near-data machine learning framework implements unified management for multiple models. When NHtapDB receives customer queries, the corresponding model must be updated until the customer leaves to satisfy their needs. The recommendation model must be updated in real-time to recommend the highly matched commodities lists if a customer searches for a commodity. As soon as a customer purchases a product, the real-time fraud detection model must be updated to identify fraudulent and anomalous activity associated with the transaction. NHtapDB implements a mixed-format store, which eliminates the inherent data transfer overhead between row-based and column-based stores that exist in the dual-format store. With the cooperation of the near-data machine learning framework, the mixed-format store, and other infrastructure, the data transfer overhead is reduced from N+1 times to 1 time. A simple upper-bound performance model in Section 2 shows the gap between the data transfer overhead of the two systems using near-data machine learning framework (NHtapDB) and separate machine learning systems (THtapDB) is 10,000 times in the case of 50 business applications.

Overall, the spotlights of NHtapDB can be summarized as follows:

(1) For real-time business insights, NHtapDB integrates a near-data machine learning framework to mitigate data transfer overhead between the database and the business applications. The machine learning engine abstracts three essential elements: state, action, and reward. To dynamically satisfy the customer's preferences, the machine learning engine uses the current state of the customer session as a new training sample to immediately update the model, then outputs the action, and finally calculates the reward to evaluate the action.

(2) NHtapDB provides a mixed-format store to handle the HTAP workloads with zero data transfer overhead. Mixed-format store splits the records into row-based update partitions and column-based read-only partitions. The row-based partitions are responsible for update operations, and the column-based partitions are leveraged for analytical queries. Mixed-format store eliminates the inherent data transfer overhead in dual-format store. Besides, the
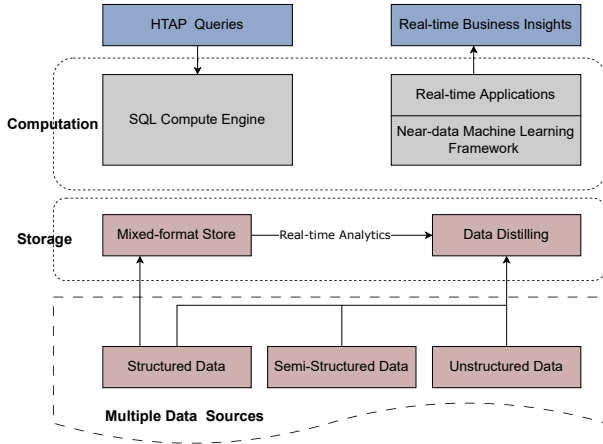
Figure 2: The overview of NHtapDB.

mixed-format store adopts the split write-ahead logging (WAL) [11] to guarantee insert and delete performance.

To demonstrate NHtapDB, we make rigorous demonstration plans. First, we will demonstrate the near-data machine learning framework provides real-time business insights. Second, we will demonstrate the mixed-format store can handle the HTAP workload well.

2 WHY DO WE NEED NHTAPDB?

To demonstrate the motivation, we build a simple upper-bound performance analysis model to evaluate the application (end-to-end) latencies under THtapDB and NHtapDB. We mainly consider the near-data machine learning framework's effect in NHTapDB. While under THtapDB, as an independent OS process, self-governed business applications leverage the data loader [12] to retrieve business data from the database and provide the service. This simple model fails to consider the effect of different stores, e.g., mixed-format stores under NHTapDB vs. dual-formal or single-store in THtapDB.

The application's latency includes two components: the data transfer latency and the computation latency. Data transfer latency is the key to significantly reducing overall latency. Consequently, we compare the data transfer latency between THtapDB and NHtapDB. We assume there are 50 business applications (N=50), and each business application needs to deal with the 1GB data analysis. For the THtapDB solution, the total transfer bandwidth is 500 MB/Second (using the state-of-art NFS solution for data transfer), and that of NHtapDB is 100GB/Second (using the memory space access on the same OS process). The data transfer latency of the THtapDB solution is 100 seconds (the transfer bandwidth for each application is 10MB/S, and the transfer latency of each application is 100 seconds). At the same time, that of NHtapDB is only 0.01 seconds (the transfer bandwidth is constant at 100GB/Second, and the transfer latency of each application is 0.01 seconds). The gap between the data transfer overhead of the two systems is 10,000 times.

3 NHTAPDB OVERVIEW

NHtapDB is a powerful HTAP database that ensures HTAP performance and provides real-time business insights based on multimodal data fusion. In this section, we sketch one possible architecture for NHtapDB, depicted in Figure 2. NHtapDB is composed of two main layers: the computation layer and the storage layer.

3.1 Computation layer

An efficient computation layer is supported by two modules:

(1) Near-data machine learning engine aims at accelerating business data into insights in real-time to satisfy the fleeting needs of customers. It defines and implements three essential elements to support online training – state, action, and reward. ML engine first receives the current state of the customer, then performs the suitable action, and finally calculates the reward to evaluate the action. ML engine needs to reuse other state-of-the-art techniques, such as feature engineering.

(2) The SQL compute engine is stateless, scalable, and aware of storage. It chooses the cheapest logical plan according to estimated execution costs and then transforms the logical plan into a physical execution plan based on the storage layout.

3.2 Storage layer

The storage layer comprises a mixed-format store and a data-distilling module.

Mixed-format store guarantees HTAP performance by eliminating the data update propagation delays inherent in dual-format stores. In dual-format stores, the OLAP delays have two sources, one is the data update propagation delays from row-based store to column-based store, and the other is the OLAP execution delays. Mixed-format store splits row records into row-based update partitions and column-based non-update partitions so that online transactions and analytical queries execute in parallel. To improve SQL performance, a mixed-format store needs cooperation with other techniques, such as caching, indexes, and SQL computation engines.

The data distilling module adopts a novel lightweight multimodal data fusion technique to acquire real-time business insights. When the predefined change thresholds are triggered, it will receive structured business data from the database. In addition, it receives multimodal business data from multiple data sources, such as stream systems. Following data cleansing, feature extraction, and other processes, the above business data is formed into training samples, which are then fed into the computation layer for online learning.

4 NHTAPDB DESIGN

In this section, we introduce two key design aspects of NHtapDB depicted in Figure 4. We describe the use case of near-data machine learning framework and mixed-format store in detail below.

4.1 Near-data machine learning framework

Near-data machine learning framework defines and implements three essential elements on top of heterogeneous models for real-time business insights. This subsection explains the three elements and the machine learning model instance as shown in Figure 3.
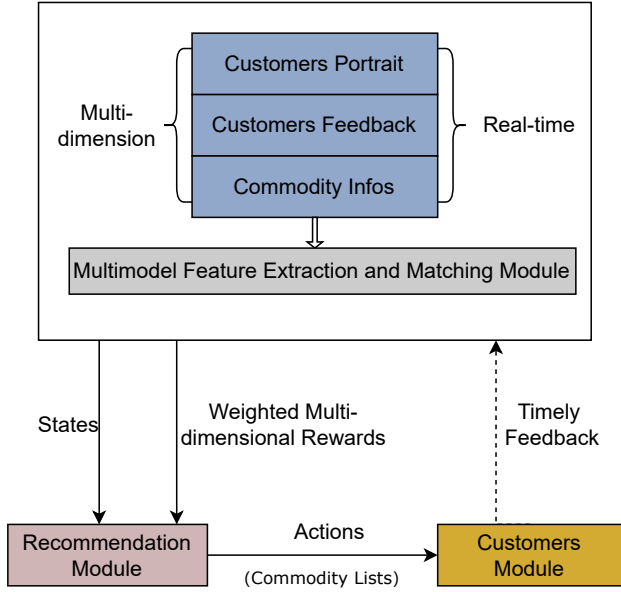
Figure 3: Machine learning model instance — real-time recommendation model.

4.1.1 Essential elements. Near-data machine learning framework defines and implements three elements: state S, action A, and reward function R.

State S is the set of all possible states, S^t represents the state at time step t.

Action A is the set of available actions depend on states, A^t is the action taken at time step t.

Rewards R provides a reward to assess the actions selected.

4.1.2 Machine learning model instance. We give the instance of a real-time recommendation model, which reuses the three essential

Table 1: Partial features of multi-dimension data (p_1–p_2: customer portrait features; c_1–c_5: click feedback features; q_1–q_2: text/image query feedback features; r_1–r_2: additional labels feedback features; i_1–i_3: commodity information features).

Variate	Features	Description	Description
p_1	time	the time of day	string
p_2	location	the location of customer	string
c_1	pv	commodity page view	bool
c_2	buy	buy commodity	bool
c_3	cart	add commodity to shopping cart	bool
c_4	favorite	favorite commodity	bool
c_5	duration	commodity page view duration	string
q_1	text query	query in natural language	string
q_2	image query	query in image	string
r_1	price	real-time price range	float
r_2	inventory	real-time inventory quantity	int
i_1	category	commodity category	one-hot
i_2	subcategory	commodity subcategory	one-hot
i_3	style	commodity style	string

elements and defines recommendation, customer, and multimodal feature extraction modules, as explained below.

At the time step t, the recommendation module precepts the current state S^t of the customers and recommends the suitable commodity list to the customer. Next, it receives weighted multi-dimensional rewards R^t representing the quality of the recommendation, and then it receives the new state S^{t+1} at time step $t+1$.

Customers module. Customers send timely feedback and receive the recommended commodity list. Timely feedback reveals interactions between customer and commodity in the current state, including click feedback and search feedback. Click feedback implied real-time customer preference information, which contains commodity page view (pv), adding commodity to shopping cart (cart), buying the commodity (buy), favorite commodity (favorite), and commodity browsing duration (duration). And search feedback refers to the real-time text/image query and additional labels, directly indicating the customer's needs. For example, in Taobao [16], customers could use text or images to search for the commodity. They could also use additional labels to filter targeted commodities quickly, and common labels include real-time inventory quantity and price range. Above feedback is sent to the multimodal feature extraction and matching module for further processing.

Multimodal feature extraction and matching module It collects multi-dimension data for feature extraction and matching, which includes customer portraits, customer feedback, and commodity information. And the feature is used for the current state S^t representation. Features of the above data are depicted in table 1. Customer portrait features refer to the time of day, location, customer scenario, etc. Given a customer i, the corresponding portrait is mapped to feature p_i^t. Customer feedback is detailedly described in the customer module. The click feedback feature is denoted as c_i^t. Text query feature is denoted as q_{text}^t. And the image query feature is denoted as q_{image}^t. Real-time analytics generates an additional label that is denoted as r_i^t. Commodity information collects the basic attributes of a commodity, such as a commodity category, commodity subcategory, and commodity style, which is denoted as i_i^t.

Next, the current state S^t is delivered to the recommendation module.

Recommendation module. It receives the current state S^t and recommends the commodity list to the customer. Later, it receives weighted multi-dimensional rewards R^t, which evaluate the quality of the recommendation. Multi-dimensional rewards have six parts: customer portrait reward R_p^t, click feedback reward R_c^t, text query reward R_{text}^t, image query reward R_{image}^t, additional label reward R_r^t, and commodity information reward R_i^t. And the final reward R^t at the time step t is:

$$R^t = \beta + \lambda_1 R_p^t + \lambda_2 R_c^t + \lambda_3 R_{text}^t + \lambda_4 R_{image}^t + \lambda_5 R_r^t + \lambda_6 R_i^t \quad (1)$$

In turn, the recommendation module proceeds with the next round of interaction with the customer module at the time step $t + 1$.

4.2 Mixed-format Store

NHtapDB adopts a mixed-format store to guarantee the performance of HTAP workloads. Mixed-format store architecture is
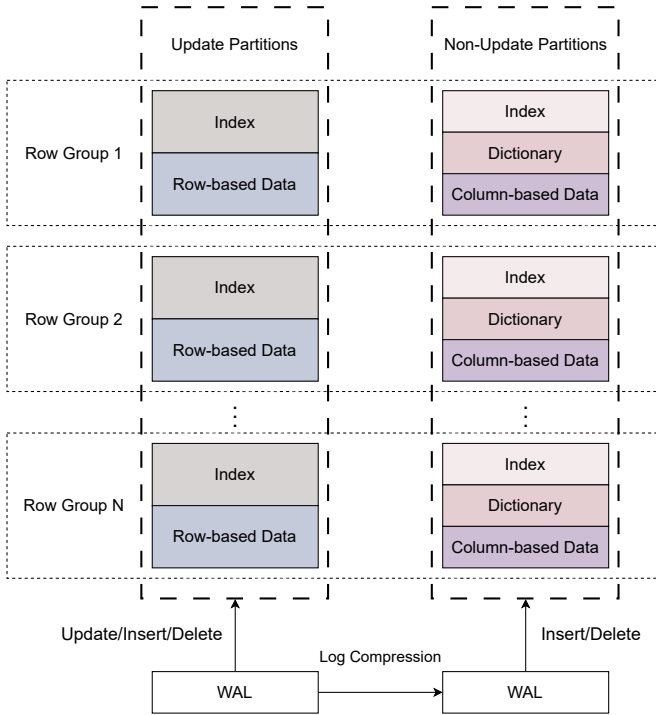
Figure 4: Mixed-format store architecture.

shown in Figure 4. Mixed-format store adopts the range partition strategy to divide all row records into *row groups* evenly to leverage modern computers' multi-core parallelism. The *row groups* splits row records into *update partitions* and *non-update partitions* based on whether their column attributes are updated. The *update partitions* are placed in row-based format for OLTP performance, and the *non-update* partitions are placed in column-based format for OLAP performance. For example, the CUSTOMER table in the TPC-C benchmark places the *C_ID*, *C_BALANCE*, and *C_DATA* attributes into the row-based store, and places the other attributes into the column-based store. At the same time, the mixed-format store avoids the data update propagating overhead between row-based and column-based data. The mixed-format store must cooperate with state-of-the-art indexes and dictionary techniques [1, 10, 15] to improve SQL performance.

NHtapDB adopts split write-ahead logging (WAL) [11] for atomicity and durability. WAL sequentially records the transactions' prewritten, commit, and rollback behaviors into the persistent devices. There are three kinds of log items – update log items, insert log items and delete log items. The data in update partitions are updated as update log items. The original insert and delete log item is split into a row log item and a column log item. Only when the row log item is committed the corresponding column data are inserted or deleted as the column log item. We adopt a log compression strategy to ease the unnecessary insert and delete pressure of column-based data. Log compression strategy deletes column log items whose row log entries are rollback. The original log item will not be committed until both the row and column log items have been committed.

5 DEMONSTRATION

NHtapDB connects the underlying storage and the upper machine learning models, shortens the data turnaround time, and uses real-time analytics to help provide real-time business insights.

Our demonstration plan aims to show that.

a) **Near-data machine learning framework** provides real-time business insights with low data transfer overhead.

b) **Mixed-format store** guarantees HTAP performance.

For the first objective, we conduct experiments on the multi-modal dataset [9]. Near-data machine learning framework has one test case.

1) *Test case* 1 : Evaluating the data transfer overhead between database and business applications.

For the second objective, we use the OLxPBench suite [4], which is the first benchmark addressing the necessity of introducing HTAP workloads that execute OLAP queries in-between online transactions [4, 8] and providing multiple HTAP workloads. We ran the OLxPBench in different configurations by varying the type and sending rate of workload. In the demonstration, we compare NHtapDB with state-of-the-practice HTAP databases – TiDB [3].

2) *Test case* 2 : Comparing the HTAP performance of TiDB and NHtapDB.

REFERENCES

[1] Martin Boissier, Rainer Schlosser, and Matthias Uflacker. 2018. Hybrid data layouts for tiered HTAP databases with pareto-optimal data placements. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 209–220.

[2] Amirali Boroumand, Saugata Ghose, Geraldo F. Oliveira, and Onur Mutlu. 2022. Polynesia: Enabling High-Performance and Energy-Efficient Hybrid Transactional/Analytical Databases with Hardware/Software Co-Design. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 2997–3011. https://doi.org/10.1109/ICDE53745.2022.00270

[3] Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, Li Shen, Liu Tang, Yuxing Zhou, Menglong Huang, et al. 2020. TiDB: a Raft-based HTAP database. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3072–3084.

[4] Guoxin Kang, Lei Wang, Wanling Gao, Fei Tang, and Jianfeng Zhan. 2022. OLxPBench: Real-time, Semantically Consistent, and Domain-specific are Essential in Benchmarking, Designing, and Implementing HTAP Systems. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 1822–1834. https://doi.org/10.1109/ICDE53745.2022.00182

[5] Tirthankar Lahiri, Shasank Chavan, Maria Colgan, Dinesh Das, Amit Ganesh, Mike Gleeson, Sanket Hase, Allison Holloway, Jesse Kamp, Teck-Hua Lee, et al. 2015. Oracle database in-memory: A dual format in-memory database. In *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 1253–1258.

[6] Per-Åke Larson, Adrian Birka, Eric N Hanson, Weiyun Huang, Michal Nowakiewicz, and Vassilis Papadimos. 2015. Real-time analytical processing with SQL server. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1740–1751.

[7] Juchang Lee, SeungHyun Moon, Kyu Hwan Kim, Deok Hoe Kim, Sang Kyun Cha, and Wook-Shin Han. 2017. Parallel replication across formats in SAP HANA for scaling out mixed OLTP/OLAP workloads. *Proceedings of the VLDB Endowment* 10, 12 (2017), 1598–1609.

[8] Guoliang Li and Chao Zhang. 2022. HTAP Databases: What is New and What is Next. In *Proceedings of the 2022 International Conference on Management of Data*. 2483–2488.

[9] Junyang Lin, An Yang, Yichang Zhang, Jie Liu, Jingren Zhou, and Hongxia Yang. 2020. InterBERT: Vision-and- Language Interaction for Multi-modal Pretraining. *CoRR* abs/2003.13198 (2020).

[10] Chen Luo, Pinar Tözün, Yuanyuan Tian, Ronald Barber, Vijayshankar Raman, and Richard Sidle. 2019. Umzi: Unified multi-zone indexing for large-scale HTAP. In *Advances in Database Technology-22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*. OpenProceedings. org, 1–12.

[11] Chandrasekaran Mohan, Don Haderle, Bruce Lindsay, Hamid Pirahesh, and Peter Schwarz. 1992. ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Transactions on Database Systems (TODS)* 17, 1 (1992), 94–162.

[12] Iason Ofeidis, Diego Kiedanski, and Leandros Tassiulas. 2022. An Overview of the Data-Loader Landscape: Comparative Performance Analysis. *arXiv preprint arXiv:2209.13705* (2022).

[13] Sharon Oviatt, Björn Schuller, Philip R Cohen, Daniel Sonntag, Gerasimos Potamianos, and Antonio Krüger. 2018. *The Handbook of Multimodal-Multisensor Interfaces: Signal Processing, Architectures, and Detection of Emotion and Cognition-Volume 2.* Association for Computing Machinery and Morgan & Claypool.

[14] Bruno L Pereira, Alberto Ueda, Gustavo Penha, Rodrygo LT Santos, and Nivio Ziviani. 2019. Online learning to rank for sequential music recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems.* 237–245.

[15] Christian Riegger, Tobias Vinçon, Robert Gottstein, and Ilia Petrov. 2019. MV-PBT: multi-version index for large datasets and HTAP workloads.

[16] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. 2019. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4902–4909.

[17] Anuruddha Thennakoon, Chee Bhagyani, Sasitha Premadasa, Shalitha Mihiranga, and Nuwan Kuruwitaarachchi. 2019. Real-time credit card fraud detection using machine learning. In *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 488–493.

[18] Bo Zhou and Tianxin Zou. 2022. Competing for recommendations: The strategic impact of personalized product recommendations in online marketplaces. *Marketing Science* (2022).