

A Statistical-ML Hybrid Approach for Robust Selectivity Estimation on Skewed Data

by

Unaspekov Timur

Submitted to the Department of Data Science
in partial fulfillment of the requirements for the degree of

Master of Science in Data Science

at the

NAZARBAYEV UNIVERSITY

June 2026

© Nazarbayev University 2026. All rights reserved.

Author
Department of Data Science
4 May 2026

Certified by
Yesdaulet Izenov
Instructor
Thesis Supervisor

Accepted by
Michael Lewis
Chair, Data Science Department

A Statistical-ML Hybrid Approach for Robust Selectivity Estimation on Skewed Data

by

Unaspekov Timur

Submitted to the Department of Data Science
on 4 May 2026, in partial fulfillment of the
requirements for the degree of
Master of Science in Data Science

Abstract

Selectivity estimation is a major challenge in modern database management systems and can be considered as one of the bottlenecks in query optimization. Traditional histogram-based summaries often lead to estimation errors when applied on skewed data. Newly appeared machine learning algorithms can improve the accuracy but their usage leads to the higher inference latency and training costs, which limits their capability to be useful in high loaded systems.

We propose Hybrid FD CDF framework for high precision, single attribute, range selectivity calculation and managing the estimator adaptability for new data and workload. This framework uses the divide-and-conquer strategy, partitioning the data domain using Freedman Diaconis Rule to calculate optimal number of buckets with fixed width. Then, using a waterfall model selection policy select a model that takes the least resources to reach high accuracy, leading to optimal resource management and optimization.

Experimental results on synthetically generated datasets with hard distribution and on real benchmarks show that our approach achieves lower Q-error and inference time compared to the Equi-Width approach.

Thesis Supervisor: Yesdaulet Izenov
Title: Instructor

Acknowledgments

Contents

1	Introduction	15
1.1	Background and Motivation	15
1.2	Problem Statement	16
1.3	Proposed Solution: Hybrid FD CDF Estimator	16
2	Related works	19
2.1	The Evolution of Selectivity Estimation	19
2.2	Classical Data-Driven Summaries	19
2.3	The Curse of Dimensionality	21
2.4	Kernel Density Estimation (KDE) and Sampling	22
2.5	Query-Driven and Feedback-Based Adaptivity	22
2.6	The Neural Cardinality Revolution	23
2.7	Specialized Estimators for String, Spatial, and Temporal Data	24
2.8	The Spectral Bias of Neural Networks	25
2.9	Latency and Resource Bottlenecks of ML-Inference	26
2.9.1	The millisecond threshold	26
2.9.2	The SIMD vs. GPU Trade-off	26
2.10	Modern Hybrid and Lightweight Approaches	27
2.10.1	Piecewise Linear/Mixture Models	27
2.10.2	Updatable Frameworks/Data Drift	28
2.11	Sample reservoir	28
2.12	Batch Processing in machine learning and deep learning	29
2.13	Research Gap: Partitioning to handle Skew	29

3	Proposed System: Hybrid FD CDF Estimator	31
3.1	Initial domain learning and buckets setup	32
3.1.1	Freedman-Diaconis (FD) Rule	32
3.1.2	Memory-Constrained Saturation and the $k = 2,000$ Cap . . .	34
3.2	Local Training Set Construction	34
3.2.1	Coordinate Normalization to Unit-Bounded Space	35
3.2.2	Complex Sampling Strategy	35
3.2.3	Density-Based Importance Weighting	35
3.3	The Waterfall Model Selection Policy	36
3.4	High-Throughput Vectorized Inference Engine	38
3.4.1	Processing in batches	38
3.4.2	Processing in queue	39
3.5	Dynamic Maintenance and Feedback Cycles	39
3.5.1	Finetune process	39
3.5.2	Rebuild process	40
3.6	Theoretical Analysis	41
3.6.1	Complexity Analysis	41
3.6.2	Error Bounds	41
3.6.3	Convergence Behavior of Dynamic Maintenance	42
3.7	Hybrid Inference and Execution Walkthrough	42
4	Experiments results	45
4.1	Datasets Generation and Management	45
4.1.1	Dataset Manager Framework	45
4.1.2	True Fractal Generation Logic	46
4.1.3	Scientific Determinism and Reproducibility	49
4.2	Static Benchmark Distributions	49
4.2.1	Baseline Distributions	50
4.2.2	DSB Benchmark	51
4.3	Drift modeling	51

4.3.1	Drift types	52
4.3.2	Linear transition and resolutions	52
4.3.3	Sample reservoir maintenance simulation	53
4.3.4	DSB Benchmark	53
4.4	Workload and Evaluation Metrics	53
4.4.1	Initial Strategy: Uniform Random Workload	53
4.4.2	Advanced Strategy: Data-Driven Workload Generation	55
4.4.3	Workload Persistence and Reproducibility	58
4.4.4	Workload for drift benchmark	58
4.5	Evaluation Metrics	58
4.5.1	Operational Efficiency Metrics	58
4.5.2	Accuracy Metrics	59
4.5.3	Drift and Adaptivity Metrics	59
4.6	Baselines for Comparison	60
4.6.1	Equi-Width Histogram	60
4.6.2	Equi-Hist (Adaptive Feedback Baseline)	61
4.7	Experimental results	61
4.7.1	Static setup batch inference	61
4.7.2	Static setup query one-by-one inference	67
4.7.3	Dynamic setup batch inference	70
4.7.4	Dynamic setup query one-by-one inference	91

5 Conclusion 103

List of Figures

2-1	Equi-Width Inference example	20
3-1	Hybrid build and maintenance stages	32
4-1	Comparison of inner micro-bucket distributions for complex non-linear gradients (Part I).	47
4-1	Comparison of inner micro-bucket distributions for statistically simpler or sparse distributions (Part II).	48
4-2	Visualization of 5 synthetic data distributions ($N = 6 \times 10^7$).	51
4-3	Visualization of narrow range queries statistics.	54
4-4	Visualization of wide range queries statistics.	55
4-5	Analysis of the Data-Driven workload (Part I: Uniform, Normal, and Zipfian).	57
4-5	Analysis of the Data-Driven workload (Part II: Sparse Cluster and Anti-Zipf).	57
4-6	10% step, normal -> zipf, workload normal -> normal	72
4-7	5% step, normal -> zipf, workload normal -> normal	75
4-8	5% step, normal -> zipf, workload normal -> zipf	78
4-9	5% step, normal -> normal + zipf, workload normal -> zipf	81
4-10	5% step, DSB dataset ws_item_sk	85
4-11	5% step, DSB dataset cr_item_sk	88
4-12	5% step, DSB dataset cr_returned_time_sk	91

List of Tables

4.1	Calculated Statistical Moments and FD Buckets Parameters ($N = 6 \times 10^7$)	49
4.2	Benchmark Results (Dataset: SIMPLE, Workload: WIDE). Error metrics are reported as Symmetric Q-Error.	62
4.3	Benchmark Results (Dataset: SIMPLE, Workload: NARROW). Error metrics are reported as Symmetric Q-Error.	63
4.4	Benchmark Results (Dataset: HARD, Workload: WIDE). Error metrics are reported as Symmetric Q-Error.	64
4.5	Benchmark Results (Dataset: HARD, Workload: NARROW uniform). The best Median Q-Error for each distribution is highlighted in bold.	65
4.6	Benchmark Results (Dataset: HARD, Workload: NARROW data driven). Best Median Q-Error per distribution is highlighted in bold.	66
4.7	Benchmark Results: Decision Support Benchmark (DSB)	67
4.8	Benchmark Results (Dataset: SIMPLE, Workload: WIDE, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.	67
4.9	Benchmark Results (Dataset: SIMPLE, Workload: NARROW, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.	68
4.10	Benchmark Results (Dataset: HARD, Workload: WIDE, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.	68

4.11	Benchmark Results (Dataset: HARD, Workload: NARROW, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold. . .	69
4.12	Benchmark Results (Dataset: HARD, Workload: NARROW data driven, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.	69
4.13	Benchmark Results (Dataset: DSB Benchmark, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.	70
4.14	10% step, normal $\rightarrow$ zipf, workload normal $\rightarrow$ normal	70
4.15	5% step, normal $\rightarrow$ zipf, workload normal $\rightarrow$ normal	73
4.16	5% step, normal $\rightarrow$ zipf, workload normal $\rightarrow$ zipf	76
4.17	5% step, normal $\rightarrow$ normal + zipf, workload normal $\rightarrow$ zipf	79
4.18	Detailed Drift Results: ws_item_sk (5% Step, Displacement Drift) . .	82
4.19	Detailed Drift Results: cr_item_sk (5% Step, Displacement Drift) . .	85
4.20	Detailed Drift Results: cr_returned_time_sk (5% Step, Displacement Drift)	88
4.21	Detailed Drift Results: Normal $\rightarrow$ Zipfian (10% Step, Displacement Drift, Normal Workload query one-by-one inference). Best Median Q-Error per distribution is highlighted in bold.	91
4.22	Detailed Drift Results: Normal $\rightarrow$ Zipfian (5% Step, Displacement Drift, Normal Workload, One-by-One Inference). Best Median Q-Error per distribution is highlighted in bold.	93
4.23	Detailed Drift Results: Normal $\rightarrow$ Normal + Zipfian (5% Step, Additive Drift, Workload Normal $\rightarrow$ Zipfian, One-by-One Inference). Best Median Q-Error per distribution is highlighted in bold.	95
4.24	Detailed Drift Results: Normal $\rightarrow$ Normal + Zipfian (5% Step, Additive Drift, Workload Normal $\rightarrow$ Zipfian, One-by-One Inference). Best Median Q-Error per distribution is highlighted in bold.	98

Chapter 1

Introduction

1.1 Background and Motivation

Selectivity estimation plays an important role in calculating the cost of a query execution plan [35]. The problem is that when having a query that consists of high number of joins and predicates, a small error in an initial cardinality estimation leads to a high accuracy error at the end [16, 22]. Such plans consume a significant amount of resources, because initially the plan provided for the consumption of a small amount of resources, but in reality a large amount of data is loaded into the system.

Commercial databases use Equi-Width and Equi-Hist algorithms to calculate selectivity, because their inference time complexity $O(1)$ and $O(\log N)$ respectively, while able to calculate selectivity in a fixed resource budget [28, 31, 15]. However, there is a uniform assumption in the algorithm, due to which the algorithms predict an incorrect selectivity value for the skewed data. This leads to the need for algorithms that operate with high accuracy under fixed memory and time budgets [8, 29].

Neural networks and learning estimators address this problem by treating it as regression problem [21, 43]. While other algorithms are achieving lower errors, but they take 10-100 times more time to inference [42, 19].

1.2 Problem Statement

We observe that approaches are divided into two groups, first which give selectivity with error that leads to the performance issues [22] and second which calculate selectivity taking much more time and it can be challenging to deploy these methods in the databases that work under heavy load [9, 47].

Also, it is known that the data changes over time so we need algorithm that would be adaptable to the newly arrived data distributions maintaining high selectivity accuracy without being rebuilt from scratch[44, 41].

To study the limitations of static histograms and neural networks in query optimizer selectivity estimation problem this thesis investigates the following research questions:

1. How can statistical methods like Freedman-Diaconis rule be used to dynamically partition data domains and minimize the intra-bucket uniformity assumption?
2. To what extent can a waterfall model-selection policy balance the trade off between the inference latency and accuracy of estimation?
3. How can the estimator adapt to the data drift using the reservoir sampling without the full retraining and maintaining query optimization latency budgets?

1.3 Proposed Solution: Hybrid FD CDF Estimator

The problem is that when using a fixed number of buckets and their width, we do not take into account the distribution of data inside the bucket. We integrate into our solution the use of Freedman Deaconis rule [10] that determine the optimal number of buckets and their width, making the number of buckets an initially non-hardcoded value. The number of buckets and their width are now statistically supported.

Also in our research, we noticed that the distribution of data inside a bucket can be varied from uniform to highly skewed and thus hard to fit distribution. This requires an adaptive approach to each bucket. In our approach, we use statistical data to determine the skew inside the bucket and, if not necessary, use cheaper algorithms to

calculate selectivity, and in the case of complex distribution inside the bucket, use a more complex algorithm up to neural network.

For inference we use two mechanisms, the first concept is processing queries and their predicates in queue one by one, and second concept is processing predicates in batches. We run the experiments in two setups and study the effect of processing the predicates in batches and it's possible deployment and feasibility in real database systems.

To solve the problem with updating the optimizer as data changes in the system, we present an algorithm that helps to maintain high accuracy in calculating selectivity, taking into account the past workload and its true selectivity [45, 48] and the number of changes in the data itself. This approach allows not to train the estimator after each query execution, which can lead to a large inference overhead, but it allows to train only when the performance degrades [44, 30], taking into account changes in the data and how much the real selectivity differs from the predicted one. For optimization purposes, we divide this process into two stages, tuning and rebuilding stages. Also, to optimize the training process we utilize the sample reservoir that database keeps up to date, the sample reservoir is the high representation of the real data and we can utilize that to work on less amount of data to make operations like sorting much more feasible and computationally fixed.

We run experiments on synthetic datasets that are generated that way so that distribution inside the buckets is complex and not uniform, this is done to stress test the algorithms and to check the overhead that Hybrid FD CDF model introduces over the Equi-Width approach. Also, we run experiments on DSB Benchmark, and results show that Hybrid FD CDF manages to achieve better results while maintaining slightly higher inference time. Second setup of experiments is conducted when the data is not static and constantly changes, these experiments show that Hybrid FD CDF approach introduced inference overhead because of the rebuild stage for about 40%, but it maintains the Q-error at low rates. That way we show that our approach that is built on top of the Equi-Width utilizes it's advantages, and deals with the uniform assumption by dividing the data domain into buckets and selecting the most

suitable and least computationally expensive machine learning or deep learning model to the bucket.

Chapter 2

Related works

2.1 The Evolution of Selectivity Estimation

Selectivity estimation involves estimating $s = \frac{|\sigma_P(T)|}{N}$ for a predicate P on a table T . Early engines made the uniformity and independence assumptions because they provide an extremely cheap way to estimate a joint probability distribution without the cost of a physical block scan at planning time[35]. These assumptions are not true usually [22]. The problem here is that assuming every attribute to be an independent, the query optimizer does not consider the nature of real world data. This means that there exists correlation between the variables. This was a necessary simplification when there were limits on the CPU cycles available. Using this approaches, the errors compound exponentially in the plan tree [16]. We find that the optimizer ends up working with a hypothetical database that has a little understanding to the real one, and join orders are mathematically incoherent with the real data skew [22].

2.2 Classical Data-Driven Summaries

Histograms are a step up from simple heuristics, and how it partitions the domain of values [15]. Equi-width data partitions are computationally cheap. Equi-depth histograms, like the ones used in PostgreSQL partition data, so that the number of tuples per bucket is approximately the same [28, 31]. But the approach remains just

as susceptible to the intra-bucket uniformity assumption. The issue is that as long as there is a heavy hitter within a bucket and the skew, the estimator will assume a smoothly distributed data within bucket, where in reality there is an isolated spike of skewness. The V-optimal histograms partitions the data and minimize the intra-bucket variance [31]. But the $O(N^2)$ cost to construct them makes the best models computationally infeasible for any workload that requires constant rebuild as the data changes.

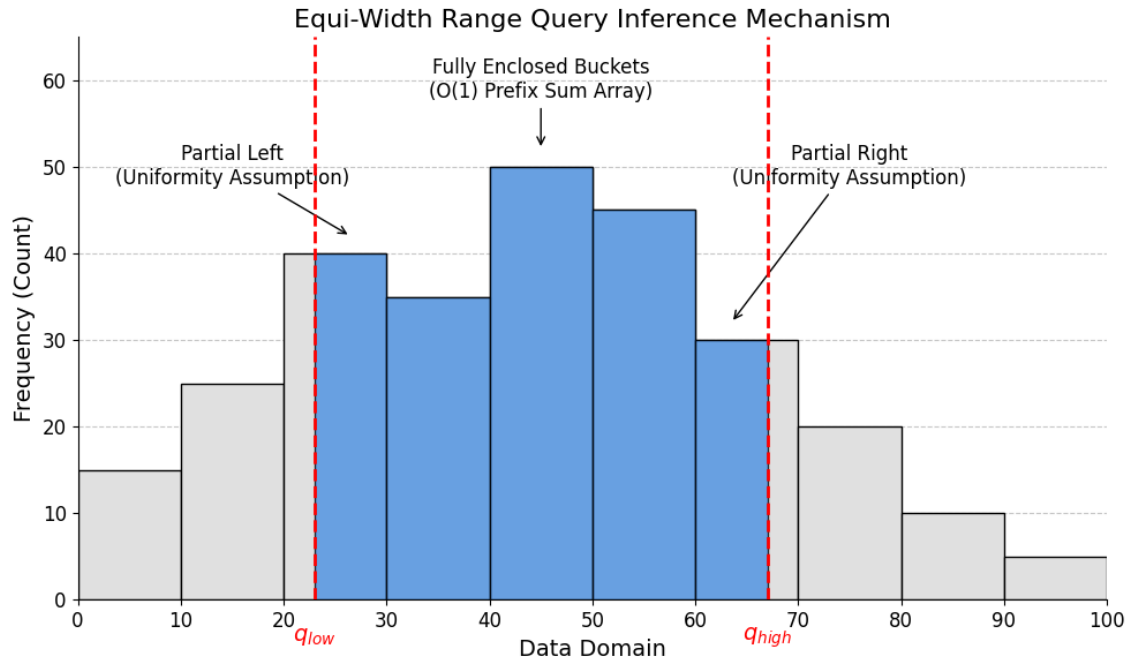


Figure 2-1: Equi-Width Inference example

The Equi-Width histogram achieves high efficiency inference by using the uniform bucket width, which allows for $O(1)$ mapping of the query predicates to the bucket indices. Given a query range, the Equi-Width calculates the start and end bucket indices using a division relative to the domain minimum and fixed bucket width. For edge buckets the uniformity assumption is applied, calculating the overlap fraction which is linearly proportion to the bucket width. For the intermediate buckets that are enclosed by the query, the precomputer prefix sum array of bucket counts is used, which results the sum to be retrieved in $O(1)$ time.

2.3 The Curse of Dimensionality

Capturing selectivity on correlated predicates is another breakdown case that has not improved on wide-tables [7][14]. The issue here independence assumption is not true and that very few queries filter on orthogonal, random dimensions. This 1D probability calculation is cheap, but, the product of these probabilities does not capture the correlations. The estimation error compounds itself for each additional predicate in the estimator. The result is a plan that underestimates the cardinality by several orders of magnitude because the estimator was evaluated on a series of uncorrelated assumptions.

Grid partitioning attempts to deal with those correlations by breaking the n -dimensional space into axis-aligned buckets [6]. This is good enough when the data is low-cardinality. But, the number of buckets grows exponentially in the number of dimensions. We see that, high-dimensional grids end up with mostly empty space or counts so small that are statistically irrelevant and wastes memory [2]. Tree partitioning, using structures like R-trees or Quad-trees, is more flexible in the sense that bucket boundaries are rotated to fit data clusters. The issue here is the update cost. Maintaining those structures on high-load workloads is prohibitively expensive [11]. Also, the overlap region between two buckets makes it hard to calculate the selectivity of range predicates [39].

The underlying issue in all these is the inability of these structures to capture the relationship between highly correlated attributes. We see that correlated predicates that fall inside the same high-density region in the table end up with a misprediction. This is true in the same sense as these data is either perfectly independent or perfectly grid-aligned. We can model this using Fourier-features or local regressions. But, traditional models always predict either zero or a very large overestimate because they cannot interpolate the empty voids of high-dimensional space.

2.4 Kernel Density Estimation (KDE) and Sampling

KDE functions replace fixed buckets with parameterized kernel functions that create a smooth, differentiable surface attempting to model the underlying distribution [13]. We find this approach is sufficient to model multi-modal distributions. This is contingent on the summation of the local approximations forming the global state. But, the cost of these evaluations in the planning process makes it theoretical [13].

KDEs are incapable of dealing with the extreme skew of modern analytical workloads because of the bandwidth selection problem. The difficulty lies in the small smoothing parameter needed for heavy-hitters leading to overfitting, and the large kernels counting records in empty spaces while ignoring sharp data peaks. The larger kernels generate a smoother surface. But, this oversmoothing leads to large relative errors in high-selectivity regions.

Sampling avoids density modeling by examining a fixed table subset in the query optimization process [25]. This naturally captures attribute correlations. But, the approach does not work on high-selectivity predicates [12]. Sample sizes achieving accuracy for small result sets often approach in cost the execution of the original query. Sampling is broken by skew since the chance of sampling sparse, relevant records is too low to provide statistical stability.

2.5 Query-Driven and Feedback-Based Adaptivity

Self-tuning optimizers such as LEO go beyond static synopses by maintaining a feedback loop to learn the cardinality of query plans that were actually executed [36]. We find this shift to Query Feedback Records essential to react to the failure of independence assumptions without having to resort to constant full-table scans [3, 4]. This assumes that the optimizer has already been through the initial query execution. The challenge here is the cold-start problem. But such systems can learn over time.

Maintaining a consistent state between overlapping predicates requires enforcing the Principle of Maximum Entropy over the observed feedback. The earliest efforts

in this space such as ISOMER [26] look for a distribution that is ignorant about data regions it has never seen. This assumes that the per-iteration overhead of this optimization step is worthwhile statistically. We find the resulting latency unacceptable for high-throughput systems. The constraints is to ensure that the probability distributions remain consistent with each other, making the optimizer spend more time in the estimate calculation than in query execution[36].

Feedback mechanisms are at a loss against rapidly changing data because of high-velocity inserts and deletes. We find that stale feedback results in an optimizer that reinvents past mistakes on a completely new data distribution. This assumes that the cost of retraining is greater than the cost of a bad plan. The challenge here is that the system suffers a double penalty: the latency of the feedback loop and the risk of a misguided optimization.

2.6 The Neural Cardinality Revolution

Deep learning approaches formulate row count estimation as a D-dimensional regression problem [21, 43]. Supervised estimators like MSCN learn to map a query predicate to an estimate directly by encoding join filters and table IDs as input vectors [21]. The key assumption here is that the training workload is representative of the production workload. The challenge is the so-called workload-shift problem [37]. To be sure, memorizing the joint distribution of a static schema can yield state-of-the-art accuracy. At the same time, it was shown that if the true query distribution deviates from the training set, these estimators break.

Autoregressive models like Naru and DeepDB, on the other hand, directly model the true underlying joint distribution via the chain rule of probability over conditionals[46]. They are able to generalize over arbitrary predicates without query-level training data. The key assumption here is that the model can be evaluated efficiently at query time. It was found that this assumption does not work in a production setting [42]. The challenge is the inference bottleneck. To compute the selectivity of a range predicate, one needs to either execute the model at many points or perform expensive numerical

integration.

No matter how high-fidelity the model, it is of no use if it does not satisfy the millisecond-level latency budget for join-intensive queries. Relational databases evaluate thousands of sub-plans during optimization, so a ten-millisecond inference cost per plan is fatal. It was found that the tradeoff between estimator fidelity and optimizer latency unacceptable. This led us to our tiered Waterfall fallback policy.

2.7 Specialized Estimators for String, Spatial, and Temporal Data

Substring matching over wildcard predicates, this operation breaks the assumption of finite predicate space due to the infinitude of strings. The character-level conditional probability models such as Astrid are considered to be more accurate in modeling n-gram counts and prefix-overlap[19, 49]. This assumes that the overhead of a Transformer/RNN is paid back by the improvement in filter precision. The trouble here is the compute cost. They are well-equipped to reason about the high cardinality of string fields. But the latency of a GPU-bound forward pass remains too high for general-purpose query planning.

Spatial predicates: These operators work on two-dimensional range boxes where the goal is to count intersections on a latitude-longitude grid. We find that traditional multidimensional histograms do not work here due to empty space: large regions of the input space that contain zero records [50]. This assumes that the data is dense enough to warrant a grid. Specialized models such as SAFE must explicitly incorporate the underlying index structure to preserve accuracy. The trouble here is the lack of portability. They target numeric coordinates or temporal correlations but have difficulty with mixed-type fields.

These works in the current literature train models on a single data type or operation. Is it inconvenient for real-world datasets that naturally join across categorical, numeric, and spatial fields in a single query [17, 24]. This assumes that a database

user will manually tune an estimator for every column type. The trouble here is the variety of data. A trajectory model is well-equipped to predict movement. But it cannot also estimate a simple numeric filter over the same table. This motivated our Hybrid FD CDF framework that targets statistical traits such as skew and sparsity rather than the data type itself.

2.8 The Spectral Bias of Neural Networks

Deep, fully-connected networks always learn the low-frequency components of a target function many orders of magnitude faster than high-frequency residuals.

We argue this inductive bias is unfortunate in the database distribution context, where distributions are typically Zipfian (power-law) (Assumption 1). This depends on a planner being able to reason with a smooth approximation to a density function. The problem is that database rows in aggregate form a large number of sharp, isolated spikes in density. The vanilla MLP learns the broad trend of the distribution quickly. But it also smooths out the individual high-density blobs that actually make up selectivity[43]. If the estimator fails to pinpoint a heavy hitter, it will miss the result by several orders of magnitude.

Injecting locality through mapping the input coordinates to high-frequency Fourier features before the network pass lets us recover these individual micro-densities. This mapping uses random sinusoids to project a scalar v into a higher-dimensional space: $\gamma(v) = [\cos(2\pi\mathbf{b}_1v), \sin(2\pi\mathbf{b}_1v), \dots]^\top$. This is predicated on the network being able to learn a mapping between these oscillating signals and the ground truth spikes in the data. We show that this simple pre-processing step sidesteps the spectral bias of fully-connected models. But the scale of the frequency vector $\mathbf{b}$ leaves us with a new tuning question: too small and the model is effectively oblivious to skew, too large and it models pure noise.

2.9 Latency and Resource Bottlenecks of ML-Inference

The primary limitation of a given selectivity estimator is the hard latency budget imposed by the query optimization pipeline. First, we observe that expressive neural models frequently encounter the inference limit problem during plan enumeration [9, 47]. This holds as long as the cost of a near-optimal plan is greater than the time to discover it. The issue is that a modern optimizer has to consider thousands of sub-plans for a given multi-way join. A deep neural network comes with a boost in accuracy. But if every forward pass costs multiple milliseconds, the compilation phase alone can take more time compared to a vanilla heuristic-based plan. We cannot tolerate such latencies in high-throughput OLTP or real-time analytical systems.

2.9.1 The millisecond threshold

Query optimizers have to work within a sub-millisecond budget per candidate plan to be competitive in a high concurrency regime. We also observe that autoregressive models like Naru have a heavy computational overhead when evaluating range predicates[46]. This holds as long as numerical integration over the model’s output is cheap which is not true. The issue is that the per-tuple selectivity of a multi-dimensional box must be computed by repeated sampling or a complex sum-reduce over the joint distribution. These models remove the independence assumption. But, the incurred overhead forces a trade-off between plan quality and system latency. We often observe such cases where tail latencies (p99) become extremely volatile whenever the optimizer is forced to run deep neural networks over every join permutation.

2.9.2 The SIMD vs. GPU Trade-off

Offloading cardinality estimation to a dedicated accelerator comes with a severe data movement bottleneck. We also observe that the PCIe bus latency between the CPU-resident optimizer and the GPU frequently amortizes away the raw inference time. This holds as long as the throughput boost of parallel hardware is enough to justify the overhead of a context switch. The issue is the fundamental mismatch between the

optimizer’s small and frequent requests and the huge batch sizes required to saturate a GPU. SIMD vectorization on the CPU is a viable lower-latency alternative. But, autoregressive models are notoriously hard to vectorize in this way due to their inbuilt recency bias and non-contiguous memory access patterns. AVX-512 style instruction sets will have gather to help with a non-aligned loads, but these kind of operations often come with a 50-cycle penalty. We find that this hardware/software friction has limited the widespread adoption of pure ML approaches so far. This was the primary design rationale for our Waterfall policy which reserves the costly MLP inference only to the high-density regions where all cheaper models lose confidence.

2.10 Modern Hybrid and Lightweight Approaches

Monolithic deep learning models are too slow to tune and execute at scale. This requires a single global network to memorize every nuance of a dataset without hitting the inference wall. The trouble here is the resource footprint. But, partitioning the modeling problem into many smaller sub-models that are each responsible for a low-dimensional projection of the original space allows for low-latency, high-fidelity estimation. We prefer a design that localizes learning to regions of the data[8, 51].

2.10.1 Piecewise Linear/Mixture Models

Piecewise linear functions or mixture models allow us to localize the functional form of the model to only portions of the domain[29]. The system approximates the overall distribution with a set of simple functions. This is based on an assumption that the data is locally linear within small enough partitions of the domain. Computing a dot product at runtime is extremely cheap. But, the partition strategy used to divide the domain is a key factor that determines whether the model bloats or succeeds. The trouble here is the trade-off between the range size of a partition and the fidelity of the estimate. If the partition is too wide, it misses the non-linear spikes. If it is too narrow, the memory usage bloats. We find that a simple rule, such as Freedman-Diaconis[10], used to set the boundaries of the partition based on the statistical

dispersion of the data, prevents this exhaustion while preserving local precision.

2.10.2 Updatable Frameworks/Data Drift

Learned models are static and will fail to provide useful estimates when data drifts. The only way to keep the model up to date is to fully retrain it [44, 45, 48]. We find that using functions or adding a thin incremental update layer allows the estimator to adapt to a new distribution without incurring the cost of a full table scan. This assumes that the ML component should be a residual learner and not a standalone predictor. The model only explains the high-frequency errors that a base histogram does not capture. But, this design pattern requires a tiered reliability policy to ensure that the system falls back to stable statistics if the ML component reports low confidence. The trouble here is the communication and coordination between the base statistics and the residual learner during high-velocity updates [30, 33].

2.11 Sample reservoir

To address the issue of the large amount and the data processing for calculating the statistics like IQR in the dynamic database environment, systems rely on reservoir sample as a high representation of the actual underlying data. Sample reservoir allow a query optimizer to have an up-to-date uniform random sample of a fixed size [1]. By keeping the sample reservoir, the optimizer can calculate the query estimates using a minimal computational resources which decreases the disk I/O [34] [32].

Reservoir algorithms ensure that every tuple has equal probability to be included in the sample reservoir, but some algorithms cannot sample correctly because of the high skew distribution [1]. These algorithms fail to capture the statistically significant number of tuples. This results in a problem in which there are empty regions in the sample reservoir [1].

2.12 Batch Processing in machine learning and deep learning

Batching request predictions is a technique to increase the throughput of the system by using the data parallel hardware optimizations, for example GPUs and BLAS libraries [5]. However, setting the batch size to a fixed size leads system to wait for the batch to fill or inference not completely filled batches which decreases overall efficiency [5]. Algorithms developed already handle this problem by dynamically setting and adjusting the batch size. The batching mechanism with proper batch size results in higher inference speed, the user does not have to wait for the inference, the inference is triggered either when the batch is full or when the time limit is reached. In our experiments we further analyze the batch processing approach along with the one-by-one query inference.

2.13 Research Gap: Partitioning to handle Skew

Existing work compels us to make a binary choice: model a dataset as either easy or hard all the way through. This comes at the assumption that a dataset’s value distribution across a table is smooth enough to support just one monolithic modeling strategy. The challenge here is that data skew is local. Discretizing an input domain into fixed-width buckets makes it easier to back a histogram. But, hard-coded partitioning schemes do not respect statistical dispersion as a differentiating factor between a long tail and a dense cluster. We observe that blackbox neural density estimators exhibit the exact opposite behavior: they waste computation cycles by applying an expensive model where a linear fit would do. The high-frequency anomalies of those datasets are largely hidden from view [8].

We find that existing work lacks a strategy for automating partitioning such that the scale of a model adapts to the intrinsic variance of a dataset. This requires that the per-sample cost of model capacity is a static parameter across the data domain. The challenge here is that, without local awareness, we overpay for modeling capacity

when smearing over uniform regions of a range. And we pay for it again when we fail to isolate what we call the heavy hitters that drive the lion’s share of modeling errors. Hybrid approaches provide a conceptual solution by bridging this gap [23]. But, they do not provide a principled rule to determine an optimal set of cut points without some manual tuning. We find that interquartile range provides an out-of-the-box criterion for adaptive, data-driven modeling. Freedman-Diaconis rule lets us ensure model density is proportional to the actual spread.

Shifting the policy to a waterfall pattern effectively moves the burden of proof from the model onto the data itself. This strategy comes at the assumption that the system can accurately assert that a cheap and fast linear approximation is unlikely to work for a given input range. The challenge here is that we need to make an engineering decision around the confidence threshold for safely falling back to a Fourier-Feature ML call. We find that centering the error metric on the local characteristics of an input range instead of using some global average helps us escape this trap. With this policy, the system invokes more heavyweight ML machinery only for dense spikes.

Chapter 3

Proposed System: Hybrid FD CDF Estimator

We propose Hybrid FD CDF estimator that is built on top of the Equi-Width histogram[28]. The advantage of building on top of Equi-Width is that the inference can be done in $O(1)$ time, also some of the commercial databases that already use Equi-Width histogram can just make an update to existing logic without replacing whole estimator's code, which can lead to unpredictable behavior. The innovation here is that we combine simple Equi-Width approach with the modern computational models which leads to the higher accuracy predictions of selectivity avoiding the uniformity assumption. We observed that some regions of the domain of the column have simple distributions, our approach divides this domain into buckets and then chooses the least computationally expensive model that is able to maintain low Q-error, in contrast to the new approaches that try to fit the whole distribution into one model.

In Equi-Width, the number of buckets is fixed. We use the Freedman Diaconis rule [10] to determine the optimal number of buckets initially, thus making our choice statistically supported. Then the waterfall strategy is applied to select a model. First, light models are trained that do not require a large number of resources and metrics are measured. If metrics exceed thresholds, then we proceed to the next stage, which tries to use a more complex model. Thus, we have different models in buckets that were selected based on the distribution of data in the bucket. To keep the models up-

to-date, we propose new two stages, the finetune stage adjusts the model parameters without full retrain, which is a low cost operation, if finetune does not bring the desired results, the rebuild stage is underway if the distribution of data in the bucket has changed and the model chosen initially is not able to improve the accuracy of the forecast.

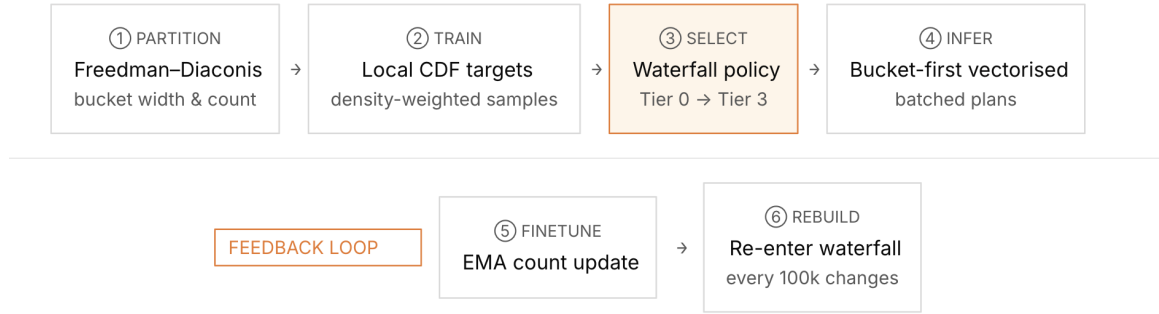


Figure 3-1: Hybrid build and maintenance stages

3.1 Initial domain learning and buckets setup

One of the main objectives is to learn how to calculate the selectivity with high precision on the skewed data. The initial Equi-Width approach uses the hardcoded value, however this does not address the issue when data is skewed and there can be case for example that in 100 buckets the first bucket contains 99% of data and possible workload, so the understanding of the data distribution within that bucket takes more resources, we need a way to split that bucket into several while preserving the equal bucket width therefore we need to calculate the bucket count more efficiently.

3.1.1 Freedman-Diaconis (FD) Rule

This algorithm calculates the optimal bucket width h based on the statistics of the data[10], by knowing the width we can further calculate the bucket count. The optimal width h is defined as:

$$h = 2 \cdot \frac{\text{IQR}(\mathbf{x})}{\sqrt[3]{N}} \quad (3.1)$$

Where the components are defined as follows:

- Sample Size (N): Represents the total cardinality of the dataset (in our case $N = 6 \times 10^7$).
- Interquartile Range (IQR($\mathbf{x}$)): $Q_3 - Q_1$, this helps to learn the distribution of the middle 50% of the data. Unlike standard deviation (σ) the IQR is a reliable statistic, meaning it is resistant to outliers. This is important for database workloads where extreme values (for example, sentinel values like -1 or 9999) can artificially affect variance estimates. Note: to calculate the exact Interquartile Range (IQR), we need to sort the whole dataset, which takes $O(N \log N)$ time and space. In the case of big datasets (like $N = 6 \times 10^7$), this is too slow for on-the-fly query optimization. To overcome this, we approximate the distributional moments using a Reservoir Sample [40] $\mathcal{S}$ of fixed size $|\mathcal{S}| = 10^5$. The following sample reservoir has fixed size of 10^5 along the whole project. The reservoir sample is calculated by performing a single pass over the data to populate the reservoir making sure that every row has equal probability of inclusion, then the samples are sorted in memory, and Q_1 and Q_3 are calculated from the sampled data. By calculating the FD bucket width on sampled dataset we reduce the sorting complexity to $O(|\mathcal{S}| \log |\mathcal{S}|)$, which leads to minimization of computations, compared to using the statistics over the whole dataset. Having the sampled dataset is helpful because, it can represent the accurate statistics of global dataset.
- bucket Width (h): The resulting scalar value defines the range of values covered by a single histogram bucket. A smaller h indicates higher granular precision, while a larger h indicates broader smoothing.

3.1.2 Memory-Constrained Saturation and the $k = 2,000$ Cap

While the FD rule gives a theoretical width, practical database systems operate under fixed memory budget. Therefore, the calculated bucket count k is subject to a hardware imposed upper bound. The final number of buckets is determined by:

$$k_{\text{target}} = \left\lceil \frac{\max(\mathbf{x}) - \min(\mathbf{x})}{h} \right\rceil \quad (3.2)$$

$$k_{\text{final}} = \min(k_{\text{target}}, k_{\text{max}}) \quad (3.3)$$

We set the limit to $k_{\text{max}} = 2,000$. This simulates a realistic memory allocation for one column statistics for single column statistics that is usually capped at a few kilobytes. If the kurtosis is low for example in uniform distribution, the high IQR leads to large H and small k_{target} for example ≈ 400 . Here the FD rule suggests that high number of buckets is unnecessary, we can handle the distribution by low number of buckets which maintaining high performance and saving memory. However, for the high kurtosis, for example in zipf distribution, the IQR is near zero which leads to the $h \rightarrow 1$ and $k_{\text{target}} \gg k_{\text{max}}$. In that case the histogram reaches the bucket limit 2000 this triggers the Hybrid approach to use the heavy models.

3.2 Local Training Set Construction

The training process of each bucket is done independently using the sampling reservoir[40]. In the training stage, the training workload is generated so that the hybrid model uses the bucket's distribution aware samples. We construct these by sampling the frequency distribution and mapping observed counts to a normalized cumulative distribution function.

3.2.1 Coordinate Normalization to Unit-Bounded Space

In machine learning the models should be transferrable and stable numerically. All input values inside the bucket range are converted into a value in range $[0; 1]$. The labels hold the local cumulative mass that we want to approximate, they are created by subtracting out the rank of the value with respect to the start of the bucket.

$$x = \frac{v - L}{H - L + 1}, \quad y = \frac{\text{Rank}(v) - \text{Rank}(L - 1)}{C} \quad (3.4)$$

C refers to bucket cardinality and $\text{Rank}(v)$ is the prefix sum of frequencies. By normalizing values we can be sure that learning tiers (Linear, Polynomial, MLP) are always learning in unit-bounded space regardless of the absolute magnitude of the domain.

The sample reservoir size is maintained at $\mathcal{S} = 10^5$. This optimal size gives sufficient resolution to capture high selectivity predicates, and ensures that the $O(|\mathcal{S}| \log |\mathcal{S}|)$ sorting operations for the IQR are calculated within a milliseconds.

3.2.2 Complex Sampling Strategy

In order to remove empty spots from local models observation space, our training set is built using a combined sampling strategies. Firstly, uniform sampling which gives the brief information about the data distribution. Grid sampling gets the samples evenly distributed across the bucket, this has shown efficiency at identifying the non linear transitions. To enhance the MLP model understanding of the data distribution we utilize edge sampling that makes sure that the model knows about the distribution near the bucket boundaries, so $L, L + 1, H - 1, H$ are forced into every training set.

3.2.3 Density-Based Importance Weighting

To improve the accuracy on high frequency values in the training process we use the density aware loss function that applies the sample weights based on the local probability density P_{local}

$$w = 1.0 + \left(\frac{P_{local}}{\max(P_{bucket})} \right) \cdot 5.0 \quad (3.5)$$

This ensures that the waterfall policy takes into the priority the learning of high density regions in which the estimation errors would result in the greatest impact on the query optimizer’s cost model. This uses the sample reservoir and it’s frequency map information.

3.3 The Waterfall Model Selection Policy

We concluded that using complex model in each bucket is inefficient use of resources[9]. Since lightweight models can achieve the same result as the complex ones, we are following the waterfall selection strategy, in which we start training from the lightweight approaches and advancing to more complex if the error is still high. This approach, which takes into account the cost of training the model, allows us to avoid using excess resources and, most importantly, helps us simplify and speed up the training process since we need to make the minimum number of improvements over Equi-Width. The models are divided into 4 tiers. Tier 0 is the identity model, which evaluates for the uniformity assumption, if a simple function $F(x) = x$ can achieve median Q-error below the threshold, in our experiments it’s 1.3, then model selection stops. The threshold of 1.3 is calculated based on the tolerance of modern query optimizers that preserve the optimal physical join order when the Q-error is below 30% which is 1.3 in Q-error. For this tier no model parameters are in the bucket, also keeping the inference as simple as $O(1)$. If Q-error is higher than threshold then next step starts with training Tier 1 parametric regressions . After careful testing of different parametric regressions, we selected three most effective ones, linear, power law and log linear models. These models can fit vast majority of intra bucket distributions. Ridge regression regularization coefficient is set to $\alpha = 1.0$, that prevents over adjustment of coefficients that can occur because we are using sample reservoir. The problem with that model is that using the least squared method it give each training sample same importance. We found that using this sample weight function

$w = 1.0 + (P_{local}/\max(P_{bucket})) \cdot 5.0$ helps model to learn the highly concentrated point area, in which the most of the critical errors occur. All 3 models fit are done in parallel to find which model results in least error. After fit each candidate is validated against the 500 synthetic range queries to ensure that the Q-error is below 1.05. The power-law ($x^{0.5}$) and log-linear ($\ln(x + \epsilon)$) models are added here for capturing skewed intra-bucket distribution. The $\epsilon = 10^{-7}$ in the log linear ensures the numerical stability near the zero coordinate boundary of the bucket. This tier captures non linear intra-bucket distributions without the storage overhead, it stores metadata with a single weight w_1 and an intercept w_0 for each bucket, also this tier serves as early exit point that ensures that the algorithm proceeds to the multi layer perceptron when the intra-distribution is mathematically chaotic.

Next tier 2 that contains non parametric learners, when tier 1 cannot handle the data distributions that have skew and large gap with no values. The rudge regression attempts to interpolate across these gaps which leads to overestimation in empty regions and we solve that by using Isotonic regression. This makes sure that the predicted CDF is monotonic. Isotonic regression can be considered as a improvements in global consistency sacrificing the local precision. In other words, tier 2 also fits the decision tree regressor that is restricted to a depth of 4. This depth is a arbitrary limit that prevents the tree from over learning the noise from the sample reservoir, while also providing enough capacity to isolate distinct sections. We validate both structural models against the same 500 query used in parametric tier, if the median Q-error remains above the exit threshold we accept the increased computational usage of these non parametric models to avoid the higher latency of neural network. This tier is great at capturing the staircase shape of the data.

Tier 3 consists of neural network with Fourier feature MLP. Standard Multi-Layer Perceptrons are mathematically biased toward low-frequency functions. They smooth the sharp transitions that occur in skewed distributions, viewing the local data spikes as a noise. We find that row $[0, 1]$ coordinate is not enough signal for a weight matrix to differentiate between closely packed heavy-hitter values. The problem that occurred is that tier 4 model spent its capacity learning the global mean instead of

local spikes. We solve this issue by calculating the scalar input into a 66 dimensional feature vector using a Fourier feature mapper [38]. This makes the neural network to see the data as multiple spatial scales at the same time. It uses $K = 32$ frequency bands distributed across a log-scale from 10^0 to 10^3 . We also include the raw input x and a log-transformed term $\ln(x + 10^{-7})$ to maintain resolution at the bucket’s zero-boundary. For every frequency f_i we generate $(\sin(\pi f_i x), \cos(\pi f_i x))$ pair, and it helps in linearize the non linear density shifts. High frequency bands can fit the heavy hitters in the data, and low frequency bands help in preserving the general curvature. These features then passed into the 2 layer MLP with a hidden dimensions of (64, 32). We set the iterations to 500 to allow the ADAM optimizer [20] to have enough steps to navigate the high dimensional loss surface in an unknown distribution. This tier is the most expensive operations, we only use it as a fallback when the median Q-error exceeds 1.05 after checking other tiers. Storing the weights for the model consumes more memory than a two coefficient ridge models, but the low Q-error that it achieves justifies the storage cost for the most chaotic regions of the domain.

3.4 High-Throughput Vectorized Inference Engine

Even if we achieve the high performance model that takes milliseconds to inference we still have to inference thousands of predicates, that forces us to find ways to optimize inference for multiple of them, and that’s when vectorized inference engine play important part. Instead of inference of query predicates one by one we do the inference grouped by bucket, meaning if more than one query predicate uses one bucket then the calculations would be done in parallel.

3.4.1 Processing in batches

Firstly, we process the workload using the overlap masking and then we utilize the SIMD accelerated pipeline for fast calculations of the start and end positions of masked queries. The batch size is set to 1 million queries, each query has at least 2 predicates. Then the inference of the models is started, for regression models the

following equation is used:

$$\mathbf{y}_{\text{pred}} = \mathbf{X}_{\text{norm}} \cdot \mathbf{W}_j + b_j \quad (3.6)$$

where $\mathbf{W}_j$ is the matrix of coefficients for bucket j . This matrix multiplication is important part since tier 1 models inference can be unified and can use it.

After inference the results are aggregated and the final selectivity is calculated

3.4.2 Processing in queue

Batch processing improves the overall throughput, but it introduces new latency penalty because now the query must wait for the batch to fill. To address this issue, if the batch processing is not applicable for the inference for latency sensitive environment, we also conduct the experiments by inferencing the queries one-by-one and compare the results.

3.5 Dynamic Maintenance and Feedback Cycles

In production environment data is not stationary and the distribution can change over time. To address this issue, the Hybrid FD CDF approach uses a pipeline that maintains the internal states of the buckets so that they would be relevant even after massive data changes, also without full retrain[44], meaning we fine tune only those buckets that perform bad, and also rebuild from scratch the buckets to which finetune process did not help.

3.5.1 Finetune process

When queries are arrived and query optimizer calculates the prediction at the end we can get the actual row count for that query, meaning we can in real time monitor the performance of query optimizer and if certain bucket results in bad prediction ($\epsilon_{\text{bad}} = 1.2$). Then this query is saved as data for future finetune process and bucket

is marked to be finetuned. In finetune process we update the row count of the bucket using an exponential moving average

$$C_{new} = C_{old} \cdot (1 - \alpha) + (C_{old} \cdot \text{Ratio}) \cdot \alpha \quad (3.7)$$

In the experiments we set the learning rate to 0.15 and Ratio is y_{true}/y_{pred} . We force ratio to be within the range of [0.2; 5.0] so that the large outliers would not heavily affect the count in the bucket, avoiding the issue when bucket forgets the initial training. That process helps estimator to adapt quickly to shifts in volume of the data.

It is worth to note that the sampling reservoir has a flaw. If the domain of data is [0 ;1.000.000] and we have sample reservoir that consists of 100.000 samples, it can successfully provide the high representation of the data, but it cannot physically provide the information whether there is certain data points or not leading to the issue that the bucket has 0 count, while in reality it has values. To address this issue we implemented a mechanism that helps in identifying that on fly and setting count to the right value

$$C_{revived} = y_{true} \cdot \alpha \quad (3.8)$$

This mass injection ensures that previous regions that were not covered by sample reservoir now have the relevant information about the bucket count. This operation helps to deal with a sample reservoir flaw making it usable.

3.5.2 Rebuild process

While the finetune process adjusts the value counter in the bucket, it do not change the model that was initially selected for that bucket and new problem appears, since the data is changing the intra-bucket distribution can change from uniform to skewed and if uniform model was initially selected for the bucket then it would not be able to handle the new skewed distribution, that introduces the need for the rebuild process. In the finetune stage, if after the finetune bucket still performs inefficiently then it is marked to be rebuilt at the checkpoint. The checkpoint is the hard-coded value of

the data changes, in our design we set it to be 100.000, that can consist on insertions, updates and deletions. After the checkpoint is reached the marked buckets for the rebuild are starting training process from scratch, using the sample reservoir that database systems keep up to date the buckets are selecting the appropriate model that can handle the new distribution.

Rebuild process consumes more resources and in that mechanism it is done only when it is needed, that ensures optimal resource usage.

3.6 Theoretical Analysis

3.6.1 Complexity Analysis

The computational efficiency of Hybrid FD CDF estimator is based on the maintenance of the low latency during the construction and inference stages. The construction complexity is following, the Freedman Diaconis rule requires the sorting the sample reservoir which is $O(|\mathcal{S}| \log |\mathcal{S}|)$. The training overhead dynamically scales and depends on the underlying distribution, the best case scenario is when all models are Tier 0, which take $O(1)$ per bucket, the worst case scenario Tier 3 using the Fourier Feature MLP takes $O(E \times B \times (F \cdot H_1 + H_1 \cdot H_2))$, where the E is the number of epochs (500), B is the bucket count, F is the feature dimension (66) and H_1, H_2 are the hidden layer dimensions (64, 32) respectively. The inference complexity is following, the Equi-Width inference is $O(1)$ per bucket, the Hybrid FD CDF estimator matches this in Tiers 0 and 1 via direct matrix multiplication. While in Tier 2 it operates in $O(d)$, where $d = 4$ is the maximum tree depth. Tier 3 inference complexity can be derived as $O(F \cdot H_1 + H_1 \cdot H_2)$.

3.6.2 Error Bounds

In the Equi-Width histogram, the estimation error is unbounded in the extreme skewed distribution. The Hybrid FD CDF estimator restricts this error by executing the candidate models and validating them against the set of 500 validation queries,

the expected error is set such that $E[\text{Q-Error}] \leq \tau$, where $\tau = 1.05$ for parameterized models. If no models satisfy this condition then Tier 3 Fourier Feature MLP serves as a function approximator.

3.6.3 Convergence Behavior of Dynamic Maintenance

The fine tuning updates the bucket counts using the exponential moving average. The convergence is defined at time step t as:

$$C_t = C_{t-1}(1 - \alpha) + (C_{t-1} \times R_t)\alpha$$

, where α is the learning rate 0.15 and R_t is the feedback ratio $\frac{y_{true}}{y_{pred}}$. We constraint $R_t \in [0.2, 5.0]$ to prevent the parameter divergence due to outliers. The maximum state modification per update step is strictly bounded. The $\alpha = 0.15$ ensured the smooth asymptotic convergence to the true target distribution without amplifying the variance, which is induced by transient workload noise.

3.7 Hybrid Inference and Execution Walkthrough

The Hybrid FD CDF estimator has a focus on low-latency execution using a bypass mechanism during the inference. To demonstrate the inference logic, we trace the execution of a single range query $Q = [q_{low}, q_{high}]$.

The estimator first maps the query boundaries to their starting and ending bucket indices B_{start} and B_{end} , this is calculated via a constant time operation relative to the minimum and the global bucket width h . Then the estimator iterates over the range $[B_{start}, B_{end}]$ and evaluates them. If bucket is fully enclosed, then is it an intermediate bucket that $q_{low} \leq \text{bucket}_{low}$ and $q_{high} \geq \text{bucket}_{high}$. The estimator does not use the machine learning inference entirely, it extracts and calculates the total scalar mass C_{bucket} of that bucket. For these buckets the estimator bypasses the unnecessary operations operating in $O(1)$ efficiency per bucket. In the edge buckets B_{start} and B_{end} the local models are invoked, the estimator gets the specific model assigned

during the waterfall model selection stage. Then the query overlap is normalized to a local coordinate space $x_{norm} \in [0, 1]$. The model predicts the CDF values at the upper and lower intersection points x_{hi} and (x_{lo}) respectively. Then the total row mass is calculated:

$$\text{Count}_{pred} = \max(0.0, \text{CDF}_{hi} - \text{CDF}_{lo}) \cdot C_{bucket} \quad (3.9)$$

Chapter 4

Experiments results

4.1 Datasets Generation and Management

4.1.1 Dataset Manager Framework

To maintain a structured project architecture, a dataset management module was developed. This module is a framework that is responsible for creating a dataset and maintaining metadata. This module also helps to simulate the real database.

The initial function of the Dataset Manager is performed as follows:

1. Data Generation: The raw data is generated depending on the probability density function.
2. Statistical Profiling: Next, the dataset manager makes one scan of the data to calculate global statistical data such as minimum maximum, number of unique values, skewness and kurtosis
3. Metadata Caching: To speed up the experiments, the manager stores all the metadata, so when we start the experiment, we don't need to calculate the metadata again, we load it. For example the relevant sample reservoir for certain step in the drift experiment scenario.
4. Visualization Diagnostics: The framework automatically generates graphs for diagnostics, global histograms and micro histograms so that we can see the

overall picture of the data distribution, as well as a micro picture to understand what distribution is inside the bucket.

4.1.2 True Fractal Generation Logic

The main problem we encountered during the selectivity calculation is that synthetic data generation works in such a way that when generating a large amount of data, the micro-distribution is uniform, and the Equi-Width approach would produce the best results on such dataset, but since we need to test data that is as close to reality as possible, we implemented an algorithm for creating micro distributions.

The micro distribution generation method uses a fixed number of 1000 mini buckets. Thus, dataset generation is divided into 2 parts. Macro-Level Distribution layer defines the destination bucket. Micro-Level Distribution is second stochastic calculation, the offset is chosen from 0 to 999 range where the value will be. The probability density function of this micro level distribution offset follows the shape of the global probability density function.

Mathematically, a value v is generated as:

$$v = \text{offset}_{lo} + (\text{bucket}_{macro} \times 1000) + \text{offset}_{micro}$$

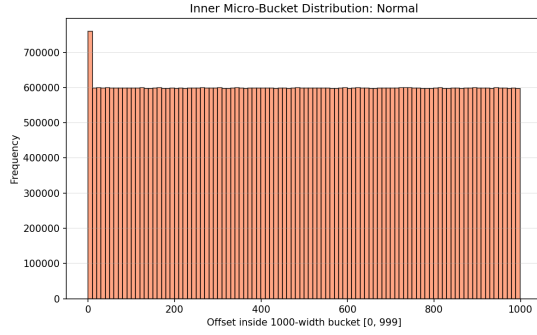
where bucket_{macro} and offset_{micro} are drawn from the same family of distributions.

For example, when generating the the fractal normal distribution dataset the micro offset is calculated using the mini bell curve at 500 with a standard deviation of approximately 166.6. By doing this we are ensuring that micro distribution within the bucket will not be uniform so we can stress test all approaches.

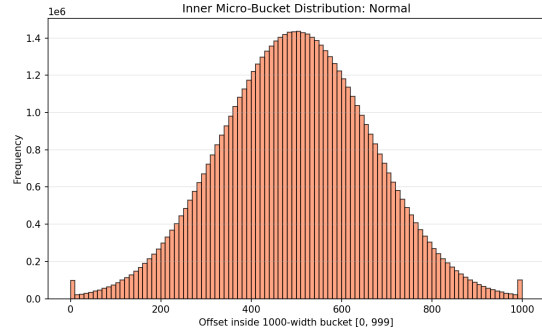
In another example, for zipf distribution this approach created a cliff inside every bucket, so that the estimator would have to learn the sharp gradients regardless the query start and end.

The visual representation between simple data generation approach and our stress test generation logic is illustrated in Figure 4-1. In Normal and Zipfian cases, the micro distribution follows the global trend to test the local model capabilities. In

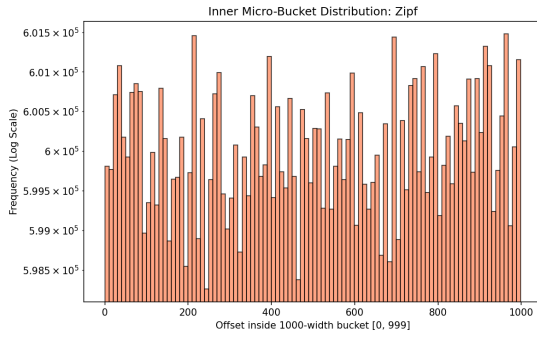
others datasets, the micro distribution the simple and our approach preserve the same picture due to the nature of the datasets



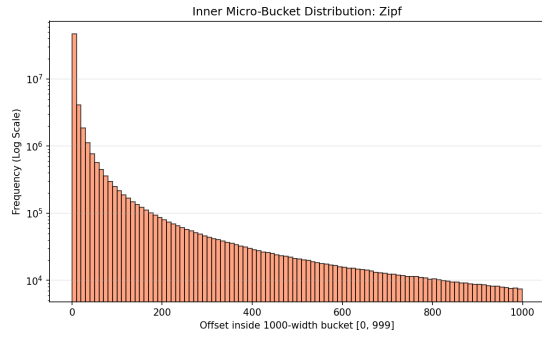
(a) Normal: Simple (Uniform Micro)



(b) Normal: Fractal (Gaussian Micro)

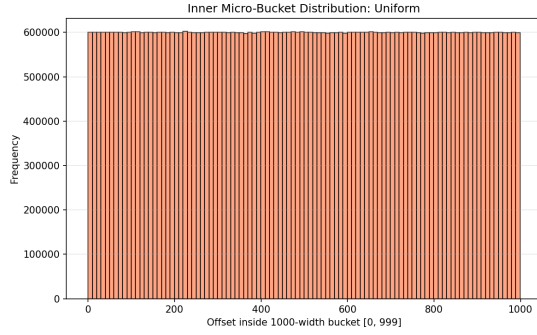


(c) Zipfian: Simple (Uniform Micro)

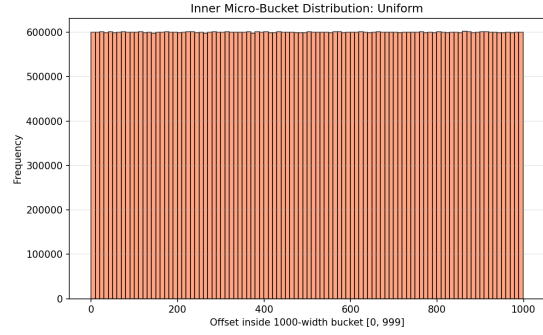


(d) Zipfian: Fractal (Zipfian Micro)

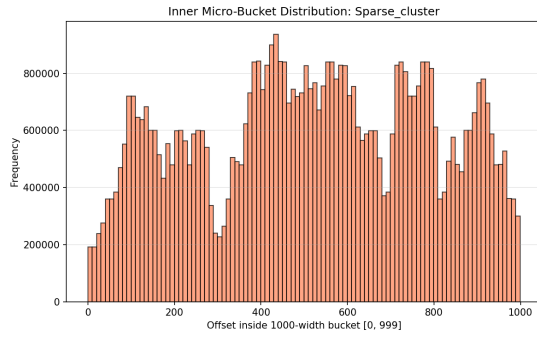
Figure 4-1: Comparison of inner micro-bucket distributions for complex non-linear gradients (Part I).



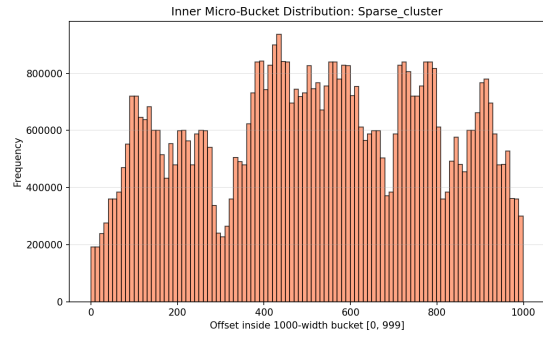
(e) Uniform: Simple (Uniform Micro)



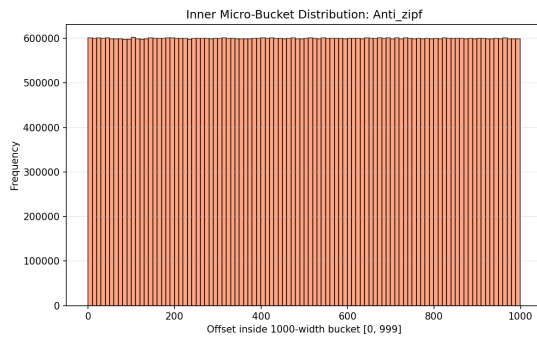
(f) Uniform: Fractal (Uniform Micro)



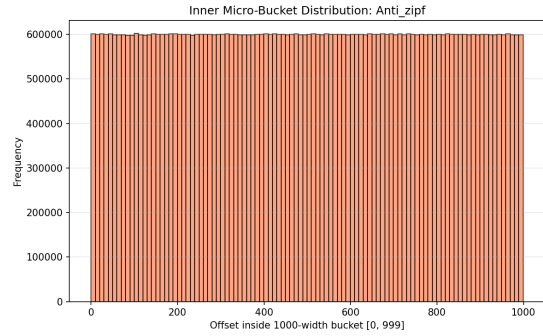
(g) Sparse: Simple (Uniform Micro)



(h) Sparse: Fractal (Clustered Micro)



(i) Anti-Zipf: Simple (Uniform Micro)



(j) Anti-Zipf: Fractal (Uniform Prefix Micro)

Figure 4-1: Comparison of inner micro-bucket distributions for statistically simpler or sparse distributions (Part II).

4.1.3 Scientific Determinism and Reproducibility

To ensure that the experiments and performance of the trained Hybrid FD CDF estimator are not biased, we setup our experiments in deterministic manner making sure the every run will have the same result. Global seed 42 is enforced through the whole module. The random seed set helps with data integrity, the exact coordinates and density regions of the 50 data clusters in sparse cluster dataset remain identical. For the sampling process, specific 100,000 rows selected for the estimator observation window, this makes sure all approaches work with the same sampled data. The training process of Fourier Feature MLPs are also deterministic so that the random selection of training points within the buckets are reproducible. This allow to tune the hyperparameter and making the conclusions whether the increased some parameter really helped and it was not the random weight initialization for example. By elimination the randomness from the experiments we can be sure that the Q-error metrics and training times reported are deterministic and can be verified and reproduces accross different compute setups.

4.2 Static Benchmark Distributions

To conduct experiments with wide variety of data distributions we selected five different probability distributions. The benchmarks have different levels of skewness, sparsity and gradient simulating the real world database scenarios. The properties of each dataset are summarized in table 4.1.

Table 4.1: Calculated Statistical Moments and FD Buckets Parameters ($N = 6 \times 10^7$)

Dataset	Skewness	Kurtosis	Bucket Width (h)	Bucket Count (k)	NDV
Uniform (Baseline)	0.0001	-1.199	2564	390	1000000
Normal	0.0001	-0.080	1150	869	929176
Zipfian ($a = 1.5$)	3.04	9.11	150	2,000 [†]	213901
Sparse Cluster	-0.298	-1.142	2166	421	4974
Anti-Zipf	0.0001	-1.199	255	392	100000

Note: Bucket count capped at 2,000. Zipf parameter $a = 1.5$

4.2.1 Baseline Distributions

The Uniform distribution is the control option for our experiments. Data is generated with near-zero skewness (≈ 0.0001) across the entire domain. This scenario validates that the Hybrid Estimator does not introduce unnecessary computational overhead or hallucinate patterns when the data is perfectly flat.

The Normal (Gaussian) distribution serves as real data testing, where in real world a lot of natural phenomenas are normal distributed. We created Normal distribution centered at 500.000 with a standard deviation of $\sigma = D_{\text{span}}/6.0$. The fractal version of this dataset was also created, that tests the estimator for ability to fit the data in the bucket that has not standard distribution.

The Zipf distribution was added for simulating the highly skewed distribution with a power law parameter 1.5. That distribution creates a skew that keeps millions of rows in a narrow range of unique values, this scenario is usually on which Equi-Width performs the worst, linear interpolation in such distribution is also not helpful. So since the highly skewed distribution is not rare occasion and Equi-Width approach poorly performs on it, minimizing Q-error on this distribution is one of our main goals.

The Sparse Cluster distribution is included to simulate scenarios where the data is distributed with dense clusters with a void between them. There are 50 dense clusters each with width of 100 units randomly placed across the domain.

The Anti-Zipf distribution simulates a scenario in which the data is distributed uniformly with a limit, for the experiments we set the limit to be 10% of a domain, $[0;100.000]$, the data is uniformly distributed and there is no data in $[100.001; 1.000.000]$. This distribution serves as another baseline in which the Equi-Width will perform good, so we need to test whether the Hybrid estimator will identify the uniform distribution thus selecting the most simple model for that buckets and also identifying the buckets with no values.

The visual representation of all distributions is in 4-2.

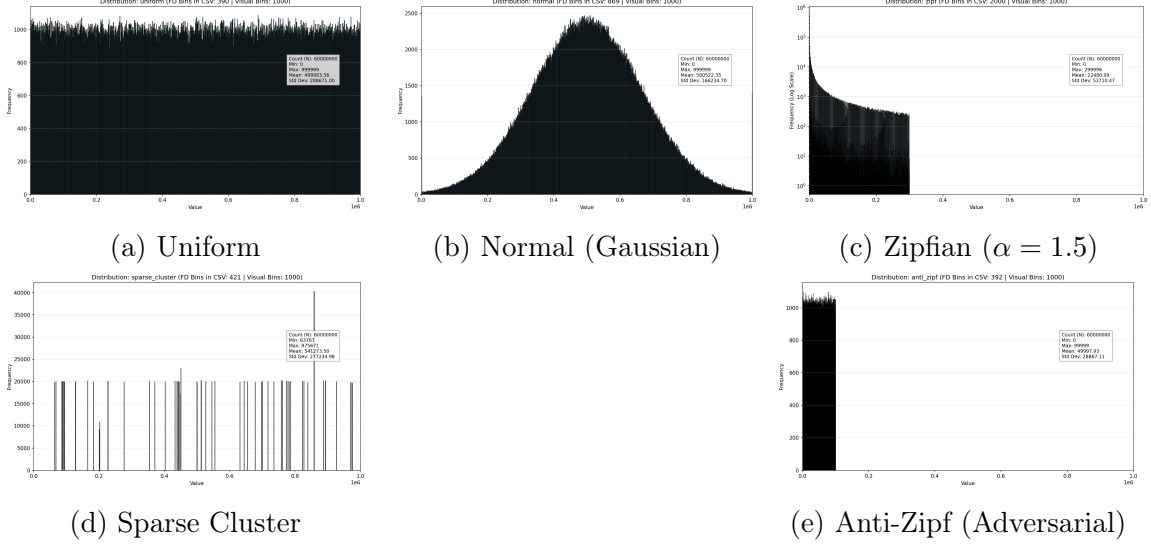


Figure 4-2: Visualization of 5 synthetic data distributions ($N = 6 \times 10^7$).

4.2.2 DSB Benchmark

To test Hybrid FD CDF on real data we use the Decision Support Benchmark (DSB)[18], which contains complex real environment and data distributions. We selected three columns for experiemnts: `ws_item_sk` is column for keys for a web sale items with a standard distribution. `cr_item_sk` is column for returns item keys, it's distribution has a skew which will complicate the task for the linear interpolation methods. `cr_ret_time_sk` is column to temporarily stored keys with low cardinality distribution, serves as a baseline for testing simple case.

4.3 Drift modeling

In production database systems, the data is changing and distribution does not remain the same, we create a drift dataset to test the Hybrid FD CDF estimator for the ability to adapt for new data while not introducing new overhead and keeping the inference time comparable to Equi-Width approach.

4.3.1 Drift types

The drift scenarios in experiments have two setups. Firstly, the displacement type is in which the initial distribution is gradually is being replaced by the target distribution. The total number of rows is kept at $N = 6 \times 10^7$. This type is for testing the Hybrid FD CDF in the scenario where data changes and buckets need to adapt, since the total number of rows is fixed, the inference time is supposed to be the almost the same at each step of data drift. Second type is additive growth of the data, the target distribution is being appended to the initial distribution. This type is for testing the Hybrid FD CDF, how the inference time is increasing compared to the baselines. The inference time is supposed to be increasing linearly, not exponentially. So that the Hybrid FD CDF approach can be applied when data is scaling. The total number of rows is changed from $N = 6 \times 10^7$ to $N = 1.2 \times 10^8$ rows.

4.3.2 Linear transition and resolutions

The transition between the initial dataset state to the final is calculated via steps. In standard setup we set the number of steps to 10% of the data, here we test the case when 10% of the data is changed and Equi-Width is rebuilt, how metrics from Hybrid FD CDF degrade with the data change and how quickly and efficiently it can finetune and rebuild buckets that perform not good. As the experiments progressed, the 10% change was not sufficiently captured the metrics difference between the approaches, so we set high resolution 5% step that will contain 21 steps overall. The step is considered as hyperparameter and is subject for further analysis, in our experiments having two setups will allow to analyze the model’s performances with different number of step sizes.

The formula for a dataset at certain step is as following $\mathbf{f}_{step} = (1 - \alpha)\mathbf{f}_{init} + \alpha\mathbf{f}_{target}$, while for additive growth, it is $\mathbf{f}_{step} = \mathbf{f}_{init} + \alpha\mathbf{f}_{target}$.

4.3.3 Sample reservoir maintenance simulation

In experiments the models do not operate on full dataset since it is not scalable. The models use the sample reservoir, which databases keep up to date using different algorithms, the sample reservoir maintenance is another research direction. For simplicity in the experiments we simulate the fresh and relevant sample reservoir presence by creating it following the steps: Firstly, the array of N rows is creating by doing the full scan. In reality full scan on data is not done since the sample reservoir is maintained across the data changes. Second step is to create a Bernoulli-trial mask is applied to the array with a selection probability $p = \min(1.0, (S \times 1.5)/N)$, where $S = 100,000$. At this step we have about 150.000 samples. Thirdly to maintain the constraint of 100.000 samples the result is randomly truncated from 150.000 to 100.000 samples.

4.3.4 DSB Benchmark

DSB columns were also subjected to a displacement drift scenario, where a 5% step size was used to test the model robustness on a real-world data structure. In contrast to synthetic data, where the fractal correlations of the synthetic data are defined by an exact mathematical relationship, the DSB column correlations are not. This drift test therefore was a test of the Hybrid FD CDF's ability to continue to perform with precision on data where the fractal nature of the data does not behave ideally. In particular, we wanted to see if the Freedman-Diaconis based structural partitioning that was defined at the beginning of the drift still held true as the data drifted, or if the drift forced a continual rebuilding of the model internals.

4.4 Workload and Evaluation Metrics

4.4.1 Initial Strategy: Uniform Random Workload

Initially the experimental workload consisted of range queries that uniformly cover the domain $[0; 1.000.000]$ and workload was data agnostic.

Query Parameters As the starting point, 1,000,000 queries were generated, this scale is chosen because it provides a statistically significant results that are not biased or affected by random queries. The generation is done in a two stages: First one is anchor selection, in which the starting point of the query range L is uniformly sampled. Formula for that:

$$L \sim \mathcal{U}(D_{\min}, D_{\max} - 1) \quad (4.1)$$

. To stress test local precision, we constraint the query length to 100 units, so the query covers 0.01% at max. The formula for query width:

$$W \sim \mathcal{U}(1, 100) \quad (4.2)$$

This uniform strategy provided starting point for the further workload generation advancement. It was discovered that in sparse dataset, in which a lot of buckets contain 0 values, if we still query this buckets then the accuracy would be nearly perfect, so in the experimental results this issue will be addressed. The fix for that is data driven workload.

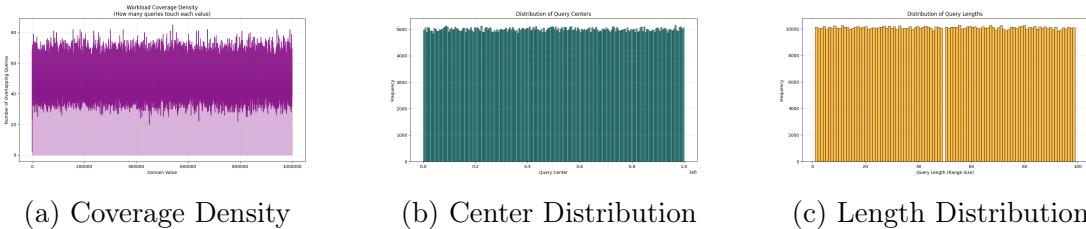


Figure 4-3: Visualization of narrow range queries statistics.

Justification for Extreme Narrowness. The logic behind constraining the query width to 100 is based on the experiments progression, it was noticed that if query covers more than 2 buckets then there can occur error averaging effect. In which the Hybrid FD CDF estimator only can have significant errors in side buckets, while the intermediate buckets give almost certain number of rows contained. Thus, if query covers 80 buckets and we need to test approach and give significant weight to the

side buckets, by covering high number of buckets we can not evaluate performance correctly. By covering least number of buckets we force the model to rely on ML models in buckets rather than linear interpolation.

The following figures will illustrate the issue:

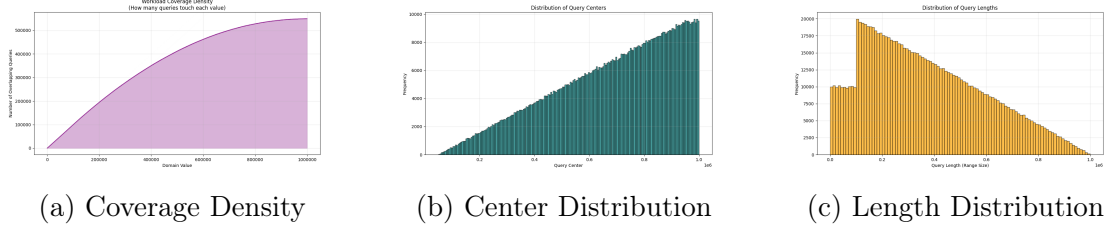


Figure 4-4: Visualization of wide range queries statistics.

Here we observe that right buckets are tested more than left ones.

4.4.2 Advanced Strategy: Data-Driven Workload Generation

The uniform strategy ensures the uniform coverage of all buckets, however in skewed or sparse data we need to stress test the regions of high density. The core objective of data driven workload generation strategy is to focus in these regions. By using the 100.000 row reservoir sample as a probability density map, queries were generated that are guaranteed to cover the real data points, by doing this, no queries can give erroneously good results or affect them in any way.

Query Parameters using Reservoir Sampling As the approach changed from the uniform to the data aware workload generation, the creation process was also changed. First anchoring, instead of uniform domain sample, an anchor point of a query is selected using random choice from the 100.000 sample reservoir, this ensures that the query starts in the area where data exists. Formula:

$$v_{anchor} = \text{rng.choice}(\text{Sample}, \text{size} = 1) \quad (4.3)$$

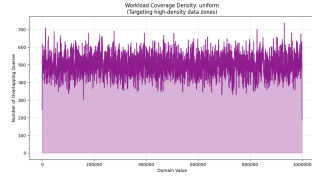
Then with random width generated of range [1; 100] units, the query is created with a v_anchor in the middle of the query.

Characterizing the Data Driven workload To confirm that the Data-Driven strategy effectively uses the data density information, we have computed and visualized the full structural breakdown of the 1,000,000 generated queries for each distribution. The result is depicted in Figure 4-5.

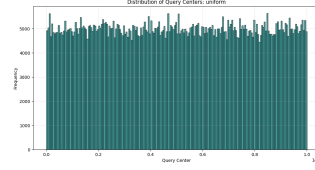
This visualization serves as empirical proof of three critical properties of our workload:

1. **Data aware center:** The *Center Distribution* plots (middle column) are no longer spatially agnostic. They have now become a perfect reflection of the PDF of the underlying dataset, with, e.g., the centers of Zipfian skewed towards the head, or, with the Sparse Cluster, 50 clearly identifiable spikes.
2. **Length distribution:** On the other hand, the Length Distribution (right column) is still strictly uniform between 1 and 100.
3. **Coverage Saturation:** The Coverage Density (left column) highlights the regions most frequently visited by queries, showing a near-perfect overlap with the actual data regions.

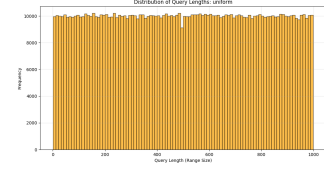
Elimination of Sparsity queries We have eliminated thousands of queries that targeted empty portions of the domain, the data driven hits are forced. Every individual query in the million-count workload is therefore guaranteed to have a non-zero result for the Q-Error to represent the actual performance of the estimator.



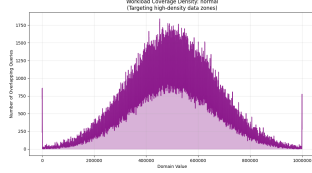
(a) Uniform: Coverage



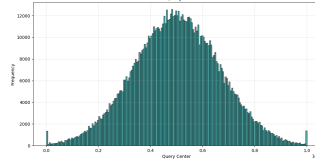
(b) Uniform: Center Dist



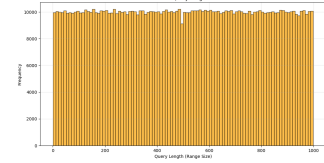
(c) Uniform: Length Dist



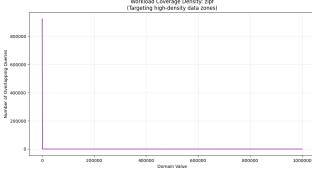
(d) Normal: Coverage



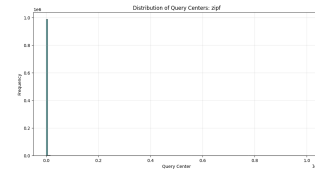
(e) Normal: Center Dist



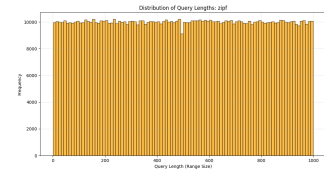
(f) Normal: Length Dist



(g) Zipfian: Coverage

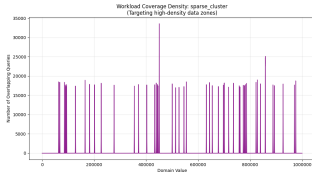


(h) Zipfian: Center Dist

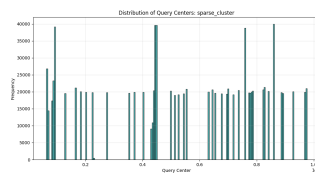


(i) Zipfian: Length Dist

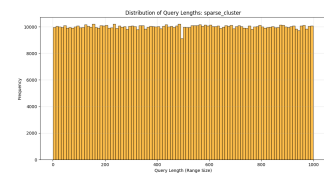
Figure 4-5: Analysis of the Data-Driven workload (Part I: Uniform, Normal, and Zipfian).



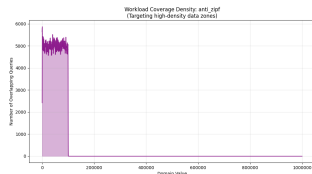
(j) Sparse: Coverage



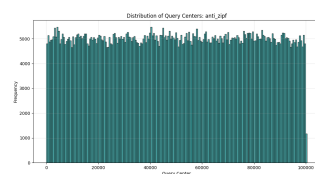
(k) Sparse: Center Dist



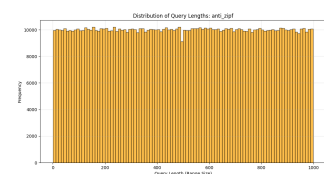
(l) Sparse: Length Dist



(m) Anti-Zipf: Coverage



(n) Anti-Zipf: Center Dist



(o) Anti-Zipf: Length Dist

Figure 4-5: Analysis of the Data-Driven workload (Part II: Sparse Cluster and Anti-Zipf).

4.4.3 Workload Persistence and Reproducibility

The workload is stochastically generated, however the random seed is forced to be 42. By using precisely the same 1,000,000 query intervals for each model, sampling variance is not a confounding factor in the Q-Error.

4.4.4 Workload for drift benchmark

We introduce the need of data driven workload to fairly evaluate the performance of proposed Hybrid FD CDF approach. In the following experiments data it will be justified by the metrics calculated along the different experimental setups, for now we consider that query patterns follow the most recent added data or the active data.

For each, data drift step we create a data driven workload that is based on the current step data reservoir to simulate the user activity that operates on the data that is present. This step is important because it simulates the user activity that follows the data as it changes across the steps, this helps to maintain the pressure on the estimator’s local resolution limits.

4.5 Evaluation Metrics

We evaluate the performance of the Hybrid FD CDF estimator against standard baseline Equi-Width Histogram and prior adaptive learners Equi-Hist using metrics on accuracy, speed, and structure.

4.5.1 Operational Efficiency Metrics

Build time metric - to construct the estimator from raw data (load and build) on a single thread. For the Hybrid Model, this includes the waterfall model selection step. We include an Early Exit Strategy in our implementation: the waterfall model selection terminates early for a bucket if the (Linear or Power-Law) estimator reaches an internal error threshold on validation, trading additional training time for reduced training overhead. Inference latency metric - the time required to estimate the selec-

tivity on the 1.000.000 workload. Update latency - for adaptive models, this measures the latency of the feedback loop. For the Hybrid model, this decomposes into two steps: *Fine-Tune Time* (T_{ft}) to adjust the bucket count, and *Rebuild Time* (T_{rb}) to incrementally perform background re-training of buckets.

4.5.2 Accuracy Metrics

Let $y_{\text{true}}(q)$ be the true selectivity and $y_{\text{pred}}(q)$ be the predicted selectivity of a query q .

The Q-Error is one of the main metric for query optimization [27]. We define it as:

$$\text{Q-Error}(q) = \max \left(\frac{y_{\text{true}}(q)}{y_{\text{pred}}(q)}, \frac{y_{\text{pred}}(q)}{y_{\text{true}}(q)} \right) \quad (4.4)$$

Mean Absolute Error: In addition to Q-Error which measures the ratio quality, we also measure MAE which is absolute magnitude of the error, specifically added for better model training process:

$$\text{MAE} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} |y_{\text{true}}(q) - y_{\text{pred}}(q)| \quad (4.5)$$

MAE is a useful metric for resource estimation (e.g. how much RAM to allocate for a hash join). An estimator may have a low Q-Error on a small query, but a very large MAE on a wide-range query will lead to observable wasted memory.

We also report the Median and P-95 Q-Errors for better understanding of the median case and the most hard case queries.

4.5.3 Drift and Adaptivity Metrics

To measure how well the model performs over a distribution shift, we track the accuracy in stages: first, a Shock Accuracy (M_{shock}), the immediate jump in Q-Error as soon as the shift occurs; then, a Fine-Tune Recovery (M_{ft}), the subsequent recovery in accuracy as Feedback Updates push the total number of rows around within the existing buckets given new query observations; and finally, a Rebuild Convergence

(M_{rb}), the eventual accuracy that is obtained after detecting bad buckets with a error greater than the threshold of 1.2 and re-training their internal model based on the reservoir sample. Rebuilt Finetune and Bucket Count, the structural metric of the number of individual buckets that were finetuned and rebuilt within a drift period. Shock metrics are included only for experiments purposes, to check how performance degrades after the data change, in production they will not be calculated.

4.6 Baselines for Comparison

4.6.1 Equi-Width Histogram

Equi-Width serves as a main baseline for the experiments , it was implemented following the basic version[28]. We need to test the performance of our new system in comparison with baseline, and since the new approach is built on top of Equi-Width, it is necessary to measure the overhead in time and memory that the Hybrid FD CDF model overlays.

Construction is done by following algorithm. The model divides the domain into k number of buckets of a fixed width, the bucket size is calculated by:

$$h = \frac{D_{\max} - D_{\min}}{k} \quad (4.6)$$

Construction stage also makes one full scan of a sample reservoir to add the counts into corresponding buckets

Equi-Width uses the uniformity assumption when it inference the range queries, it finds the side buckets using the binary search and then calculates the predicted number of rows in that buckets, and then adding the counts from the intermediate buckets.

4.6.2 Equi-Hist (Adaptive Feedback Baseline)

Equi-Hist serves as a baseline from the approaches that are built on top of the Equi-Width histogram and are learning the data domain as the workload comes [3, 4]. The implementation is done by following next steps. First, initialize the Equi-Width histogram. Next the learning parameters, in our experiments the learning rate is set to 0.5. Second stage is inference, when the engine gives the true row count of a certain query the Equi-Hist updates the counts in buckets to learn from that. Equi-Hist changes the count in every bucket that overlaps with the query, calculating how much certain bucket contributed in advance. The contribution is calculated as:

$$\omega_i = \frac{\min(q_{hi}, B_{hi,i}) - \max(q_{lo}, B_{lo,i}) + 1}{B_{hi,i} - B_{lo,i} + 1} \quad (4.7)$$

. The bucket that is fully covered has $\omega = 1.0$. Then each counter in bucket overlapped in updated using formula:

$$C_i^{\text{new}} = \max(1.0, C_i^{\text{old}} \times R^{(\eta \cdot \omega_i)}) \quad (4.8)$$

This ensures that buckets that overlap more are also changed more. Low overlap buckets are adjusted slightly.

4.7 Experimental results

4.7.1 Static setup batch inference

Synthetic data

The following experiments are done on the synthetic data. The datasets are divided into simple (without intra bucket distribution) and hard (with the specific intra bucket distribution). The workload is wide (range query that covers > 2 buckets) and narrow (range query that covers 2 bucket at most) where the query range is limited at 100.

First experiments are done on the simple dataset without intra bucket distribu-

tions. The workload used is wide. As the result in Table 4.2 we see that almost all models perform well with perfect Q-error = 1, this happens because the wide queries are averaging the error so that the side bucket’s errors do not have enough weight to show the actual performance. It is worth to note the Equi-Hist inference time, since this algorithm updates its bucket’s counts after each query, it creates the significant overhead.

Table 4.2: Benchmark Results (Dataset: SIMPLE, Workload: WIDE). Error metrics are reported as Symmetric Q-Error.

Distribution	Model	Avg Q-Err	25%	75%	95%	Median Q-Err	Train (s)	Infer (s)
Anti-Zipf	Equi-Hist	1.000	1.000	1.000	1.000	1.000	6.404	4.833
	Equi-Width	1.000	1.000	1.000	1.000	1.000	0.001	0.383
	Hybrid	1.002	1.002	1.002	1.002	1.002	0.839	0.556
Normal	Equi-Hist	1.005	1.000	1.000	1.002	1.000	134.636	108.392
	Equi-Width	1.005	1.000	1.000	1.000	1.000	0.003	12.106
	Hybrid	1.001	1.000	1.001	1.001	1.001	1.225	14.947
Sparse Cluster	Equi-Hist	1.014	1.001	1.009	1.034	1.004	65.804	52.073
	Equi-Width	1.009	1.000	1.000	1.025	1.000	0.003	0.798
	Hybrid	1.055	1.034	1.053	1.148	1.042	0.090	1.046
Uniform	Equi-Hist	1.000	1.000	1.000	1.000	1.000	60.803	48.860
	Equi-Width	1.000	1.000	1.000	1.000	1.000	0.004	5.454
	Hybrid	1.000	1.000	1.000	1.000	1.000	0.404	6.897
Zipf	Equi-Hist	1.003	1.000	1.002	1.009	1.001	96.049	73.253
	Equi-Width	1.001	1.000	1.000	1.001	1.000	0.002	6.796
	Hybrid	1.001	1.000	1.000	1.001	1.000	4.832	10.672

The next experiment is done on the simple dataset with the narrow queries. The change of the workload showed the true performance of algorithms in the buckets. In uniform and anti zipf distributions the performance is good because of the uniformity assumption, while in the normal, zipf and sparse distributions the Q-error degrades. Equi-Width has the worst Q-error showing the need of the approach that will remove with the uniformity assumption. To summarize, the Q-error increased in all approaches, but is still acceptable, having Equi-Width 1.139 on normal and 1.231 on zipf distributions. This showed the need of deep diving inside the bucket and analyzing the intra bucket distribution to understand why Equi-Width performs good. Table 4.3

Table 4.3: Benchmark Results (Dataset: SIMPLE, Workload: NARROW). Error metrics are reported as Symmetric Q-Error.

Distribution	Model	Avg Q-Err	25%	75%	95%	Median	Train (s)	Infer (s)
Anti-Zipf	Equi-Hist	1.005	1.002	1.007	1.016	1.004	3.855	3.844
	Equi-Width	1.005	1.002	1.007	1.016	1.004	0.000	0.099
	Hybrid	1.009	1.001	1.011	1.034	1.004	0.815	0.103
Normal	Equi-Hist	1.173	1.007	1.037	1.121	1.016	89.161	88.944
	Equi-Width	1.139	1.007	1.037	1.122	1.016	0.004	1.620
	Hybrid	1.025	1.006	1.026	1.080	1.013	1.153	1.690
Sparse Cluster	Equi-Hist	5.093	2.328	6.248	14.581	3.632	0.307	0.304
	Equi-Width	14.150	8.111	21.647	21.700	15.173	0.002	0.003
	Hybrid	2.486	1.003	1.475	11.596	1.100	0.102	0.005
Uniform	Equi-Hist	1.019	1.006	1.024	1.054	1.013	38.047	37.989
	Equi-Width	1.018	1.006	1.024	1.053	1.013	0.003	0.954
	Hybrid	1.019	1.006	1.024	1.054	1.013	0.370	0.849
Zipf	Equi-Hist	1.236	1.042	1.269	1.841	1.115	56.430	56.413
	Equi-Width	1.231	1.042	1.259	1.824	1.113	0.002	1.212
	Hybrid	1.045	1.000	1.037	1.175	1.011	3.951	1.189

The next setup complicates the experiments by using the hard dataset with intra bucket distributions. For experimental purposes, we conduct two variations: hard dataset with wide workload and hard dataset with narrow workload. First one is to show that the use wide workload can still affect the results giving the false metrics results that are not true for real environment.

As the first experiment we observe that the introduction of hard dataset with micro distribution adds the complexity but because of the error averaging, q error is still about 1.0 for all methods. Table 4.4

Table 4.4: Benchmark Results (Dataset: HARD, Workload: WIDE). Error metrics are reported as Symmetric Q-Error.

Distribution	Model	Avg Q-Err	25%	75%	95%	Median	Train (s)	Infer (s)
Anti-Zipf	Equi-Hist	1.000	1.000	1.000	1.000	1.000	6.126	4.624
	Equi-Width	1.000	1.000	1.000	1.000	1.000	0.001	0.356
	Hybrid	1.002	1.002	1.002	1.002	1.002	0.841	0.544
Normal	Equi-Hist	1.004	1.000	1.002	1.004	1.001	129.244	104.044
	Equi-Width	1.004	1.000	1.002	1.004	1.001	0.004	11.760
	Hybrid	1.000	1.000	1.000	1.000	1.000	1.798	18.307
Sparse Cluster	Equi-Hist	1.014	1.001	1.009	1.034	1.004	61.892	49.170
	Equi-Width	1.009	1.000	1.000	1.025	1.000	0.003	0.741
	Hybrid	1.013	1.010	1.017	1.021	1.015	0.107	1.028
Uniform	Equi-Hist	1.000	1.000	1.000	1.000	1.000	60.097	48.415
	Equi-Width	1.000	1.000	1.000	1.000	1.000	0.003	5.480
	Hybrid	1.000	1.000	1.000	1.000	1.000	0.373	7.233
Zipf	Equi-Hist	1.005	1.000	1.003	1.010	1.001	94.646	72.615
	Equi-Width	1.002	1.000	1.000	1.006	1.000	0.002	6.698
	Hybrid	1.000	1.000	1.000	1.000	1.000	7.675	11.899

The most valuable experiment in static setup is 4th one. We use the hard dataset with itnra bucket distributions so that Equi-Width cannot use uniformity assumption and we stress test it, also we use narrow queries to remove the averaing false metrics results.

For antizipf and uniform distributions, Equi-Width and hybrid approaches performs the same because hybrid detects uniformity and uses the tier 0 model, while Equi-Hist is learning the actual data distribution and improves the Q-error to 1.004 and 1.014 for these distributions respectively, it is worth to note again the inference time for the Equi-Hist, which is unacceptable for real database scenarios. For normal, zipf and sparse distributions, Equi-Width performance degraded to > 2.2 , and hybrid can not learn the data distribution either having the Q-error > 1.2 . After the analysis of the results, the importance of the data driven workload was emphasized. The 3.0 and 4.0 median Q-error for Equi-Width and hybrid respectively showed that since it is median, and workload is uniformly distributed then almost all queries hit the buckets that are not in the dense region, since the buckets are not in the dense regions then the data sampled can contain 0 values and this results in 3.0 4.0 errors, meaning that Equi-Width and hybrid approaches that are built upon sample reservoir, predicting

1 count, while 50% of the queries result in 3 or 4 counts in the buckets. Table 4.5

Table 4.5: Benchmark Results (Dataset: HARD, Workload: NARROW uniform). The best Median Q-Error for each distribution is highlighted in bold.

Distribution	Model	Avg Q-Err	25%	75%	95%	Median	Train (s)	Infer (s)
Anti-Zipf	Equi-Hist	1.007	1.002	1.008	1.023	1.004	3.826	3.821
	Equi-Width	1.048	1.020	1.070	1.116	1.041	0.003	0.093
	Hybrid	1.049	1.020	1.071	1.119	1.042	0.758	0.098
Normal	Equi-Hist	3.783	1.802	4.330	9.522	2.897	87.417	87.241
	Equi-Width	5.123	1.575	5.104	16.163	2.241	0.004	1.677
	Hybrid	5.699	1.259	3.492	25.061	1.702	4.229	1.700
Sparse Cluster	Equi-Hist	5.093	2.323	6.243	14.630	3.630	0.306	0.304
	Equi-Width	14.160	8.143	21.286	22.279	15.216	0.002	0.003
	Hybrid	7.961	1.059	5.828	9.614	1.223	0.240	0.005
Uniform	Equi-Hist	1.020	1.006	1.025	1.058	1.014	39.115	38.979
	Equi-Width	1.057	1.023	1.082	1.140	1.049	0.003	0.946
	Hybrid	1.057	1.024	1.082	1.140	1.049	1.136	1.013
Zipf	Equi-Hist	3.691	1.082	1.773	3.259	1.292	164.968	164.855
	Equi-Width	8.361	1.886	7.000	25.000	3.000	0.005	0.499
	Hybrid	8.570	2.000	8.000	30.000	4.000	1.447	0.567

The final experiment for the static setup uses the hard dataset with micro distributions, and narrow data driven workload meaning the queries will hit the most heavily populated regions, thus stress testing the models on how they perform in learning the distributions inside these buckets. To start analysis, the antizipf and uniform distributions results are still the same, Equi-Width and hybrid perform the same, while Equi-Hist is learning the distribution but it takes more time.

For normal distribution we observe that hybrid reaches the best median Q-error 1.104 while Equi-Width can not capture the hard intra bucket distribution, Equi-Hist is learning but cannot reach better Q-error because of the uniformity assumption. It is important to note that hybrid managed to achieve better Q-error, while the build time increased to 4.2 seconds. The inference time is the same because of the optimized inference approach in the Hybrid FD CDF model.

For zipf and sparse distributions we can observe the same picture that the Hybrid FD CDF model performs better, taking more build time but keeping the same inference time as Equi-Width.

Table 4.6

Table 4.6: Benchmark Results (Dataset: HARD, Workload: NARROW data driven). Best Median Q-Error per distribution is highlighted in bold.

Distribution	Model	Avg Q-Err	25%	75%	95%	Median	Train (s)	Infer (s)
Anti-Zipf	Equi-Hist	1.001	1.000	1.001	1.005	1.001	38.946	38.626
	Equi-Width	1.033	1.012	1.047	1.084	1.026	0.003	1.015
	Hybrid	1.033	1.012	1.047	1.085	1.026	0.694	0.976
Normal	Equi-Hist	1.588	1.142	1.741	2.528	1.359	86.433	86.213
	Equi-Width	1.620	1.157	1.829	2.585	1.410	0.004	1.716
	Hybrid	1.244	1.045	1.221	1.671	1.104	4.226	1.720
Sparse Cluster	Equi-Hist	2.108	1.195	2.319	5.704	1.469	40.249	40.062
	Equi-Width	6.318	2.880	7.577	20.551	4.223	0.003	0.208
	Hybrid	2.300	1.019	1.568	5.206	1.061	0.259	0.225
Uniform	Equi-Hist	1.006	1.002	1.007	1.018	1.004	37.533	37.398
	Equi-Width	1.052	1.021	1.075	1.123	1.044	0.003	0.874
	Hybrid	1.052	1.022	1.075	1.123	1.044	1.147	0.839
Zipf	Equi-Hist	503.907	1.090	1.984	11.447	1.250	187.867	187.551
	Equi-Width	2.988	1.001	2.059	11.126	1.045	0.005	0.618
	Hybrid	1.049	1.000	1.001	1.017	1.001	1.238	0.655

DSB Benchmark

The further experiments on DSB Benchmark illustrate that, hybrid has the lowest median and average Q-errors across the 3 datasets. For the P95 Q-error, in `cr_item_sk` the Equi-Width has 2.081 error while Hybrid maintains 1.038. This assumes that 2x estimation in a production database would be enough to result in a not efficient join order. For the columns of temporal keys `cr_returned_time_sk`, it confirms that all methods work well of flat data with low cardinality. The hybrid manages to achieve 1.0000 median Q-error in that dataset. The proposed Equi-Hist approach that updates the counts after each query, results in 9 seconds inference time while hybrid manages to inference in 0.3 seconds and giving better results, the training time that is more than Equi-Width histogram is small price to pay for that decrease of Q-error.

Table 4.7: Benchmark Results: Decision Support Benchmark (DSB)

Dataset	Model	Avg Q-Err	25%	75%	95%	Median	Train (s)	Infer (s)
ws_item_sk	Equi-Width	1.170	1.010	1.091	1.900	1.024	0.002	0.339
	Equi-Hist	1.167	1.020	1.176	1.725	1.055	9.596	9.271
	Hybrid	1.020	1.006	1.026	1.057	1.014	0.200	0.376
cr_item_sk	Equi-Hist	1.183	1.041	1.211	1.634	1.100	6.774	6.388
	Equi-Width	1.186	1.009	1.171	2.081	1.021	0.002	0.239
	Hybrid	1.026	1.005	1.016	1.038	1.010	0.164	0.278
cr_ret_time_sk	Equi-Hist	1.005	1.002	1.007	1.013	1.004	7.950	7.568
	Equi-Width	1.014	1.005	1.017	1.040	1.011	0.003	0.283
	Hybrid	1.014	1.006	1.017	1.039	1.000	0.221	0.300

The static benchmark on DSB columns (Table 4.7) shows that the Hybrid estimator has the lowest Median and Average Q-Errors. The most significant hard experiment in this dataset is `cr_item_sk` column where Equi-Width Histograms got a P95 Q-error of 2.081, while the Hybrid model was able to maintain accuracy 1.038. In production systems, 2x estimation error is usually the point where sub-optimal join orders get pushed; the Hybrid model manages to stay away from this. In addition, while Equi-Hist is accurate, its inference latency (≈ 9 seconds) is 10x slower than the Hybrid method (≈ 0.3 seconds).

4.7.2 Static setup query one-by-one inference

Synthetic data

Table 4.8: Benchmark Results (Dataset: SIMPLE, Workload: WIDE, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.

Distribution	Model	Avg Q	25% Q	75% Q	95% Q	Med Q	Train (s)	Tot Inf (s)	Med Inf (μs)	Avg Inf (μs)	P95 Inf (μs)
Anti-Zipf	Equi-Hist	1.000	1.000	1.000	1.001	1.000	6.521	4.125	40.17	41.29	48.38
	Equi-Width	1.003	1.000	1.002	1.015	1.001	0.003	0.138	1.29	1.32	1.50
	Hybrid	1.004	1.000	1.002	1.015	1.001	0.700	3.974	39.83	39.69	73.58
Normal	Equi-Hist	1.004	1.000	1.002	1.005	1.001	135.770	85.132	85.75	85.13	96.21
	Equi-Width	1.015	1.001	1.009	1.071	1.004	0.004	1.607	1.33	1.53	1.71
	Hybrid	1.012	1.001	1.008	1.070	1.004	3.968	73.361	66.75	73.28	150.71
Sparse Cluster	Equi-Hist	1.014	1.001	1.009	1.034	1.004	64.465	41.314	41.54	42.35	48.21
	Equi-Width	1.013	1.001	1.009	1.027	1.004	0.002	1.460	1.33	1.43	1.67
	Hybrid	1.006	1.001	1.008	1.017	1.003	0.249	12.695	11.33	12.93	29.46
Uniform	Equi-Hist	1.000	1.000	1.000	1.000	1.000	60.097	40.467	39.50	40.47	46.63
	Equi-Width	1.006	1.001	1.005	1.025	1.002	0.003	1.483	1.33	1.41	1.71
	Hybrid	1.006	1.001	1.005	1.025	1.002	1.137	28.630	25.92	28.55	59.83
Zipf	Equi-Hist	1.009	1.001	1.005	1.021	1.002	305.930	181.214	181.04	181.21	204.00
	Equi-Width	1.158	1.035	1.140	1.555	1.061	0.005	1.533	1.33	1.46	1.67
	Hybrid	1.152	1.035	1.140	1.547	1.061	1.264	46.378	39.92	46.29	109.50

The one by one inference results in near perfect accuracy metrics across all models mirroring the results of batch inference because of the error averaging effect of wide queries. The evaluation without vectorization causes inference to scale linearly.

Table 4.9: Benchmark Results (Dataset: SIMPLE, Workload: NARROW, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.

Distribution	Model	Avg Q	25% Q	75% Q	95% Q	Med Q	Train (s)	Tot Inf (s)	Med Inf (μs)	Avg Inf (μs)	P95 Inf (μs)
Anti-Zipf	Equi-Hist	1.007	1.002	1.008	1.023	1.004	3.927	4.044	39.42	40.44	45.88
	Equi-Width	1.048	1.020	1.070	1.116	1.041	0.003	0.104	0.88	0.98	1.42
	Hybrid	1.048	1.020	1.070	1.117	1.041	0.749	0.149	1.25	1.42	1.92
Normal	Equi-Hist	3.783	1.802	4.330	9.522	2.897	87.555	83.673	83.75	83.83	94.71
	Equi-Width	5.123	1.575	5.104	16.163	2.241	0.004	0.941	0.83	0.87	1.13
	Hybrid	4.968	1.313	3.673	17.463	1.813	4.010	13.351	2.50	13.29	44.21
Sparse Cluster	Equi-Hist	5.093	2.323	6.243	14.630	3.630	0.291	0.310	41.46	42.13	49.88
	Equi-Width	14.160	8.143	21.286	22.279	15.216	0.002	0.007	0.92	0.93	1.08
	Hybrid	7.145	1.108	6.995	10.190	2.629	0.261	0.147	2.79	19.89	48.13
Uniform	Equi-Hist	1.020	1.006	1.025	1.058	1.014	39.134	40.282	39.33	40.28	46.04
	Equi-Width	1.057	1.023	1.082	1.140	1.049	0.003	0.992	0.83	0.92	1.08
	Hybrid	1.057	1.023	1.082	1.140	1.049	1.110	1.431	1.25	1.35	1.63
Zipf	Equi-Hist	3.691	1.082	1.773	3.259	1.292	165.781	153.170	182.96	181.74	201.88
	Equi-Width	8.361	1.886	7.000	25.000	3.000	0.005	0.851	0.83	0.93	1.38
	Hybrid	8.356	2.000	8.000	29.000	4.000	1.173	4.669	0.71	5.46	42.96

With the narrow workload the Q-Error metrics reproduce the previous narrow experiment, however the inference latency highlights the computational overhead of sequential execution of advanced models.

Table 4.10: Benchmark Results (Dataset: HARD, Workload: WIDE, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.

Distribution	Model	Avg Q	25% Q	75% Q	95% Q	Med Q	Train (s)	Tot Inf (s)	Med Inf (μs)	Avg Inf (μs)	P95 Inf (μs)
Anti-Zipf	Equi-Hist	1.000	1.000	1.000	1.001	1.000	6.472	4.146	40.46	41.50	47.33
	Equi-Width	1.003	1.000	1.002	1.015	1.001	0.003	0.142	1.29	1.35	1.50
	Hybrid	1.004	1.000	1.002	1.015	1.001	0.696	4.159	41.17	41.54	78.79
Normal	Equi-Hist	1.004	1.000	1.002	1.005	1.001	135.067	83.524	81.50	83.52	96.92
	Equi-Width	1.015	1.001	1.009	1.071	1.004	0.004	1.532	1.33	1.46	1.67
	Hybrid	1.012	1.001	1.008	1.070	1.004	4.242	75.732	68.71	75.64	155.25
Sparse Cluster	Equi-Hist	1.014	1.001	1.009	1.034	1.004	64.534	42.661	42.13	43.73	50.67
	Equi-Width	1.013	1.001	1.009	1.027	1.004	0.002	1.514	1.33	1.48	1.71
	Hybrid	1.006	1.001	1.008	1.017	1.003	0.232	13.129	11.67	13.38	30.38
Uniform	Equi-Hist	1.000	1.000	1.000	1.000	1.000	60.675	41.126	39.92	41.13	48.00
	Equi-Width	1.006	1.001	1.005	1.025	1.002	0.003	1.461	1.29	1.39	1.67
	Hybrid	1.006	1.001	1.005	1.025	1.002	1.152	29.274	26.42	29.19	61.25
Zipf	Equi-Hist	1.009	1.001	1.005	1.021	1.002	305.924	182.436	177.92	182.44	212.33
	Equi-Width	1.158	1.035	1.140	1.555	1.061	0.005	1.532	1.33	1.46	1.71
	Hybrid	1.152	1.035	1.140	1.547	1.061	1.212	46.537	39.63	46.44	110.83

Micro distributions in the hard dataset are masked by the error averaging effect of the wide workload which results in low Q-Errors across all approaches. The inference latency is high compared to the batch inference.

Table 4.11: Benchmark Results (Dataset: HARD, Workload: NARROW, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.

Distribution	Model	Avg Q	25% Q	75% Q	95% Q	Med Q	Train (s)	Tot Inf (s)	Med Inf (μs)	Avg Inf (μs)	P95 Inf (μs)
Anti-Zipf	Equi-Hist	1.007	1.002	1.008	1.023	1.004	3.866	4.014	39.25	40.13	45.88
	Equi-Width	1.048	1.020	1.070	1.116	1.041	0.003	0.102	0.88	0.96	1.42
	Hybrid	1.048	1.020	1.070	1.117	1.041	0.678	0.158	1.33	1.50	2.17
Normal	Equi-Hist	3.783	1.802	4.330	9.522	2.897	86.962	82.568	81.29	82.72	93.92
	Equi-Width	5.123	1.575	5.104	16.163	2.241	0.004	1.033	0.88	0.96	1.38
	Hybrid	4.935	1.313	3.658	17.274	1.808	3.937	13.092	2.46	13.02	44.08
Sparse Cluster	Equi-Hist	5.093	2.323	6.243	14.630	3.630	0.294	0.307	41.29	41.71	46.96
	Equi-Width	14.160	8.143	21.286	22.279	15.216	0.002	0.007	0.92	0.92	1.08
	Hybrid	7.640	1.095	7.187	10.211	2.193	0.237	0.157	2.92	21.24	48.79
Uniform	Equi-Hist	1.020	1.006	1.025	1.058	1.014	38.448	39.263	38.38	39.26	44.67
	Equi-Width	1.057	1.023	1.082	1.140	1.049	0.003	0.994	0.83	0.92	1.08
	Hybrid	1.057	1.023	1.082	1.140	1.049	1.126	1.447	1.25	1.37	1.67
Zipf	Equi-Hist	3.691	1.082	1.773	3.259	1.292	162.577	151.398	176.88	179.64	199.96
	Equi-Width	8.361	1.886	7.000	25.000	3.000	0.004	0.849	0.83	0.93	1.42
	Hybrid	8.351	2.000	8.000	29.000	4.000	1.180	4.547	0.75	5.32	42.21

Evaluation of narrow queries with micro distributions eliminates the error averaging effect, and proves the Hybrid model advantage in capturing the non uniform local distributions, it achieves the best median Q-Error on normal and Sparse distributions. Still without the batch vectorization, the sequential inference results in higher inference of Hybrid FD CDF model.

Table 4.12: Benchmark Results (Dataset: HARD, Workload: NARROW data driven, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.

Distribution	Model	Avg Q	25% Q	75% Q	95% Q	Med Q	Train (s)	Tot Inf (s)	Med Inf (μs)	Avg Inf (μs)	P95 Inf (μs)
Anti-Zipf	Equi-Hist	1.001	1.000	1.001	1.005	1.001	40.000	40.693	40.17	40.69	46.50
	Equi-Width	1.033	1.012	1.047	1.084	1.026	0.003	1.499	1.33	1.42	1.71
	Hybrid	1.033	1.012	1.047	1.084	1.026	0.695	2.137	1.96	2.05	2.75
Normal	Equi-Hist	1.588	1.142	1.741	2.528	1.359	87.204	84.212	83.92	84.21	94.42
	Equi-Width	1.620	1.157	1.829	2.585	1.410	0.004	1.355	1.08	1.28	1.71
	Hybrid	1.292	1.054	1.291	1.895	1.131	4.012	11.664	2.75	11.58	37.38
Sparse Cluster	Equi-Hist	2.108	1.195	2.319	5.704	1.469	42.388	44.877	44.17	44.88	50.96
	Equi-Width	6.318	2.880	7.577	20.551	4.223	0.002	1.125	0.88	1.05	1.50
	Hybrid	2.432	1.037	2.000	6.491	1.315	0.263	16.513	2.46	16.42	47.46
Uniform	Equi-Hist	1.006	1.002	1.007	1.018	1.004	40.339	40.910	40.33	40.91	46.71
	Equi-Width	1.052	1.021	1.075	1.123	1.044	0.003	1.100	0.88	1.03	1.46
	Hybrid	1.052	1.021	1.075	1.123	1.044	1.166	1.577	1.29	1.50	2.21
Zipf	Equi-Hist	503.907	1.090	1.984	11.447	1.250	199.347	190.532	195.63	190.53	210.83
	Equi-Width	2.988	1.001	2.059	11.126	1.045	0.004	1.252	1.21	1.18	1.58
	Hybrid	1.051	1.000	1.001	1.019	1.001	1.195	10.512	14.42	10.42	23.08

Data driven narrow workload forces the queries to be in the dense regions, complex micro distributions, in that regions Hybrid FD CDF model achieves the highest accuracy across all datasets.

DSB Benchmark

Table 4.13: Benchmark Results (Dataset: DSB Benchmark, One-by-One Inference). Inference times per query are reported in microseconds (μs). Best Median Q-Error per distribution is highlighted in bold.

Dataset	Model	Avg Q	25% Q	75% Q	95% Q	Med Q	Train (s)	Tot Inf (s)	Med Inf (μs)	Avg Inf (μs)	P95 Inf (μs)
cr_item_sk	Equi-Hist	1.183	1.041	1.211	1.634	1.100	7.130	9.837	9.08	9.84	13.54
	Equi-Width	1.186	1.009	1.171	2.081	1.021	0.001	1.512	1.29	1.44	1.63
	Hybrid	1.020	1.005	1.019	1.081	1.011	0.154	7.799	2.33	7.72	22.21
cr_ret_time_sk	Equi-Hist	1.005	1.002	1.007	1.013	1.004	8.486	11.339	10.50	11.43	15.17
	Equi-Width	1.014	1.005	1.017	1.040	1.011	0.002	1.408	1.25	1.35	1.58
	Hybrid	1.014	1.005	1.017	1.040	1.011	0.207	2.230	2.00	2.17	2.58
ws_item_sk	Equi-Hist	1.167	1.020	1.176	1.725	1.055	10.077	13.251	12.42	13.34	17.29
	Equi-Width	1.170	1.010	1.091	1.900	1.024	0.003	1.414	1.25	1.35	1.63
	Hybrid	1.020	1.006	1.027	1.062	1.014	0.204	7.303	2.13	7.27	22.13

In the DSB benchmark, Hybrid model prevents the high P95 Q-Error which happens in Equi-Width, the error is below 1.09 compared to Equi-Wodth 2.08 in cr_item_sk.

4.7.3 Dynamic setup batch inference

Synthetic data

For the dynamic setup, we use data driven workload, hard normal and hard zipf datasets. The experiments will be using different drift setups, the updates behavior and the additive manner. The goal here to to check whether the hybrid model can achieve better Q-error than Equi-Width histogram and Equi-Hist, while keeping the inference and build time the same.

First dynamic experiment is done using 10% step which is 600.000 data changes, the data is changed from normal to zipf distribution, while the workload remains the same as for normal distribution.

Table 4.14: 10% step, normal $\rightarrow$ zipf, workload normal $\rightarrow$ normal

Drift	Model	Median	P95	Avg	Time	Count
0 %	Equi-Hist	2.896	9.510	3.762	174.632	-
	Equi-Width	2.229	15.908	4.819	1.669	-
	Hybrid_FT	1.106	1.681	1.340	0.000	0
	Hybrid_RB	1.106	1.681	1.340	0.000	0
	Hybrid_Shock	1.106	1.681	1.340	1.656	-
10 %	Equi-Hist	2.806	9.080	3.595	173.896	-
	Equi-Width	2.122	16.754	4.887	1.708	-

Continued on next page

Table 4.14 – Continued from previous page

Drift	Model	Median	P95	Avg	Time	Count
	Hybrid_FT	1.096	1.679	1.321	0.609	873
	Hybrid_RB	1.096	1.679	1.321	0.682	147
	Hybrid_Shock	1.108	1.712	1.382	1.647	-
20 %	Equi-Hist	2.755	8.829	3.515	180.463	-
	Equi-Width	2.037	18.557	5.217	1.700	-
	Hybrid_FT	1.097	1.664	1.305	0.611	873
	Hybrid_RB	1.097	1.664	1.311	0.728	151
	Hybrid_Shock	1.108	1.692	1.335	1.618	-
30 %	Equi-Hist	2.729	8.755	3.491	173.742	-
	Equi-Width	2.008	21.204	5.768	1.679	-
	Hybrid_FT	1.103	1.680	1.309	0.624	873
	Hybrid_RB	1.103	1.681	1.324	0.651	154
	Hybrid_Shock	1.117	1.713	1.433	1.616	-
40 %	Equi-Hist	2.687	8.504	3.412	173.543	-
	Equi-Width	2.149	23.986	6.394	1.752	-
	Hybrid_FT	1.106	1.699	1.323	0.654	873
	Hybrid_RB	1.106	1.705	1.353	0.776	163
	Hybrid_Shock	1.119	1.738	1.365	1.705	-
50 %	Equi-Hist	2.742	9.970	3.829	173.966	-
	Equi-Width	2.487	34.515	9.071	1.696	-
	Hybrid_FT	1.109	1.720	1.339	0.683	873
	Hybrid_RB	1.108	1.722	1.369	0.810	176
	Hybrid_Shock	1.124	1.765	1.486	1.692	-
60 %	Equi-Hist	2.688	9.388	3.638	173.587	-
	Equi-Width	2.978	40.340	10.254	1.736	-
	Hybrid_FT	1.115	1.755	1.365	0.728	873
	Hybrid_RB	1.114	1.765	1.404	0.803	192
	Hybrid_Shock	1.130	1.808	1.605	1.785	-
70 %	Equi-Hist	2.610	8.710	3.446	174.040	-
	Equi-Width	3.765	48.442	12.271	1.737	-
	Hybrid_FT	1.127	1.813	1.398	0.796	873
	Hybrid_RB	1.126	1.860	1.482	1.155	245
	Hybrid_Shock	1.146	1.885	1.671	1.850	-
80 %	Equi-Hist	2.614	9.337	3.631	175.319	-
	Equi-Width	5.441	78.758	19.696	1.714	-
	Hybrid_FT	1.138	1.945	1.476	0.862	873
	Hybrid_RB	1.137	2.058	1.740	1.299	283
	Hybrid_Shock	1.157	2.044	1.924	1.691	-
90 %	Equi-Hist	2.587	10.552	3.987	175.865	-
	Equi-Width	10.574	185.864	45.495	1.746	-
	Hybrid_FT	1.175	2.182	1.660	1.058	873
	Hybrid_RB	1.173	2.512	2.055	1.597	375
	Hybrid_Shock	1.201	2.321	3.674	1.694	-
100 %	Equi-Hist	1.240	3.111	4.020	175.139	-

Continued on next page

Table 4.14 – Continued from previous page

Drift	Model	Median	P95	Avg	Time	Count
	Equi-Width	547.906	3858.639	1081.844	1.683	-
	Hybrid_FT	127.177	355.648	191.641	2.029	873
	Hybrid_RB	6.732	13.265	9.967	2.779	841
	Hybrid_Shock	21.000	89.000	43.289	0.458	-

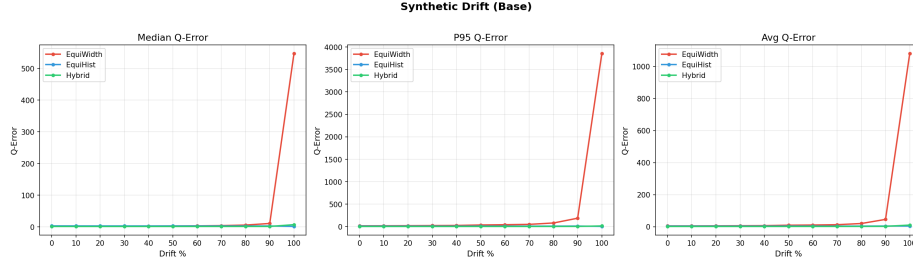


Figure 4-6: 10% step, normal -> zipf, workload normal -> normal

The Equi-Hist takes the most time for the training+inference which is 174 seconds for 1.000.000 queries, this is not acceptable regardless the Q-error, the shock metrics that are included only for the experimental purposes to observe how metrics degrade after data change, show that the median Q-error gets worse after each each step, but still maintaining good Q-error at 1.1, except the last step at which the remaining 10% of the zipf which are at the most left bucket are inserted. Then the median Q-error 1.173 after the rebuild at 90% step is increased to 21% hybrid shock median Q-error, then the finetune did not manage to achieve better results by getting the error worse to 127, however the rebuild mechanism successfully returned the median Q-error to 6.732, while Equi-Width performed with median Q-error at 547, and Equi-Hist successfully managed to achieved 1.240 however taking much more time than other method.

The metrics, finetune and rebuild bucket count is observed to be 873 and 150 respectively, meaning almost all buckets are finetune because the step of 10% is a lot of data and it touches all buckets. The need in lower step size is emerged and in the next experiment we decrease it to 5%, so that the overall steps count is 21 giving the hybrid model more checkpoints at which it can finetune lower number of buckets and this triggering less rebuild operations supposedly.

Next experiment is with the same parameters except the step size is changed to 5%.

Table 4.15: 5% step, normal -> zipf, workload normal -> normal

Drift	Model	Median	P95	Avg	Time	Count
0 %	Equi-Hist	1.359	2.529	1.589	169.476	—
	Equi-Width	1.415	2.602	1.642	1.777	—
	Hybrid_Shock	1.107	1.701	1.358	1.709	—
	Hybrid_FT	1.107	1.701	1.358	0.000	0
	Hybrid_RB	1.107	1.701	1.358	0.000	0
5 %	Equi-Hist	1.359	2.533	1.589	168.635	—
	Equi-Width	1.382	2.552	1.631	1.773	—
	Hybrid_Shock	1.109	1.711	1.344	1.779	—
	Hybrid_FT	1.097	1.685	1.327	0.582	873
	Hybrid_RB	1.097	1.675	1.328	0.739	151
10 %	Equi-Hist	1.359	2.535	1.597	169.456	—
	Equi-Width	1.355	2.527	1.632	1.830	—
	Hybrid_Shock	1.107	1.716	1.387	1.605	—
	Hybrid_FT	1.097	1.686	1.313	0.600	873
	Hybrid_RB	1.097	1.686	1.319	0.664	139
15 %	Equi-Hist	1.359	2.535	1.605	169.567	—
	Equi-Width	1.339	2.567	1.641	1.775	—
	Hybrid_Shock	1.106	1.705	1.412	1.626	—
	Hybrid_FT	1.094	1.677	1.312	0.585	873
	Hybrid_RB	1.094	1.677	1.322	0.620	142
20 %	Equi-Hist	1.359	2.535	1.591	169.635	—
	Equi-Width	1.334	2.690	1.663	1.842	—
	Hybrid_Shock	1.110	1.716	1.336	1.651	—
	Hybrid_FT	1.098	1.686	1.318	0.602	873
	Hybrid_RB	1.098	1.688	1.323	0.680	132
25 %	Equi-Hist	1.358	2.534	1.600	169.846	—
	Equi-Width	1.338	2.857	1.698	1.648	—
	Hybrid_Shock	1.115	1.719	1.385	1.641	—
	Hybrid_FT	1.102	1.686	1.320	0.610	872
	Hybrid_RB	1.102	1.687	1.333	0.695	130
30 %	Equi-Hist	1.358	2.534	1.599	168.445	—
	Equi-Width	1.349	3.055	1.750	1.785	—
	Hybrid_Shock	1.117	1.728	1.373	1.717	—
	Hybrid_FT	1.104	1.694	1.328	0.615	873
	Hybrid_RB	1.104	1.699	1.338	0.813	155
35 %	Equi-Hist	1.358	2.533	1.592	169.441	—
	Equi-Width	1.370	3.281	1.821	1.735	—
	Hybrid_Shock	1.115	1.752	1.364	1.786	—
	Hybrid_FT	1.103	1.711	1.337	0.641	873
	Hybrid_RB	1.103	1.722	1.356	0.923	176
40 %	Equi-Hist	1.358	2.533	1.601	168.376	—

Continued on next page

Table 4.15 – Continued from previous page

Drift	Model	Median	P95	Avg	Time	Count
	Equi-Width	1.410	3.549	1.916	1.729	–
	Hybrid_Shock	1.118	1.758	1.439	1.703	–
	Hybrid_FT	1.103	1.721	1.342	0.653	872
	Hybrid_RB	1.103	1.716	1.361	0.737	173
45 %	Equi-Hist	1.357	2.530	1.592	169.242	–
	Equi-Width	1.481	3.863	2.040	1.777	–
	Hybrid_Shock	1.120	1.769	1.390	1.778	–
	Hybrid_FT	1.105	1.727	1.350	0.649	873
	Hybrid_RB	1.105	1.731	1.370	0.860	175
50 %	Equi-Hist	1.358	2.534	1.599	165.027	–
	Equi-Width	1.613	4.258	2.210	1.772	–
	Hybrid_Shock	1.121	1.775	1.411	1.686	–
	Hybrid_FT	1.106	1.730	1.357	0.659	873
	Hybrid_RB	1.106	1.727	1.367	0.850	188
55 %	Equi-Hist	1.358	2.537	1.603	165.478	–
	Equi-Width	1.791	4.736	2.429	1.682	–
	Hybrid_Shock	1.131	1.794	1.486	1.674	–
	Hybrid_FT	1.114	1.743	1.364	0.709	873
	Hybrid_RB	1.114	1.759	1.404	0.817	196
60 %	Equi-Hist	1.357	2.531	1.599	166.134	–
	Equi-Width	2.011	5.309	2.704	1.673	–
	Hybrid_Shock	1.127	1.832	1.473	1.653	–
	Hybrid_FT	1.112	1.775	1.392	0.700	873
	Hybrid_RB	1.111	1.777	1.432	0.981	223
65 %	Equi-Hist	1.357	2.532	1.613	165.850	–
	Equi-Width	2.297	6.061	3.069	1.681	–
	Hybrid_Shock	1.134	1.866	1.714	1.669	–
	Hybrid_FT	1.118	1.808	1.409	0.738	873
	Hybrid_RB	1.118	1.835	1.451	0.983	224
70 %	Equi-Hist	1.356	2.522	1.596	166.004	–
	Equi-Width	2.671	7.014	3.535	1.667	–
	Hybrid_Shock	1.142	1.907	1.612	1.655	–
	Hybrid_FT	1.124	1.842	1.437	0.763	873
	Hybrid_RB	1.123	1.862	1.498	1.174	240
75 %	Equi-Hist	1.357	2.532	1.626	165.450	–
	Equi-Width	3.209	8.455	4.226	1.694	–
	Hybrid_Shock	1.151	1.956	2.023	1.785	–
	Hybrid_FT	1.132	1.886	1.478	0.819	873
	Hybrid_RB	1.131	1.927	1.565	1.212	272
80 %	Equi-Hist	1.356	2.530	1.625	164.631	–
	Equi-Width	4.003	10.525	5.212	1.759	–
	Hybrid_Shock	1.158	2.075	2.526	1.645	–
	Hybrid_FT	1.137	1.984	1.560	0.853	873
	Hybrid_RB	1.136	2.052	1.667	1.320	302
85 %	Equi-Hist	1.355	2.530	1.642	164.180	–

Continued on next page

Table 4.15 – Continued from previous page

Drift	Model	Median	P95	Avg	Time	Count
	Equi-Width	5.316	13.923	6.845	1.801	–
	Hybrid_Shock	1.184	2.237	2.644	1.591	–
	Hybrid_FT	1.159	2.117	1.651	0.987	872
	Hybrid_RB	1.157	2.278	1.903	1.722	352
90 %	Equi-Hist	1.357	2.552	1.657	166.169	–
	Equi-Width	8.001	21.063	10.259	1.803	–
	Hybrid_Shock	1.212	2.734	3.397	1.855	–
	Hybrid_FT	1.184	2.508	1.813	1.106	872
	Hybrid_RB	1.181	2.768	2.179	2.303	415
95 %	Equi-Hist	1.352	2.533	1.702	166.542	–
	Equi-Width	15.701	40.976	19.960	1.807	–
	Hybrid_Shock	1.306	4.748	7.347	1.778	–
	Hybrid_FT	1.263	3.895	2.406	1.383	872
	Hybrid_RB	1.262	10.370	3.549	2.717	553
100 %	Equi-Hist	1.096	1.721	1.613	166.484	–
	Equi-Width	1685.737	3406.806	1718.994	1.775	–
	Hybrid_Shock	22.000	90.000	43.967	0.456	–
	Hybrid_FT	128.770	369.896	196.041	1.954	872
	Hybrid_RB	6.843	13.386	10.071	2.663	839

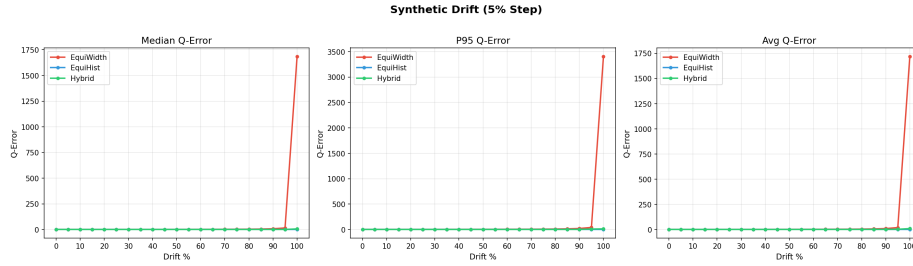


Figure 4-7: 5% step, normal -> zipf, workload normal -> normal

The experiment with 5% step supposedly was to show that the number of buckets finetuned or rebuilt will get lower since the data change has become 300.000 rows instead of 600.000, but analyzing the results we can see that 300.000 data changes are enough to trigger finetune process in all buckets. Also, lower step size gives the hybrid model additional checkpoints at which it can run the finetune and rebuild processes so as the results we observe not the big metrics drop after each data drift, except the last one at which the 5% of the zipf distribution are changed. The Q-error metrics are the same as in the experiment before. The high Q-error is explained that the workload does not change and we query the data that is not present at the final

step, this arise the need to change the workload from normal to zipf at each step with the percentage of workload query changes.

The next goal is to check the performance when the workload follows the pattern of queries that hit the data that exists in the database along with the data drift.

Table 4.16: 5% step, normal -> zipf, workload normal -> zipf

Drift	Model	Median	P95	Avg	Time	Count
0 %	Equi-Hist	1.360	2.547	1.591	168.004	–
	Equi-Width	1.409	2.590	1.621	1.736	–
	Hybrid_Shock	1.106	1.666	1.247	1.644	–
	Hybrid_FT	1.106	1.666	1.247	0.000	0
	Hybrid_RB	1.106	1.666	1.247	0.000	0
5 %	Equi-Hist	1.368	2.804	2.254	164.776	–
	Equi-Width	1.415	12.333	10.423	1.753	–
	Hybrid_Shock	1.110	2.062	4.600	1.748	–
	Hybrid_FT	1.101	1.991	3.993	0.626	873
	Hybrid_RB	1.089	1.631	1.273	0.599	137
10 %	Equi-Hist	1.373	3.126	2.882	166.202	–
	Equi-Width	1.420	143.049	36.182	1.705	–
	Hybrid_Shock	1.123	1.645	1.311	1.605	–
	Hybrid_FT	1.111	1.620	1.296	0.702	873
	Hybrid_RB	1.110	1.614	1.275	0.653	139
15 %	Equi-Hist	1.382	3.636	4.665	168.458	–
	Equi-Width	1.434	313.872	80.566	1.709	–
	Hybrid_Shock	1.131	1.629	1.309	1.544	–
	Hybrid_FT	1.118	1.603	1.292	0.748	873
	Hybrid_RB	1.118	1.597	1.271	0.640	138
20 %	Equi-Hist	1.389	4.137	2.349	165.949	–
	Equi-Width	1.457	539.103	138.760	1.635	–
	Hybrid_Shock	1.149	1.615	1.324	1.598	–
	Hybrid_FT	1.134	1.589	1.303	0.848	873
	Hybrid_RB	1.134	1.587	1.289	0.626	129
25 %	Equi-Hist	1.397	4.784	2.630	165.142	–
	Equi-Width	1.497	829.578	217.715	1.757	–
	Hybrid_Shock	1.171	1.606	1.349	1.571	–
	Hybrid_FT	1.154	1.580	1.326	0.909	873
	Hybrid_RB	1.154	1.576	1.311	0.605	127
30 %	Equi-Hist	1.404	5.387	6.611	171.035	–
	Equi-Width	1.568	1179.448	312.821	1.639	–
	Hybrid_Shock	1.199	1.591	1.381	1.553	–
	Hybrid_FT	1.176	1.566	1.356	0.989	873
	Hybrid_RB	1.176	1.562	1.336	0.617	144
35 %	Equi-Hist	1.412	6.090	8.231	163.268	–
	Equi-Width	1.709	1591.129	426.216	1.756	–

Continued on next page

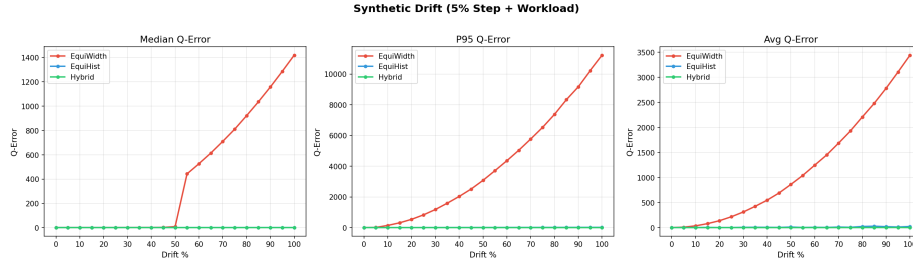
Table 4.16 – Continued from previous page

Drift	Model	Median	P95	Avg	Time	Count
	Hybrid_Shock	1.213	1.580	1.346	1.600	–
	Hybrid_FT	1.173	1.552	1.319	1.028	873
	Hybrid_RB	1.173	1.550	1.303	0.720	157
40 %	Equi-Hist	1.419	6.681	6.674	165.847	–
	Equi-Width	2.003	2030.526	548.631	1.634	–
	Hybrid_Shock	1.228	1.570	1.409	1.597	–
	Hybrid_FT	1.185	1.543	1.374	1.076	873
	Hybrid_RB	1.185	1.537	1.353	0.722	172
45 %	Equi-Hist	1.428	7.358	4.677	165.837	–
	Equi-Width	2.768	2512.616	691.687	1.703	–
	Hybrid_Shock	1.231	1.543	1.477	1.560	–
	Hybrid_FT	1.189	1.518	1.436	1.149	873
	Hybrid_RB	1.189	1.513	1.415	0.745	161
50 %	Equi-Hist	1.439	8.287	12.608	166.460	–
	Equi-Width	9.333	3077.905	863.072	1.757	–
	Hybrid_Shock	1.226	1.529	1.467	1.561	–
	Hybrid_FT	1.184	1.503	1.415	1.202	872
	Hybrid_RB	1.184	1.500	1.393	0.777	169
55 %	Equi-Hist	1.447	8.961	3.988	167.587	–
	Equi-Width	443.249	3707.654	1043.413	1.650	–
	Hybrid_Shock	1.230	1.520	1.435	1.566	–
	Hybrid_FT	1.188	1.494	1.381	1.264	872
	Hybrid_RB	1.188	1.489	1.362	0.772	185
60 %	Equi-Hist	1.456	9.812	6.851	169.936	–
	Equi-Width	526.509	4356.086	1247.702	1.736	–
	Hybrid_Shock	1.228	1.493	1.694	1.527	–
	Hybrid_FT	1.187	1.467	1.595	1.309	871
	Hybrid_RB	1.187	1.460	1.614	0.920	204
65 %	Equi-Hist	1.465	10.591	4.637	175.381	–
	Equi-Width	614.213	5036.269	1451.129	1.686	–
	Hybrid_Shock	1.234	1.489	1.576	1.532	–
	Hybrid_FT	1.192	1.459	1.457	1.398	871
	Hybrid_RB	1.191	1.456	1.273	0.997	217
70 %	Equi-Hist	1.475	11.631	12.176	171.318	–
	Equi-Width	710.195	5769.007	1687.845	1.673	–
	Hybrid_Shock	1.231	1.462	1.469	1.565	–
	Hybrid_FT	1.190	1.435	1.417	1.434	873
	Hybrid_RB	1.190	1.429	1.607	1.022	231
75 %	Equi-Hist	1.484	12.507	7.536	165.002	–
	Equi-Width	810.366	6527.897	1931.980	1.691	–
	Hybrid_Shock	1.239	1.437	1.824	1.505	–
	Hybrid_FT	1.196	1.407	1.620	1.479	868
	Hybrid_RB	1.196	1.404	1.587	1.028	247
80 %	Equi-Hist	1.494	13.169	23.965	164.345	–
	Equi-Width	921.615	7377.437	2209.731	1.593	–

Continued on next page

Table 4.16 – Continued from previous page

Drift	Model	Median	P95	Avg	Time	Count
	Hybrid_Shock	1.237	1.410	2.208	1.472	–
	Hybrid_FT	1.195	1.376	1.900	1.572	867
	Hybrid_RB	1.195	1.369	1.900	1.224	279
85 %	Equi-Hist	1.506	14.188	29.200	163.928	–
	Equi-Width	1036.414	8335.168	2482.004	1.510	–
	Hybrid_Shock	1.238	1.387	2.582	1.439	–
	Hybrid_FT	1.196	1.356	2.204	1.629	857
	Hybrid_RB	1.195	1.345	2.303	1.344	305
90 %	Equi-Hist	1.519	15.135	20.208	165.280	–
	Equi-Width	1157.655	9174.554	2784.533	1.621	–
	Hybrid_Shock	1.239	1.348	3.094	1.555	–
	Hybrid_FT	1.196	1.318	2.643	1.652	846
	Hybrid_RB	1.196	1.312	2.407	1.616	355
95 %	Equi-Hist	1.531	16.147	13.211	164.626	–
	Equi-Width	1286.434	10224.030	3108.611	1.492	–
	Hybrid_Shock	1.241	1.293	3.773	1.478	–
	Hybrid_FT	1.198	1.264	3.059	1.767	816
	Hybrid_RB	1.198	1.261	3.331	1.879	435
100 %	Equi-Hist	1.542	17.010	23.726	163.399	–
	Equi-Width	1419.415	11232.072	3443.789	0.875	–
	Hybrid_Shock	1.240	1.262	41.546	0.482	–
	Hybrid_FT	1.197	1.219	1.244	1.711	338
	Hybrid_RB	1.197	1.219	1.250	0.922	233

Figure 4-8: 5% step, normal $\rightarrow$ zipf, workload normal $\rightarrow$ zipf

Utilizing the workload drift with a data drift with a step size of 5% shows that the hybrid model manages to keep the median, p95 and average Q-error metrics at stable path, after the workload is following the data distribution change, at each step the performance degrades, however the finetune process fixed the bucket counts and rebuild process further advances it. At the initial steps it is worth to note that finetune process time + rebuild time is almost the same at the Equi-Width rebuild time which yields that after the initial training hybrid models supports the ability to

perform better than Equi-Width while taking the same time. As for the Equi-Hist, it manages to learn the distribution and achieves 1.542 median Q-error taking 163 seconds, and Equi-Width is with the median Q-error of 1419.415.

For the next experiment, the ability to be scalable is tested. The supposed results is that as the data increases in size, the hybrid model should still take the same time because of the use the sample reservoir, while maintaining the same performance. As the experiments suggests, the more data points the more error can occur.

Table 4.17: 5% step, normal -> normal + zipf, workload normal -> zipf

Drift	Model	Median	P95	Avg	Time	Count
0 %	Equi-Hist	1.360	2.547	1.591	169.160	—
	Equi-Width	1.409	2.590	1.621	1.760	—
	Hybrid_Shock	1.106	1.674	1.238	1.709	—
	Hybrid_FT	1.106	1.674	1.238	0.000	0
	Hybrid_RB	1.106	1.674	1.238	0.000	0
5 %	Equi-Hist	1.368	2.787	2.038	168.712	—
	Equi-Width	1.451	8.006	10.095	1.638	—
	Hybrid_Shock	1.108	2.045	4.511	1.662	—
	Hybrid_FT	1.100	1.969	3.912	0.629	873
	Hybrid_RB	1.088	1.649	1.265	0.661	140
10 %	Equi-Hist	1.374	3.070	4.055	166.021	—
	Equi-Width	1.484	128.519	32.557	1.734	—
	Hybrid_Shock	1.122	1.670	1.318	1.686	—
	Hybrid_FT	1.110	1.643	1.301	0.728	873
	Hybrid_RB	1.110	1.634	1.279	0.642	141
15 %	Equi-Hist	1.378	3.436	2.857	165.599	—
	Equi-Width	1.523	273.843	69.744	1.688	—
	Hybrid_Shock	1.128	1.642	1.304	1.771	—
	Hybrid_FT	1.116	1.617	1.288	0.735	873
	Hybrid_RB	1.115	1.611	1.271	0.541	126
20 %	Equi-Hist	1.382	3.750	4.237	167.299	—
	Equi-Width	1.560	456.856	116.447	1.661	—
	Hybrid_Shock	1.143	1.633	1.323	1.609	—
	Hybrid_FT	1.129	1.605	1.303	0.817	873
	Hybrid_RB	1.128	1.601	1.285	0.646	136
25 %	Equi-Hist	1.389	4.101	5.896	165.326	—
	Equi-Width	1.603	676.183	172.395	1.791	—
	Hybrid_Shock	1.152	1.625	1.335	1.776	—
	Hybrid_FT	1.136	1.599	1.315	0.870	873
	Hybrid_RB	1.136	1.595	1.297	0.598	142
30 %	Equi-Hist	1.394	4.449	2.608	166.791	—
	Equi-Width	1.644	918.241	239.515	1.657	—

Continued on next page

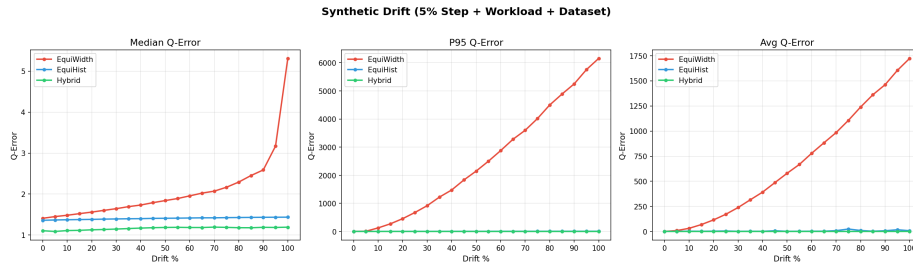
Table 4.17 – Continued from previous page

Drift	Model	Median	P95	Avg	Time	Count
	Hybrid_Shock	1.164	1.613	1.335	1.694	–
	Hybrid_FT	1.147	1.586	1.312	0.892	873
	Hybrid_RB	1.146	1.582	1.296	0.591	124
35 %	Equi-Hist	1.398	4.841	3.018	169.881	–
	Equi-Width	1.693	1223.234	316.723	1.760	–
	Hybrid_Shock	1.177	1.604	1.347	1.734	–
	Hybrid_FT	1.159	1.577	1.324	0.936	873
	Hybrid_RB	1.159	1.575	1.304	0.689	137
40 %	Equi-Hist	1.401	5.181	2.702	170.009	–
	Equi-Width	1.733	1476.635	392.855	1.735	–
	Hybrid_Shock	1.189	1.606	1.439	1.599	–
	Hybrid_FT	1.170	1.578	1.413	0.970	873
	Hybrid_RB	1.170	1.576	1.397	0.813	143
45 %	Equi-Hist	1.407	5.540	8.281	169.481	–
	Equi-Width	1.791	1833.376	487.000	1.721	–
	Hybrid_Shock	1.202	1.581	1.328	1.730	–
	Hybrid_FT	1.180	1.557	1.303	1.023	873
	Hybrid_RB	1.180	1.557	1.283	0.666	139
50 %	Equi-Hist	1.410	5.741	2.668	169.709	–
	Equi-Width	1.842	2148.902	580.365	1.783	–
	Hybrid_Shock	1.220	1.583	1.383	1.789	–
	Hybrid_FT	1.186	1.556	1.355	0.987	873
	Hybrid_RB	1.186	1.551	1.336	0.623	148
55 %	Equi-Hist	1.412	6.071	2.984	169.828	–
	Equi-Width	1.891	2500.342	667.910	1.658	–
	Hybrid_Shock	1.231	1.575	1.337	1.649	–
	Hybrid_FT	1.189	1.551	1.310	0.989	873
	Hybrid_RB	1.189	1.546	1.290	0.683	160
60 %	Equi-Hist	1.417	6.504	3.566	169.998	–
	Equi-Width	1.957	2877.857	778.782	1.802	–
	Hybrid_Shock	1.227	1.562	1.405	1.664	–
	Hybrid_FT	1.185	1.540	1.374	1.060	873
	Hybrid_RB	1.185	1.540	1.356	0.758	145
65 %	Equi-Hist	1.421	6.767	3.814	169.895	–
	Equi-Width	2.024	3278.341	882.895	1.678	–
	Hybrid_Shock	1.226	1.570	1.443	1.669	–
	Hybrid_FT	1.184	1.546	1.410	1.076	873
	Hybrid_RB	1.184	1.538	1.391	0.710	152
70 %	Equi-Hist	1.421	6.879	9.348	169.272	–
	Equi-Width	2.072	3600.113	985.215	1.621	–
	Hybrid_Shock	1.238	1.557	1.362	1.693	–
	Hybrid_FT	1.194	1.535	1.323	1.085	872
	Hybrid_RB	1.194	1.531	1.305	0.716	167
75 %	Equi-Hist	1.426	7.250	25.318	180.995	–
	Equi-Width	2.168	4015.648	1106.383	1.740	–

Continued on next page

Table 4.17 – Continued from previous page

Drift	Model	Median	P95	Avg	Time	Count
	Hybrid_Shock	1.230	1.566	1.559	1.617	–
	Hybrid_FT	1.187	1.538	1.515	1.114	872
	Hybrid_RB	1.187	1.534	1.491	0.770	182
80 %	Equi-Hist	1.428	7.419	10.978	169.297	–
	Equi-Width	2.294	4499.004	1239.629	1.660	–
	Hybrid_Shock	1.221	1.561	1.462	1.567	–
	Hybrid_FT	1.180	1.533	1.423	1.161	873
	Hybrid_RB	1.180	1.528	1.395	0.738	173
85 %	Equi-Hist	1.431	7.674	3.766	169.037	–
	Equi-Width	2.453	4885.718	1361.699	1.665	–
	Hybrid_Shock	1.219	1.551	1.456	1.657	–
	Hybrid_FT	1.179	1.526	1.413	1.198	873
	Hybrid_RB	1.179	1.521	1.387	0.757	180
90 %	Equi-Hist	1.433	7.861	8.361	169.291	–
	Equi-Width	2.589	5250.376	1461.920	1.707	–
	Hybrid_Shock	1.231	1.551	1.613	1.598	–
	Hybrid_FT	1.189	1.525	1.567	1.202	871
	Hybrid_RB	1.189	1.521	1.549	0.885	174
95 %	Equi-Hist	1.435	8.048	18.389	169.395	–
	Equi-Width	3.172	5757.197	1604.307	1.705	–
	Hybrid_Shock	1.228	1.535	1.479	1.612	–
	Hybrid_FT	1.186	1.509	1.437	1.216	872
	Hybrid_RB	1.186	1.505	1.410	0.806	175
100 %	Equi-Hist	1.437	8.173	8.627	169.056	–
	Equi-Width	5.312	6155.812	1723.247	1.657	–
	Hybrid_Shock	1.236	1.531	1.633	1.703	–
	Hybrid_FT	1.192	1.505	1.572	1.220	872
	Hybrid_RB	1.192	1.501	1.537	0.804	172

Figure 4-9: 5% step, normal $\rightarrow$ normal + zipf, workload normal $\rightarrow$ zipf

This experiment shows the results that initially was set as the goals of that research, under the circumstances of additive data change, in which the hybrid model does not unlearn the initial distributions as it changes, but it rather learns new data that comes from zipf, the rebuild time is lower than in previous experiments, and

finetune time + rebuild time is the same as the Equi-Width rebuild time. While keeping the same time as baseline, Hybrid FD CDF approach manages to outperform the baseline and keeping the best median, p95 and average Q-error across all the steps. We found out that in the scenario of data change, the hybrid model performs better than Equi-Width, however it takes more time because of the rebuild process. But in the additive experiment we observe that additive manner of change result in better finetune process, making less buckets being rebuilt.

DSB Benchmark

Evaluation on DSB Benchmark is important since we test approach on real data distributions and schema.

Table 4.18: Detailed Drift Results: `ws_item_sk` (5% Step, Displacement Drift)

Drift	Model	Median	P95	Avg	Time (s)	Count
0 %	Equi-Hist	1.051	1.592	1.140	19.870	–
	Equi-Width	1.140	1.756	1.221	0.295	–
	Hybrid_Shock	1.015	1.055	1.020	0.378	–
	Hybrid_FT	1.015	1.055	1.020	0.000	0
	Hybrid_RB	1.015	1.055	1.020	0.000	0
5 %	Equi-Hist	1.051	1.595	1.141	19.803	–
	Equi-Width	1.134	1.762	1.218	0.317	–
	Hybrid_Shock	1.020	1.057	1.024	0.361	–
	Hybrid_FT	1.022	1.061	1.026	0.006	1
	Hybrid_RB	1.020	1.057	1.024	0.000	0
10 %	Equi-Hist	1.051	1.592	1.140	19.473	–
	Equi-Width	1.140	1.756	1.221	0.331	–
	Hybrid_Shock	1.015	1.054	1.019	0.344	–
	Hybrid_FT	1.015	1.059	1.021	0.006	1
	Hybrid_RB	1.015	1.054	1.019	0.000	0
15 %	Equi-Hist	1.051	1.592	1.140	19.775	–
	Equi-Width	1.141	1.755	1.222	0.341	–
	Hybrid_Shock	1.014	1.046	1.018	0.351	–
	Hybrid_FT	1.014	1.050	1.019	0.007	1
	Hybrid_RB	1.014	1.046	1.018	0.000	0
20 %	Equi-Hist	1.051	1.592	1.140	19.841	–
	Equi-Width	1.140	1.756	1.222	0.358	–
	Hybrid_Shock	1.015	1.050	1.020	0.369	–
	Hybrid_FT	1.015	1.054	1.021	0.005	1
Continued on next page						

Table 4.18 – Continued from previous page

Drift	Model	Median	P95	Avg	Time (s)	Count
	Hybrid_RB	1.015	1.050	1.020	0.000	0
25 %	Equi-Hist	1.051	1.588	1.139	19.782	–
	Equi-Width	1.146	1.754	1.225	0.313	–
	Hybrid_Shock	1.015	1.063	1.021	0.382	–
	Hybrid_FT	1.015	1.072	1.023	0.005	1
	Hybrid_RB	1.015	1.063	1.021	0.000	0
30 %	Equi-Hist	1.050	1.565	1.135	19.799	–
	Equi-Width	1.169	1.746	1.238	0.324	–
	Hybrid_Shock	1.018	1.062	1.024	0.367	–
	Hybrid_FT	1.019	1.067	1.025	0.005	1
	Hybrid_RB	1.018	1.062	1.024	0.000	0
35 %	Equi-Hist	1.049	1.548	1.131	19.832	–
	Equi-Width	1.189	1.712	1.251	0.340	–
	Hybrid_Shock	1.020	1.060	1.025	0.401	–
	Hybrid_FT	1.021	1.067	1.027	0.005	1
	Hybrid_RB	1.020	1.060	1.025	0.000	0
40 %	Equi-Hist	1.048	1.531	1.127	19.801	–
	Equi-Width	1.210	1.683	1.264	0.337	–
	Hybrid_Shock	1.019	1.061	1.023	0.429	–
	Hybrid_FT	1.019	1.068	1.024	0.006	1
	Hybrid_RB	1.019	1.061	1.023	0.000	0
45 %	Equi-Hist	1.048	1.515	1.124	19.805	–
	Equi-Width	1.221	1.656	1.271	0.339	–
	Hybrid_Shock	1.022	1.075	1.028	0.409	–
	Hybrid_FT	1.024	1.083	1.030	0.005	1
	Hybrid_RB	1.022	1.075	1.028	0.000	0
50 %	Equi-Hist	1.047	1.500	1.121	19.763	–
	Equi-Width	1.245	1.650	1.287	0.313	–
	Hybrid_Shock	1.023	1.076	1.028	0.395	–
	Hybrid_FT	1.023	1.082	1.029	0.005	1
	Hybrid_RB	1.023	1.076	1.028	0.000	0
55 %	Equi-Hist	1.047	1.487	1.119	19.795	–
	Equi-Width	1.256	1.670	1.295	0.330	–
	Hybrid_Shock	1.022	1.072	1.029	0.409	–
	Hybrid_FT	1.024	1.080	1.031	0.005	1
	Hybrid_RB	1.022	1.072	1.029	0.000	0
60 %	Equi-Hist	1.046	1.474	1.116	19.837	–
	Equi-Width	1.265	1.693	1.303	0.356	–
	Hybrid_Shock	1.023	1.080	1.030	0.379	–
	Hybrid_FT	1.025	1.089	1.032	0.006	1
	Hybrid_RB	1.023	1.080	1.030	0.000	0
65 %	Equi-Hist	1.046	1.457	1.112	19.768	–

Continued on next page

Table 4.18 – Continued from previous page

Drift	Model	Median	P95	Avg	Time (s)	Count
	Equi-Width	1.281	1.712	1.316	0.318	–
	Hybrid_Shock	1.023	1.078	1.030	0.402	–
	Hybrid_FT	1.022	1.074	1.029	0.005	2
	Hybrid_RB	1.023	1.078	1.030	0.000	0
70 %	Equi-Hist	1.045	1.445	1.110	19.513	–
	Equi-Width	1.291	1.703	1.326	0.333	–
	Hybrid_Shock	1.026	1.082	1.032	0.385	–
	Hybrid_FT	1.025	1.082	1.032	0.005	2
	Hybrid_RB	1.026	1.082	1.032	0.000	0
75 %	Equi-Hist	1.045	1.432	1.107	19.517	–
	Equi-Width	1.302	1.707	1.339	0.398	–
	Hybrid_Shock	1.022	1.077	1.029	0.346	–
	Hybrid_FT	1.023	1.092	1.031	0.008	4
	Hybrid_RB	1.022	1.077	1.029	0.000	0
80 %	Equi-Hist	1.044	1.422	1.105	19.493	–
	Equi-Width	1.316	1.711	1.348	0.335	–
	Hybrid_Shock	1.024	1.089	1.033	0.346	–
	Hybrid_FT	1.024	1.093	1.033	0.004	2
	Hybrid_RB	1.024	1.089	1.033	0.000	0
85 %	Equi-Hist	1.044	1.414	1.104	19.245	–
	Equi-Width	1.327	1.694	1.357	0.363	–
	Hybrid_Shock	1.022	1.085	1.031	0.434	–
	Hybrid_FT	1.023	1.086	1.032	0.006	2
	Hybrid_RB	1.022	1.085	1.031	0.000	0
90 %	Equi-Hist	1.044	1.406	1.102	19.214	–
	Equi-Width	1.336	1.689	1.366	0.356	–
	Hybrid_Shock	1.024	1.087	1.032	0.450	–
	Hybrid_FT	1.026	1.100	1.034	0.010	4
	Hybrid_RB	1.024	1.087	1.032	0.000	0
95 %	Equi-Hist	1.043	1.397	1.100	19.102	–
	Equi-Width	1.350	1.690	1.379	0.345	–
	Hybrid_Shock	1.023	1.076	1.031	0.400	–
	Hybrid_FT	1.025	1.084	1.032	0.023	8
	Hybrid_RB	1.023	1.076	1.031	0.000	0
100 %	Equi-Hist	1.043	1.389	1.098	19.085	–
	Equi-Width	1.360	1.705	1.387	0.368	–
	Hybrid_Shock	1.022	1.101	1.032	0.412	–
	Hybrid_FT	1.022	1.090	1.031	0.012	9
	Hybrid_RB	1.022	1.101	1.032	0.000	0

On this benchmark which consists of lower amount of rows, we observe that the step in 5% does not trigger finetune or rebuild processes in hybrid that much, all

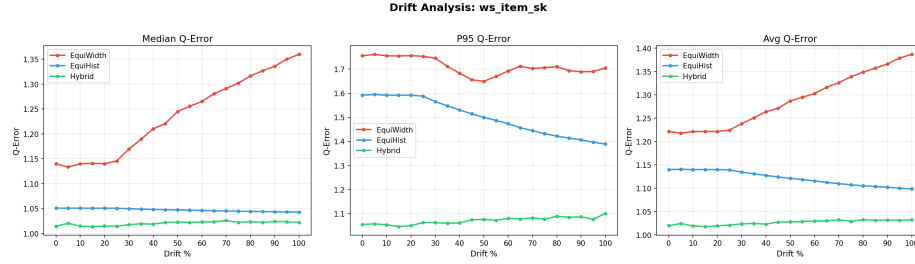


Figure 4-10: 5% step, DSB dataset ws_item_sk

three methods performs relatively good maintaining Q-error under 1 across all the steps.

Table 4.19: Detailed Drift Results: cr_item_sk (5% Step, Displacement Drift)

Drift	Model	Median	P95	Avg	Time (s)	Count
0 %	Equi-Hist	1.090	1.536	1.157	13.241	—
	Equi-Width	1.106	1.924	1.205	0.220	—
	Hybrid_Shock	1.013	1.075	1.029	0.307	—
	Hybrid_FT	1.013	1.075	1.029	0.000	0
	Hybrid_RB	1.013	1.075	1.029	0.000	0
5 %	Equi-Hist	1.087	1.504	1.149	13.364	—
	Equi-Width	1.165	1.813	1.230	0.250	—
	Hybrid_Shock	1.017	1.088	1.035	0.330	—
	Hybrid_FT	1.021	1.083	1.035	0.084	5
	Hybrid_RB	1.017	1.088	1.035	0.000	0
10 %	Equi-Hist	1.085	1.484	1.143	13.562	—
	Equi-Width	1.208	1.762	1.255	0.245	—
	Hybrid_Shock	1.017	1.093	1.035	0.277	—
	Hybrid_FT	1.020	1.078	1.035	0.084	5
	Hybrid_RB	1.017	1.093	1.035	0.000	0
15 %	Equi-Hist	1.083	1.473	1.140	13.386	—
	Equi-Width	1.210	1.740	1.255	0.220	—
	Hybrid_Shock	1.017	1.122	1.034	0.326	—
	Hybrid_FT	1.020	1.082	1.034	0.084	5
	Hybrid_RB	1.017	1.122	1.034	0.000	0
20 %	Equi-Hist	1.082	1.463	1.137	13.278	—
	Equi-Width	1.213	1.725	1.258	0.266	—
	Hybrid_Shock	1.017	1.117	1.036	0.292	—
	Hybrid_FT	1.021	1.089	1.036	0.089	5
	Hybrid_RB	1.017	1.117	1.036	0.000	0
25 %	Equi-Hist	1.081	1.451	1.134	13.386	—
	Equi-Width	1.208	1.718	1.259	0.244	—
	Hybrid_Shock	1.020	1.128	1.039	0.315	—

Continued on next page

Table 4.19 – Continued from previous page

Drift	Model	Median	P95	Avg	Time (s)	Count
	Hybrid_FT	1.024	1.086	1.039	0.087	5
	Hybrid_RB	1.020	1.128	1.039	0.000	0
30 %	Equi-Hist	1.080	1.442	1.132	13.331	–
	Equi-Width	1.214	1.707	1.264	0.239	–
	Hybrid_Shock	1.022	1.135	1.042	0.309	–
	Hybrid_FT	1.026	1.106	1.042	0.091	5
	Hybrid_RB	1.022	1.135	1.042	0.000	0
35 %	Equi-Hist	1.079	1.433	1.129	13.451	–
	Equi-Width	1.211	1.701	1.265	0.256	–
	Hybrid_Shock	1.020	1.133	1.038	0.293	–
	Hybrid_FT	1.021	1.109	1.037	0.087	5
	Hybrid_RB	1.020	1.133	1.038	0.000	0
40 %	Equi-Hist	1.078	1.425	1.127	13.380	–
	Equi-Width	1.209	1.694	1.265	0.266	–
	Hybrid_Shock	1.020	1.123	1.037	0.299	–
	Hybrid_FT	1.023	1.100	1.037	0.089	5
	Hybrid_RB	1.020	1.123	1.037	0.000	0
45 %	Equi-Hist	1.077	1.416	1.124	13.416	–
	Equi-Width	1.209	1.688	1.267	0.252	–
	Hybrid_Shock	1.021	1.137	1.041	0.290	–
	Hybrid_FT	1.026	1.114	1.041	0.091	8
	Hybrid_RB	1.021	1.137	1.041	0.000	0
50 %	Equi-Hist	1.076	1.407	1.122	13.294	–
	Equi-Width	1.206	1.697	1.268	0.279	–
	Hybrid_Shock	1.024	1.153	1.043	0.297	–
	Hybrid_FT	1.027	1.111	1.042	0.093	11
	Hybrid_RB	1.024	1.153	1.043	0.000	0
55 %	Equi-Hist	1.075	1.397	1.120	13.379	–
	Equi-Width	1.214	1.687	1.274	0.254	–
	Hybrid_Shock	1.024	1.135	1.043	0.338	–
	Hybrid_FT	1.029	1.121	1.044	0.102	8
	Hybrid_RB	1.024	1.135	1.043	0.000	0
60 %	Equi-Hist	1.074	1.391	1.118	13.453	–
	Equi-Width	1.217	1.701	1.278	0.266	–
	Hybrid_Shock	1.025	1.154	1.044	0.305	–
	Hybrid_FT	1.028	1.126	1.044	0.105	11
	Hybrid_RB	1.025	1.154	1.044	0.000	0
65 %	Equi-Hist	1.073	1.385	1.115	13.336	–
	Equi-Width	1.218	1.705	1.281	0.262	–
	Hybrid_Shock	1.026	1.145	1.044	0.312	–
	Hybrid_FT	1.030	1.125	1.045	0.105	11
	Hybrid_RB	1.026	1.145	1.044	0.000	0

Continued on next page

Table 4.19 – Continued from previous page

Drift	Model	Median	P95	Avg	Time (s)	Count
70 %	Equi-Hist	1.072	1.378	1.114	13.417	–
	Equi-Width	1.221	1.735	1.286	0.250	–
	Hybrid_Shock	1.028	1.155	1.046	0.307	–
	Hybrid_FT	1.030	1.137	1.046	0.103	11
	Hybrid_RB	1.028	1.155	1.046	0.000	0
75 %	Equi-Hist	1.071	1.372	1.112	13.467	–
	Equi-Width	1.221	1.742	1.288	0.250	–
	Hybrid_Shock	1.029	1.161	1.048	0.287	–
	Hybrid_FT	1.033	1.136	1.048	0.101	11
	Hybrid_RB	1.029	1.161	1.048	0.000	0
80 %	Equi-Hist	1.070	1.365	1.110	13.446	–
	Equi-Width	1.231	1.764	1.294	0.269	–
	Hybrid_Shock	1.030	1.155	1.049	0.317	–
	Hybrid_FT	1.032	1.137	1.049	0.113	12
	Hybrid_RB	1.030	1.155	1.049	0.000	0
85 %	Equi-Hist	1.069	1.359	1.108	13.502	–
	Equi-Width	1.237	1.771	1.299	0.265	–
	Hybrid_Shock	1.029	1.155	1.048	0.298	–
	Hybrid_FT	1.032	1.137	1.048	0.107	12
	Hybrid_RB	1.029	1.155	1.048	0.000	0
90 %	Equi-Hist	1.068	1.354	1.106	13.422	–
	Equi-Width	1.244	1.768	1.305	0.246	–
	Hybrid_Shock	1.029	1.149	1.049	0.288	–
	Hybrid_FT	1.034	1.139	1.050	0.108	15
	Hybrid_RB	1.029	1.149	1.049	0.000	0
95 %	Equi-Hist	1.067	1.346	1.104	13.377	–
	Equi-Width	1.249	1.781	1.310	0.256	–
	Hybrid_Shock	1.032	1.169	1.054	0.305	–
	Hybrid_FT	1.035	1.148	1.052	0.114	15
	Hybrid_RB	1.032	1.169	1.054	0.000	0
100 %	Equi-Hist	1.066	1.339	1.103	13.419	–
	Equi-Width	1.257	1.809	1.316	0.243	–
	Hybrid_Shock	1.033	1.164	1.053	0.321	–
	Hybrid_FT	1.038	1.158	1.053	0.111	15
	Hybrid_RB	1.033	1.164	1.053	0.000	0

Comparing the Hybrid with a Equi-Hist, second one takes a much more time for inference maintaining good Q-error across the steps, while hybrid performs operations faster in, and maintains median Q-error (1.013-1.033). The Equi-Width experiences statistical decay resulting in P95 error of 1.809. The hybrid models shows the persistence and stability between the steps, which may suggest that initially partitioned

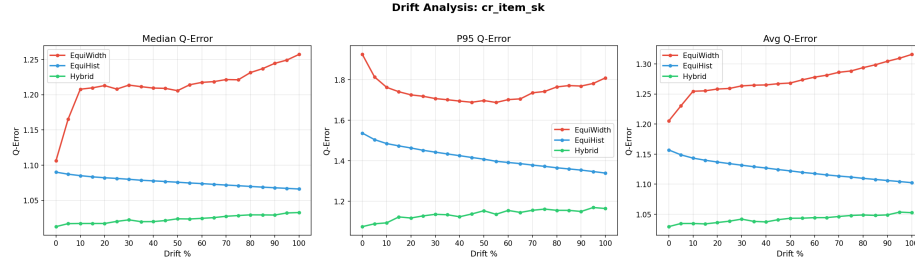


Figure 4-11: 5% step, DSB dataset cr_item_sk

Freedman Diaconis Rule helped to create a robust structure that prevents the catastrophic data strutcure changes.

Table 4.20: Detailed Drift Results: cr_returned_time_sk (5% Step, Displacement Drift)

Drift	Model	Median	P95	Avg	Time (s)	Count
0 %	Equi-Hist	1.007	1.025	1.009	15.730	—
	Equi-Width	1.040	1.162	1.056	0.283	—
	Hybrid_Shock	1.012	1.037	1.015	0.332	—
	Hybrid_FT	1.012	1.037	1.015	0.000	0
	Hybrid_RB	1.012	1.037	1.015	0.000	0
5 %	Equi-Hist	1.008	1.030	1.011	15.719	—
	Equi-Width	1.055	1.267	1.093	0.272	—
	Hybrid_Shock	1.012	1.044	1.016	0.349	—
	Hybrid_FT	1.012	1.044	1.016	0.004	0
	Hybrid_RB	1.012	1.044	1.016	0.000	0
10 %	Equi-Hist	1.009	1.034	1.012	16.034	—
	Equi-Width	1.085	1.342	1.126	0.304	—
	Hybrid_Shock	1.014	1.044	1.017	0.337	—
	Hybrid_FT	1.014	1.044	1.017	0.005	1
	Hybrid_RB	1.014	1.044	1.017	0.000	0
15 %	Equi-Hist	1.010	1.037	1.013	15.997	—
	Equi-Width	1.078	1.357	1.129	0.306	—
	Hybrid_Shock	1.014	1.046	1.019	0.307	—
	Hybrid_FT	1.014	1.046	1.019	0.004	1
	Hybrid_RB	1.014	1.046	1.019	0.000	0
20 %	Equi-Hist	1.010	1.039	1.014	15.957	—
	Equi-Width	1.078	1.383	1.134	0.276	—
	Hybrid_Shock	1.015	1.050	1.019	0.319	—
	Hybrid_FT	1.014	1.050	1.018	0.005	1
	Hybrid_RB	1.015	1.050	1.019	0.000	0
25 %	Equi-Hist	1.011	1.041	1.015	15.722	—
	Equi-Width	1.077	1.387	1.136	0.291	—

Continued on next page

Table 4.20 – Continued from previous page

Drift	Model	Median	P95	Avg	Time (s)	Count
	Hybrid_Shock	1.016	1.049	1.020	0.340	–
	Hybrid_FT	1.016	1.049	1.020	0.005	1
	Hybrid_RB	1.016	1.049	1.020	0.000	0
30 %	Equi-Hist	1.011	1.043	1.015	15.736	–
	Equi-Width	1.083	1.402	1.142	0.291	–
	Hybrid_Shock	1.018	1.058	1.022	0.335	–
	Hybrid_FT	1.018	1.058	1.022	0.004	1
	Hybrid_RB	1.018	1.058	1.022	0.000	0
35 %	Equi-Hist	1.012	1.045	1.016	15.619	–
	Equi-Width	1.089	1.413	1.146	0.263	–
	Hybrid_Shock	1.016	1.060	1.021	0.335	–
	Hybrid_FT	1.016	1.060	1.021	0.005	1
	Hybrid_RB	1.016	1.060	1.021	0.000	0
40 %	Equi-Hist	1.013	1.047	1.017	15.943	–
	Equi-Width	1.099	1.412	1.149	0.285	–
	Hybrid_Shock	1.017	1.057	1.022	0.311	–
	Hybrid_FT	1.017	1.057	1.022	0.005	1
	Hybrid_RB	1.017	1.057	1.022	0.000	0
45 %	Equi-Hist	1.013	1.050	1.018	16.006	–
	Equi-Width	1.105	1.423	1.154	0.285	–
	Hybrid_Shock	1.019	1.061	1.023	0.368	–
	Hybrid_FT	1.019	1.061	1.023	0.005	1
	Hybrid_RB	1.019	1.061	1.023	0.000	0
50 %	Equi-Hist	1.014	1.052	1.018	15.920	–
	Equi-Width	1.112	1.421	1.159	0.271	–
	Hybrid_Shock	1.020	1.062	1.024	0.337	–
	Hybrid_FT	1.020	1.062	1.024	0.005	1
	Hybrid_RB	1.020	1.062	1.024	0.000	0
55 %	Equi-Hist	1.014	1.054	1.019	17.723	–
	Equi-Width	1.118	1.444	1.167	0.293	–
	Hybrid_Shock	1.018	1.065	1.023	0.355	–
	Hybrid_FT	1.018	1.065	1.024	0.005	1
	Hybrid_RB	1.018	1.065	1.023	0.000	0
60 %	Equi-Hist	1.015	1.055	1.020	16.224	–
	Equi-Width	1.124	1.459	1.174	0.276	–
	Hybrid_Shock	1.018	1.068	1.024	0.357	–
	Hybrid_FT	1.018	1.068	1.024	0.004	1
	Hybrid_RB	1.018	1.068	1.024	0.000	0
65 %	Equi-Hist	1.016	1.058	1.021	18.241	–
	Equi-Width	1.132	1.464	1.180	0.292	–
	Hybrid_Shock	1.019	1.068	1.026	0.322	–
	Hybrid_FT	1.019	1.068	1.027	0.005	1

Continued on next page

Table 4.20 – Continued from previous page

Drift	Model	Median	P95	Avg	Time (s)	Count
	Hybrid_RB	1.019	1.068	1.026	0.000	0
70 %	Equi-Hist	1.016	1.060	1.021	17.936	–
	Equi-Width	1.137	1.473	1.187	0.298	–
	Hybrid_Shock	1.023	1.074	1.029	0.348	–
	Hybrid_FT	1.023	1.074	1.029	0.004	1
	Hybrid_RB	1.023	1.074	1.029	0.000	0
75 %	Equi-Hist	1.017	1.061	1.022	16.834	–
	Equi-Width	1.144	1.475	1.193	0.261	–
	Hybrid_Shock	1.021	1.074	1.027	0.374	–
	Hybrid_FT	1.021	1.074	1.027	0.006	1
	Hybrid_RB	1.021	1.074	1.027	0.000	0
80 %	Equi-Hist	1.017	1.063	1.022	16.075	–
	Equi-Width	1.152	1.484	1.200	0.297	–
	Hybrid_Shock	1.020	1.076	1.027	0.327	–
	Hybrid_FT	1.020	1.076	1.027	0.004	1
	Hybrid_RB	1.020	1.076	1.027	0.000	0
85 %	Equi-Hist	1.017	1.064	1.023	16.201	–
	Equi-Width	1.158	1.507	1.206	0.302	–
	Hybrid_Shock	1.024	1.081	1.030	0.348	–
	Hybrid_FT	1.024	1.081	1.030	0.006	1
	Hybrid_RB	1.024	1.081	1.030	0.000	0
90 %	Equi-Hist	1.018	1.065	1.023	16.224	–
	Equi-Width	1.164	1.531	1.215	0.273	–
	Hybrid_Shock	1.021	1.080	1.028	0.394	–
	Hybrid_FT	1.021	1.080	1.028	0.006	1
	Hybrid_RB	1.021	1.080	1.028	0.000	0
95 %	Equi-Hist	1.018	1.066	1.024	16.125	–
	Equi-Width	1.168	1.548	1.221	0.304	–
	Hybrid_Shock	1.024	1.081	1.030	0.335	–
	Hybrid_FT	1.024	1.081	1.031	0.004	1
	Hybrid_RB	1.024	1.081	1.030	0.000	0
100 %	Equi-Hist	1.019	1.068	1.025	15.738	–
	Equi-Width	1.175	1.570	1.229	0.283	–
	Hybrid_Shock	1.024	1.083	1.031	0.371	–
	Hybrid_FT	1.024	1.083	1.031	0.007	1
	Hybrid_RB	1.024	1.083	1.031	0.000	0

Dynamic drift results on DSB columns validate the Hybrid FD CDF approach. For `ws_item_sk` and `cr_item_sk`, we note that the model’s Median Q-Error stayed between 1.013 and 1.033 across all steps of the 5% displacement drift, while visible statistical decay occurred for the Equi-Width baseline whose P95 error rose to 1.809

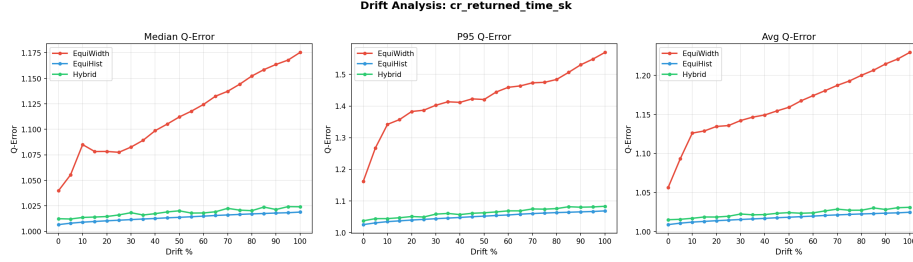


Figure 4-12: 5% step, DSB dataset `cr_returned_time_sk`

for `cr_item_sk`.

We also note that the 5% drift steps in these real-world columns resulted in significantly fewer rebuild operations when compared to the synthetic "Hard" datasets. This confirms our hypothesis that the initial partitioning from the Freedman-Diaconis rule provides a solid structural baseline from which real-world shifts are primarily handled by FineTuning. Equi-Hist could keep the error low, but the constant latency overhead is prohibitive, while the Hybrid had near-instantaneous recovery from the drift shock at each checkpoint.

4.7.4 Dynamic setup query one-by-one inference

Synthetic data

Table 4.21: Detailed Drift Results: Normal $\rightarrow$ Zipfian (10% Step, Displacement Drift, Normal Workload query one-by-one inference). Best Median Q-Error per distribution is highlighted in bold.

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
0%	Equi-Hist	1.359	2.529	1.589	167.809	—
	Equi-Width	1.415	2.602	1.642	1.246	—
	Hybrid_Shock	1.138	1.934	1.376	11.487	—
	Hybrid_FT	1.138	1.934	1.376	0.000	0
	Hybrid_RB	1.138	1.934	1.376	0.000	0
10%	Equi-Hist	1.359	2.536	1.599	167.679	—
	Equi-Width	1.355	2.527	1.632	1.199	—
	Hybrid_Shock	1.140	1.953	1.409	11.484	—
	Hybrid_FT	1.128	1.915	1.354	0.876	873
	Hybrid_RB	1.127	1.897	1.349	0.870	164
20%	Equi-Hist	1.359	2.535	1.596	167.805	—
	Continued on next page					

Table 4.21 – Continued from previous page

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
	Equi-Width	1.334	2.690	1.663	1.198	–
	Hybrid_Shock	1.140	1.933	1.368	11.688	–
	Hybrid_FT	1.128	1.900	1.340	0.790	873
	Hybrid_RB	1.127	1.889	1.341	0.738	151
30%	Equi-Hist	1.358	2.535	1.602	166.980	–
	Equi-Width	1.349	3.055	1.750	1.216	–
	Hybrid_Shock	1.147	1.951	1.457	11.689	–
	Hybrid_FT	1.133	1.909	1.340	0.912	873
	Hybrid_RB	1.132	1.905	1.351	0.751	156
40%	Equi-Hist	1.358	2.532	1.597	167.069	–
	Equi-Width	1.410	3.549	1.916	1.238	–
	Hybrid_Shock	1.149	1.974	1.391	11.818	–
	Hybrid_FT	1.134	1.928	1.351	0.905	873
	Hybrid_RB	1.134	1.928	1.368	0.889	166
50%	Equi-Hist	1.358	2.534	1.606	168.475	–
	Equi-Width	1.613	4.258	2.210	1.213	–
	Hybrid_Shock	1.154	1.972	1.498	11.882	–
	Hybrid_FT	1.138	1.928	1.360	0.935	873
	Hybrid_RB	1.138	1.928	1.386	0.931	173
60%	Equi-Hist	1.357	2.533	1.610	168.375	–
	Equi-Width	2.011	5.309	2.704	1.219	–
	Hybrid_Shock	1.160	2.021	1.603	12.006	–
	Hybrid_FT	1.143	1.963	1.385	0.956	872
	Hybrid_RB	1.143	1.978	1.416	1.004	209
70%	Equi-Hist	1.356	2.524	1.607	168.192	–
	Equi-Width	2.671	7.014	3.535	1.206	–
	Hybrid_Shock	1.176	2.093	1.680	12.153	–
	Hybrid_FT	1.156	2.017	1.417	1.043	872
	Hybrid_RB	1.155	2.049	1.459	1.387	267
80%	Equi-Hist	1.356	2.530	1.622	166.898	–
	Equi-Width	4.003	10.525	5.212	1.215	–
	Hybrid_Shock	1.188	2.197	1.879	12.352	–
	Hybrid_FT	1.168	2.098	1.459	1.055	872
	Hybrid_RB	1.166	2.177	1.674	1.403	287
90%	Equi-Hist	1.357	2.552	1.671	167.521	–
	Equi-Width	8.001	21.063	10.259	1.222	–
	Hybrid_Shock	1.229	2.504	3.561	12.585	–
	Hybrid_FT	1.201	2.377	1.642	1.269	871
	Hybrid_RB	1.198	2.609	2.004	1.915	385
100%	Equi-Hist	1.096	1.722	1.612	171.368	–
	Equi-Width	1685.737	3406.806	1718.994	1.205	–
	Hybrid_Shock	21.000	88.000	43.121	3.492	–
	Hybrid_FT	127.281	329.941	188.079	2.149	872
	Hybrid_RB	8.964	452.110	124.412	2.948	844

The inference of queries one by one with the 10% drift shows that the Hybrid

model accuracy and recovery mechanisms function identically to the batches setup, bringing the Q-Error from 1685.7 to 8.9. However without the batch processing the inference time increased significantly, taking up to 4 seconds to process the workload, finetune buckets and rebuild the least performance ones.

Table 4.22: Detailed Drift Results: Normal $\rightarrow$ Zipfian (5% Step, Displacement Drift, Normal Workload, One-by-One Inference). Best Median Q-Error per distribution is highlighted in bold.

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
0%	Equi-Hist	1.359	2.529	1.589	168.514	–
	Equi-Width	1.415	2.602	1.642	1.235	–
	Hybrid_Shock	1.140	1.944	1.378	10.914	–
	Hybrid_FT	1.140	1.944	1.378	0.000	0
	Hybrid_RB	1.140	1.944	1.378	0.000	0
5%	Equi-Hist	1.359	2.533	1.589	169.175	–
	Equi-Width	1.382	2.552	1.631	1.254	–
	Hybrid_Shock	1.139	1.956	1.363	10.986	–
	Hybrid_FT	1.128	1.918	1.347	0.853	873
	Hybrid_RB	1.127	1.902	1.347	0.760	154
10%	Equi-Hist	1.359	2.535	1.597	169.025	–
	Equi-Width	1.355	2.527	1.632	1.250	–
	Hybrid_Shock	1.141	1.944	1.409	11.098	–
	Hybrid_FT	1.130	1.906	1.336	0.879	873
	Hybrid_RB	1.129	1.904	1.338	0.681	141
15%	Equi-Hist	1.359	2.535	1.605	170.322	–
	Equi-Width	1.339	2.567	1.641	1.269	–
	Hybrid_Shock	1.138	1.942	1.430	11.151	–
	Hybrid_FT	1.126	1.906	1.334	0.802	873
	Hybrid_RB	1.126	1.901	1.344	0.679	145
20%	Equi-Hist	1.359	2.535	1.591	168.531	–
	Equi-Width	1.334	2.690	1.663	1.249	–
	Hybrid_Shock	1.142	1.944	1.354	11.337	–
	Hybrid_FT	1.130	1.907	1.339	0.879	873
	Hybrid_RB	1.130	1.900	1.341	0.637	140
25%	Equi-Hist	1.358	2.534	1.600	168.832	–
	Equi-Width	1.338	2.857	1.698	1.264	–
	Hybrid_Shock	1.145	1.955	1.402	11.521	–
	Hybrid_FT	1.133	1.912	1.340	0.839	872
	Hybrid_RB	1.132	1.910	1.349	0.713	148
30%	Equi-Hist	1.358	2.534	1.599	168.115	–
	Equi-Width	1.349	3.055	1.750	1.240	–
	Hybrid_Shock	1.146	1.951	1.388	11.467	–
	Hybrid_FT	1.133	1.911	1.345	0.912	873
	Hybrid_RB	1.133	1.905	1.352	0.786	157

Continued on next page

Table 4.22 – Continued from previous page

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
35%	Equi-Hist	1.358	2.533	1.592	166.452	–
	Equi-Width	1.370	3.281	1.821	1.251	–
	Hybrid_Shock	1.145	1.963	1.374	11.207	–
	Hybrid_FT	1.132	1.916	1.350	0.848	873
	Hybrid_RB	1.132	1.919	1.361	0.722	159
40%	Equi-Hist	1.358	2.533	1.601	168.027	–
	Equi-Width	1.410	3.549	1.916	1.249	–
	Hybrid_Shock	1.149	1.961	1.448	11.471	–
	Hybrid_FT	1.135	1.921	1.353	0.893	872
	Hybrid_RB	1.134	1.918	1.363	0.774	167
45%	Equi-Hist	1.357	2.530	1.592	167.879	–
	Equi-Width	1.481	3.863	2.040	1.261	–
	Hybrid_Shock	1.149	1.968	1.392	11.504	–
	Hybrid_FT	1.135	1.924	1.356	0.909	872
	Hybrid_RB	1.134	1.923	1.372	0.770	165
50%	Equi-Hist	1.358	2.534	1.599	167.211	–
	Equi-Width	1.613	4.258	2.210	1.263	–
	Hybrid_Shock	1.153	1.974	1.415	11.689	–
	Hybrid_FT	1.138	1.929	1.365	0.881	873
	Hybrid_RB	1.138	1.929	1.371	0.897	183
55%	Equi-Hist	1.358	2.537	1.603	167.814	–
	Equi-Width	1.791	4.736	2.429	1.260	–
	Hybrid_Shock	1.159	2.011	1.489	11.579	–
	Hybrid_FT	1.142	1.958	1.371	0.875	873
	Hybrid_RB	1.141	1.964	1.401	0.933	203
60%	Equi-Hist	1.357	2.531	1.599	168.049	–
	Equi-Width	2.011	5.309	2.704	1.260	–
	Hybrid_Shock	1.157	2.036	1.469	11.828	–
	Hybrid_FT	1.141	1.980	1.394	0.916	873
	Hybrid_RB	1.140	1.988	1.423	1.087	213
65%	Equi-Hist	1.357	2.532	1.613	168.258	–
	Equi-Width	2.297	6.061	3.069	1.234	–
	Hybrid_Shock	1.165	2.056	1.695	11.945	–
	Hybrid_FT	1.148	1.993	1.406	0.936	873
	Hybrid_RB	1.147	2.009	1.431	0.971	218
70%	Equi-Hist	1.356	2.522	1.596	167.291	–
	Equi-Width	2.671	7.014	3.535	1.219	–
	Hybrid_Shock	1.174	2.095	1.590	11.952	–
	Hybrid_FT	1.155	2.028	1.424	1.038	871
	Hybrid_RB	1.154	2.018	1.468	1.206	240
75%	Equi-Hist	1.357	2.532	1.626	169.005	–
	Equi-Width	3.209	8.455	4.226	1.265	–
	Hybrid_Shock	1.179	2.154	1.978	12.522	–
	Hybrid_FT	1.161	2.078	1.455	1.052	872
Continued on next page						

Table 4.22 – Continued from previous page

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
	Hybrid_RB	1.159	2.080	1.528	1.284	278
80%	Equi-Hist	1.356	2.530	1.625	168.042	–
	Equi-Width	4.003	10.525	5.212	1.254	–
	Hybrid_Shock	1.183	2.196	2.425	12.579	–
	Hybrid_FT	1.164	2.116	1.523	1.018	872
	Hybrid_RB	1.162	2.162	1.621	1.444	296
85%	Equi-Hist	1.355	2.530	1.642	168.441	–
	Equi-Width	5.316	13.923	6.845	1.234	–
	Hybrid_Shock	1.209	2.372	2.546	12.970	–
	Hybrid_FT	1.185	2.263	1.612	1.198	870
	Hybrid_RB	1.182	2.326	1.786	1.535	344
90%	Equi-Hist	1.357	2.552	1.657	167.802	–
	Equi-Width	8.001	21.063	10.259	1.257	–
	Hybrid_Shock	1.236	2.753	3.230	13.306	–
	Hybrid_FT	1.207	2.568	1.728	1.249	871
	Hybrid_RB	1.202	2.672	2.056	2.023	400
95%	Equi-Hist	1.352	2.533	1.702	167.989	–
	Equi-Width	15.701	40.976	19.960	1.245	–
	Hybrid_Shock	1.322	4.189	7.040	14.506	–
	Hybrid_FT	1.283	3.730	2.310	1.486	869
	Hybrid_RB	1.276	5.638	3.397	2.517	538
100%	Equi-Hist	1.096	1.721	1.613	172.533	–
	Equi-Width	1685.737	3406.806	1718.994	1.234	–
	Hybrid_Shock	22.000	89.000	43.686	3.830	–
	Hybrid_FT	131.573	342.375	195.360	2.174	872
	Hybrid_RB	9.074	461.763	127.994	2.971	840

The 5% step provides higher resolution and more steps to recover the data changes, Hybrid model maintains the Q-Error at low rates but the inference is still 2-4x times higher than the Equi-Width.

Table 4.23: Detailed Drift Results: Normal $\rightarrow$ Normal + Zipfian (5% Step, Additive Drift, Workload Normal $\rightarrow$ Zipfian, One-by-One Inference). Best Median Q-Error per distribution is highlighted in bold.

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
0%	Equi-Hist	1.360	2.547	1.591	167.691	–
	Equi-Width	1.409	2.590	1.621	1.236	–
	Hybrid_Shock	1.139	1.925	1.303	10.573	–
	Hybrid_FT	1.139	1.925	1.303	0.000	0
	Hybrid_RB	1.139	1.925	1.303	0.000	0
	Equi-Hist	1.368	2.804	2.254	169.541	–

5%

Continued on next page

Table 4.23 – Continued from previous page

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
	Equi-Width	1.415	12.333	10.423	1.308	–
	Hybrid_Shock	1.146	2.298	3.899	10.209	–
	Hybrid_FT	1.135	2.217	3.486	0.871	873
	Hybrid_RB	1.118	1.876	1.316	0.617	138
10%	Equi-Hist	1.373	3.126	2.882	168.354	–
	Equi-Width	1.420	143.049	36.182	1.212	–
	Hybrid_Shock	1.164	1.886	1.343	11.604	–
	Hybrid_FT	1.150	1.854	1.329	0.876	873
	Hybrid_RB	1.149	1.846	1.316	0.648	132
15%	Equi-Hist	1.382	3.636	4.665	168.467	–
	Equi-Width	1.434	313.872	80.566	1.335	–
	Hybrid_Shock	1.176	1.862	1.359	12.189	–
	Hybrid_FT	1.148	1.834	1.344	0.880	873
	Hybrid_RB	1.148	1.830	1.333	0.493	124
20%	Equi-Hist	1.389	4.137	2.349	167.712	–
	Equi-Width	1.457	539.103	138.760	1.187	–
	Hybrid_Shock	1.197	1.854	1.372	12.594	–
	Hybrid_FT	1.163	1.821	1.354	1.064	873
	Hybrid_RB	1.163	1.815	1.341	0.671	136
25%	Equi-Hist	1.397	4.784	2.630	169.845	–
	Equi-Width	1.497	829.578	217.715	1.258	–
	Hybrid_Shock	1.191	1.843	1.392	13.270	–
	Hybrid_FT	1.156	1.810	1.373	0.891	873
	Hybrid_RB	1.156	1.806	1.363	0.604	125
30%	Equi-Hist	1.404	5.387	6.611	171.035	–
	Equi-Width	1.568	1179.448	312.821	1.143	–
	Hybrid_Shock	1.185	1.814	1.431	13.436	–
	Hybrid_FT	1.152	1.786	1.411	0.910	873
	Hybrid_RB	1.152	1.780	1.395	0.696	146
35%	Equi-Hist	1.412	6.090	8.231	167.918	–
	Equi-Width	1.709	1591.129	426.216	1.202	–
	Hybrid_Shock	1.182	1.800	1.401	14.044	–
	Hybrid_FT	1.149	1.769	1.378	0.799	873
	Hybrid_RB	1.149	1.763	1.343	0.680	142
40%	Equi-Hist	1.419	6.681	6.674	167.547	–
	Equi-Width	2.003	2030.526	548.631	1.108	–
	Hybrid_Shock	1.197	1.773	1.424	14.364	–
	Hybrid_FT	1.161	1.744	1.403	1.010	873
	Hybrid_RB	1.161	1.738	1.389	0.649	163
45%	Equi-Hist	1.428	7.358	4.677	167.580	–
	Equi-Width	2.768	2512.616	691.687	1.184	–
	Hybrid_Shock	1.199	1.745	1.472	14.744	–
	Hybrid_FT	1.163	1.717	1.449	1.006	872
	Hybrid_RB	1.163	1.710	1.437	0.634	156

Continued on next page

Table 4.23 – Continued from previous page

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
50%	Equi-Hist	1.439	8.287	12.608	168.641	–
	Equi-Width	9.333	3077.905	863.072	1.102	–
	Hybrid_Shock	1.194	1.720	1.442	15.164	–
	Hybrid_FT	1.160	1.689	1.419	0.978	871
	Hybrid_RB	1.160	1.684	1.403	0.896	167
55%	Equi-Hist	1.447	8.961	3.988	168.431	–
	Equi-Width	443.249	3707.654	1043.413	1.169	–
	Hybrid_Shock	1.198	1.709	1.402	15.558	–
	Hybrid_FT	1.163	1.677	1.377	0.999	872
	Hybrid_RB	1.163	1.672	1.363	0.970	185
60%	Equi-Hist	1.456	9.812	6.851	168.670	–
	Equi-Width	526.509	4356.086	1247.702	1.085	–
	Hybrid_Shock	1.197	1.668	1.578	15.877	–
	Hybrid_FT	1.163	1.637	1.552	0.982	871
	Hybrid_RB	1.163	1.632	1.540	1.077	208
65%	Equi-Hist	1.465	10.591	4.637	170.063	–
	Equi-Width	614.213	5036.269	1451.129	1.091	–
	Hybrid_Shock	1.203	1.649	1.391	16.506	–
	Hybrid_FT	1.168	1.617	1.363	1.082	871
	Hybrid_RB	1.168	1.610	1.354	0.880	206
70%	Equi-Hist	1.475	11.631	12.176	167.676	–
	Equi-Width	710.195	5769.007	1687.845	1.111	–
	Hybrid_Shock	1.201	1.616	1.505	16.864	–
	Hybrid_FT	1.165	1.583	1.472	1.089	873
	Hybrid_RB	1.165	1.574	1.449	1.062	228
75%	Equi-Hist	1.484	12.507	7.536	168.855	–
	Equi-Width	810.366	6527.897	1931.980	1.035	–
	Hybrid_Shock	1.208	1.565	1.477	17.302	–
	Hybrid_FT	1.171	1.531	1.446	1.266	868
	Hybrid_RB	1.171	1.523	1.433	1.188	258
80%	Equi-Hist	1.494	13.169	23.965	167.604	–
	Equi-Width	921.615	7377.437	2209.731	1.071	–
	Hybrid_Shock	1.205	1.508	1.550	17.541	–
	Hybrid_FT	1.168	1.480	1.518	1.057	867
	Hybrid_RB	1.168	1.472	1.503	1.178	284
85%	Equi-Hist	1.506	14.188	29.200	168.375	–
	Equi-Width	1036.414	8335.168	2482.004	1.018	–
	Hybrid_Shock	1.205	1.457	1.596	18.090	–
	Hybrid_FT	1.168	1.428	1.561	1.077	856
	Hybrid_RB	1.168	1.418	1.546	1.261	292
90%	Equi-Hist	1.519	15.135	20.208	168.315	–
	Equi-Width	1157.655	9174.554	2784.533	0.981	–
	Hybrid_Shock	1.205	1.395	1.751	18.399	–
	Hybrid_FT	1.168	1.366	1.716	1.165	846
Continued on next page						

Table 4.23 – Continued from previous page

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
	Hybrid_RB	1.168	1.358	1.699	1.583	336
95%	Equi-Hist	1.531	16.147	13.211	168.675	–
	Equi-Width	1286.434	10224.030	3108.611	1.039	–
	Hybrid_Shock	1.206	1.306	1.597	18.690	–
	Hybrid_FT	1.168	1.283	1.554	1.130	816
	Hybrid_RB	1.168	1.277	1.549	2.024	417
100%	Equi-Hist	1.542	17.010	23.726	173.151	–
	Equi-Width	1419.415	11232.072	3443.789	0.888	–
	Hybrid_Shock	1.205	1.226	41.660	18.794	–
	Hybrid_FT	1.168	1.188	1.411	1.056	337
	Hybrid_RB	1.168	1.667	1.518	0.709	152

Drifting the workload with the 5% data distribution shifts demonstrates the Hybrid FD CDF estimator robust adaptability that is constantly pulling the median Q-Errors back to the 1.16 while Equi-Width degrades to 1419. The sequential execution incurs the higher latency overhead, proving that the usage of ML and DL models should be paired with batch vectorization.

Table 4.24: Detailed Drift Results: Normal $\rightarrow$ Normal + Zipfian (5% Step, Additive Drift, Workload Normal $\rightarrow$ Zipfian, One-by-One Inference). Best Median Q-Error per distribution is highlighted in bold.

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
0%	Equi-Hist	1.360	2.547	1.591	168.941	–
	Equi-Width	1.409	2.590	1.621	1.297	–
	Hybrid_Shock	1.137	1.927	1.301	10.966	–
	Hybrid_FT	1.137	1.927	1.301	0.000	0
	Hybrid_RB	1.137	1.927	1.301	0.000	0
5%	Equi-Hist	1.368	2.787	2.038	169.303	–
	Equi-Width	1.451	8.006	10.095	1.236	–
	Hybrid_Shock	1.142	2.276	3.840	10.551	–
	Hybrid_FT	1.132	2.194	3.431	0.881	873
	Hybrid_RB	1.116	1.881	1.314	0.681	134
10%	Equi-Hist	1.374	3.070	4.055	168.243	–
	Equi-Width	1.484	128.519	32.557	1.268	–
	Hybrid_Shock	1.160	1.907	1.354	11.884	–
	Hybrid_FT	1.147	1.874	1.341	0.965	873
	Hybrid_RB	1.146	1.865	1.328	0.654	146
15%	Equi-Hist	1.378	3.436	2.857	170.125	–
	Equi-Width	1.523	273.843	69.744	1.213	–
	Hybrid_Shock	1.169	1.877	1.354	12.278	–

Continued on next page

Table 4.24 – Continued from previous page

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
	Hybrid_FT	1.154	1.846	1.340	0.900	873
	Hybrid_RB	1.154	1.840	1.329	0.559	118
20%	Equi-Hist	1.382	3.750	4.237	168.503	–
	Equi-Width	1.560	456.856	116.447	1.259	–
	Hybrid_Shock	1.184	1.868	1.367	12.510	–
	Hybrid_FT	1.161	1.834	1.350	0.998	873
	Hybrid_RB	1.161	1.826	1.337	0.607	129
25%	Equi-Hist	1.389	4.101	5.896	170.705	–
	Equi-Width	1.603	676.183	172.395	1.191	–
	Hybrid_Shock	1.194	1.850	1.384	12.894	–
	Hybrid_FT	1.158	1.818	1.367	0.971	873
	Hybrid_RB	1.158	1.812	1.353	0.732	140
30%	Equi-Hist	1.394	4.449	2.608	168.245	–
	Equi-Width	1.644	918.241	239.515	1.249	–
	Hybrid_Shock	1.197	1.832	1.377	13.141	–
	Hybrid_FT	1.161	1.801	1.358	0.973	873
	Hybrid_RB	1.161	1.795	1.346	0.638	133
35%	Equi-Hist	1.398	4.841	3.018	168.721	–
	Equi-Width	1.693	1223.234	316.723	1.169	–
	Hybrid_Shock	1.187	1.823	1.389	13.410	–
	Hybrid_FT	1.154	1.795	1.371	0.900	873
	Hybrid_RB	1.154	1.791	1.360	0.640	132
40%	Equi-Hist	1.401	5.181	2.702	168.471	–
	Equi-Width	1.733	1476.635	392.855	1.246	–
	Hybrid_Shock	1.200	1.821	1.488	13.557	–
	Hybrid_FT	1.165	1.791	1.466	1.127	873
	Hybrid_RB	1.164	1.791	1.454	0.703	143
45%	Equi-Hist	1.407	5.540	8.281	169.236	–
	Equi-Width	1.791	1833.376	487.000	1.154	–
	Hybrid_Shock	1.187	1.800	1.379	13.769	–
	Hybrid_FT	1.155	1.772	1.359	0.909	873
	Hybrid_RB	1.155	1.767	1.347	0.721	152
50%	Equi-Hist	1.410	5.741	2.668	168.569	–
	Equi-Width	1.842	2148.902	580.365	1.240	–
	Hybrid_Shock	1.194	1.795	1.432	14.074	–
	Hybrid_FT	1.160	1.765	1.409	0.982	873
	Hybrid_RB	1.160	1.754	1.393	0.761	152
55%	Equi-Hist	1.412	6.071	2.984	168.505	–
	Equi-Width	1.891	2500.342	667.910	1.128	–
	Hybrid_Shock	1.197	1.778	1.386	14.197	–
	Hybrid_FT	1.163	1.750	1.364	1.051	873
	Hybrid_RB	1.163	1.743	1.330	0.702	153
60%	Equi-Hist	1.417	6.504	3.566	167.222	–
	Equi-Width	1.957	2877.857	778.782	1.239	–

Continued on next page

Table 4.24 – Continued from previous page

Drift	Model	Median Q	P95 Q	Avg Q	Time (s)	Count
	Hybrid_Shock	1.194	1.779	1.424	14.403	–
	Hybrid_FT	1.160	1.752	1.403	0.962	873
	Hybrid_RB	1.160	1.745	1.388	0.778	139
65%	Equi-Hist	1.421	6.767	3.814	165.731	–
	Equi-Width	2.024	3278.341	882.895	1.110	–
	Hybrid_Shock	1.194	1.763	1.460	14.431	–
	Hybrid_FT	1.160	1.735	1.439	1.013	873
	Hybrid_RB	1.160	1.728	1.425	0.726	154
70%	Equi-Hist	1.421	6.879	9.348	163.477	–
	Equi-Width	2.072	3600.113	985.215	1.266	–
	Hybrid_Shock	1.204	1.758	1.362	14.619	–
	Hybrid_FT	1.169	1.731	1.341	1.366	872
	Hybrid_RB	1.169	1.726	1.329	0.854	158
75%	Equi-Hist	1.426	7.250	25.318	161.463	–
	Equi-Width	2.168	4015.648	1106.383	1.102	–
	Hybrid_Shock	1.198	1.764	1.550	14.832	–
	Hybrid_FT	1.163	1.732	1.528	1.066	872
	Hybrid_RB	1.163	1.724	1.512	0.821	171
80%	Equi-Hist	1.428	7.419	10.978	167.175	–
	Equi-Width	2.294	4499.004	1239.629	1.210	–
	Hybrid_Shock	1.190	1.746	1.448	14.977	–
	Hybrid_FT	1.156	1.714	1.425	0.872	873
	Hybrid_RB	1.156	1.710	1.410	0.802	163
85%	Equi-Hist	1.506	14.188	29.200	168.375	–
	Equi-Width	1036.414	8335.168	2482.004	1.018	–
	Hybrid_Shock	1.205	1.457	1.596	18.090	–
	Hybrid_FT	1.168	1.428	1.561	1.077	856
	Hybrid_RB	1.168	1.418	1.546	1.261	292
90%	Equi-Hist	1.519	15.135	20.208	168.315	–
	Equi-Width	1157.655	9174.554	2784.533	0.981	–
	Hybrid_Shock	1.205	1.395	1.751	18.399	–
	Hybrid_FT	1.168	1.366	1.716	1.165	846
	Hybrid_RB	1.168	1.358	1.699	1.583	336
95%	Equi-Hist	1.531	16.147	13.211	168.675	–
	Equi-Width	1286.434	10224.030	3108.611	1.039	–
	Hybrid_Shock	1.206	1.306	1.597	18.690	–
	Hybrid_FT	1.168	1.283	1.554	1.130	816
	Hybrid_RB	1.168	1.277	1.549	2.024	417
100%	Equi-Hist	1.542	17.010	23.726	173.151	–
	Equi-Width	1419.415	11232.072	3443.789	0.888	–
	Hybrid_Shock	1.205	1.226	41.660	18.794	–
	Hybrid_FT	1.168	1.188	1.411	1.056	337
	Hybrid_RB	1.168	1.667	1.518	0.709	152

The additive experiment in data drift shows that Hybrid model captures the newly

appended Zipf data without forgetting the existing normal distribution, maintaining the Q-Error around 1.16 throughout the transition, however the latency penalty during is imposed and Hybrid finetune + rebuild phases take up to 3-4 times more than Equi-Width.

Chapter 5

Conclusion

The Equi-Width histogram is a good approach that has a severe flaw in design which is the uniformity assumption, while having the $O(1)$ inference time, having ability to do binary search for bucket lookup, if there was a modification upon the Equi-Width that would remove the uniformity assumption thus making the Equi-Width learn about the underlying distribution inside the bucket, then this approach would be superior to the existing methods in industry. In this thesis work, we show that the use of Equi-Width in simulations, that cover the wide range of real life scenarios, is unpredictable and does not perform good on certain datasets which are skewed or sparse. Our proposed approach Hybrid FD CDF estimator is built on top of the Equi-Width to utilize it's ability to do fast inference, we try to put the machine learning algorithms and deep learning models inside the buckets and check the overhead that it adds. The deliberate choose of the bucket size and width removed one hyperparameter and optimized its choice. Then the waterfall model choose policy optimized the choose of the algorithm inside the bucket leading to better training and inference times. The Equi-Hist approach showed that having the high latency makes the approach unusable and not scalable. By utilizing the query to query plan nature we managed to do computations in parallel which helped in decreased inference time. By the experiments results, Hybrid FD CDF approach outperforms the evaluated baselines in almost all tested static and drift scenarios, achieving competitive performance for single attribute range queries. the algorithm of finetuning and

rebuild was introduced that helped in maintaining performance after the data change. The future research directions were identified such as sample reservoir maintenance, model selection policy, training sampling, finetune process optimization, rebuild process optimization, along with the need of the benchmarks that simulate the data drift by checkpoints, since the models like Equi-Hist showed that one process with the update after each query is not able to keep up with the workload under limited budget of memory and time constraints. The end to end DBMS integration is necessary to quantify the true execution time improvements rather than isolated inference benchmarks. Also, future iterations should do more research over the sample reservoir size and confidence thresholds with the usage of broader range of recent learned estimators on more real world datasets.

Bibliography

- [1] Mohammed Al-Kateb and Byung Suk Lee. Stratified reservoir sampling over heterogeneous data streams. *Lecture Notes in Computer Science*, pages 621–639, 2010.
- [2] Béla Bollobás, Gautam Das, Dimitrios Gunopulos, and George Kollios. Selectivity estimation for sparse high-dimensional data. In *Proceedings of the 12th International Conference on Database Theory (ICDT)*, pages 210–221, 2011.
- [3] Nicolas Bruno and Surajit Chaudhuri. Statistics on query expressions in self-tuning database systems. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 163–174, 2002.
- [4] Surajit Chaudhuri, Venkatesh Ganti, and Vivek Narasayya. Exploiting statistics on query expressions for optimization. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, pages 407–418, 2009.
- [5] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. *arXiv preprint arXiv:1612.03079*, 2017.
- [6] Sebastian Michel Damjan Gjurovski, Angjela Davitkova. Grid-ar: A grid-based booster for learned cardinality estimation and range joins. *arXiv preprint arXiv:2405.xxxxx*, 2024.
- [7] Amol Deshpande, Minos Garofalakis, and Rajeev Rastogi. Independence is good: Dependency-based histogram synopses for high-dimensional data. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 199–210, 2001.
- [8] Anshuman Dutt, Chi Wang, Azade Azade, Shibi Nettyam, and Surajit Chaudhuri. Selectivity estimation for range predicates using lightweight models. *Proceedings of the VLDB Endowment*, 12(9):1044–1057, 2019.
- [9] Anshuman Dutt, Chi Wang, Surajit Chaudhuri, and Stavros Galakatos. Efficiently approximating selectivity functions using low overhead regression models. *Proceedings of the VLDB Endowment*, 13(12):2215–2228, 2020.

- [10] David Freedman and Persi Diaconis. On the histogram as a density estimator: L2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476, 1981.
- [11] Dimitrios Gunopulos, George Kollios, Panayiotis Tsaparas, and John Gunopulos. Approximating multi-dimensional aggregate range queries over real attributes. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 463–474, 2000.
- [12] Peter J Haas, Jeffrey F Naughton, S Seshadri, and Lynne Stokes. Sampling-based estimation of the number of distinct values of an attribute. *VLDB Journal*, 4(3):339–373, 1995.
- [13] Max Heimerl, Michael Saecker, Volker Markl, Holger Pirk, and Stefan Manegold. Self-tuning, hardware-accelerated kernel density estimators for multi-dimensional selectivity estimation. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 37–48, 2013.
- [14] Ihab F Ilyas, Volker Markl, Peter J Haas, Paul Brown, and Ashraf Aboulmaga. Corn: Correlation-aware query optimization. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 468–479, 2004.
- [15] Yannis E Ioannidis. The history of histograms (abridged). *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, pages 19–30, 2003.
- [16] Yannis E Ioannidis and Stavros Christodoulakis. Universality of serial histograms. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data*, pages 256–265, 1991.
- [17] Nan Tang Ji Sun, Guoliang Li. Learned cardinality estimation for similarity queries. In *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*, 2021.
- [18] Marcus Kießling et al. Dsb: A decision support benchmark for query optimizers. In *Proceedings of the VLDB Endowment*, 2023.
- [19] Jinho Kim, Jihwan Back, Hyungyu Lee, Juhun Lee, and Young-Ho Park. Astrid: Accurate selectivity estimation for string predicates using deep learning. *Proceedings of the VLDB Endowment*, 14(11):2455–2467, 2021.
- [20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [21] Andreas Kipf, Ryan Walenz, Dominik Varley, Christian Lang, Thomas Neumann, and Alfons Kemper. Learned cardinalities: Estimating correlated joins with deep learning. In *Proceedings of the 9th Biennial Conference on Innovative Data Systems Research (CIDR)*, 2019.

- [22] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. How good are query optimizers, really? In *Proceedings of the VLDB Endowment*, volume 9, pages 204–215, 2015.
- [23] Jialiang Li et al. Speeding up end-to-end query execution via learning-based progressive ce. In *Proceedings of the 2023 ACM SIGMOD International Conference on Management of Data*, 2023.
- [24] Zeyu Li et al. Space: Ce for path queries using cardinality-aware sequence-based learning. In *Proceedings of the 2025 ACM SIGMOD International Conference on Management of Data*, 2025.
- [25] Richard J Lipton, Jeffrey F Naughton, and Donovan A Schneider. Practical selectivity estimation through adaptive sampling. *ACM SIGMOD Record*, 19(2):1–11, 1990.
- [26] Volker Markl, Nimrod Megiddo, Stefan Kutsch, Tuong Truong Tran, and Peter J Haas. Consistent selectivity estimation via maximum entropy. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 1267–1278, 2007.
- [27] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proceedings of the VLDB Endowment*, 2(1):982–993, 2009.
- [28] M Muralikrishna and David J DeWitt. Equi-depth histograms for estimating selectivity of multi-dimensional queries. In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, pages 28–36, 1988.
- [29] Yongjoo Park, Shuxian Zhong, and Barzan Mozafari. Quicksel: Quick selectivity learning with mixture models. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020.
- [30] Tieying Zhang Jianjun Chen Ryan Marcus Zachary G. Ives Peizhi Wu, Rong Kang. Data-agnostic cardinality learning from imperfect workloads. *Proceedings of the VLDB Endowment (VLDB 2025 forthcoming)*, 2025.
- [31] Viswanath Poosala, Yannis E Ioannidis, Peter J Haas, and Eugene J Shekita. Improved histograms for selectivity estimation of range queries. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 294–305, 1996.
- [32] Yuan Qiu, Yilei Wang, Ke Yi, Feifei Li, Bin Wu, and Chaoqun Zhan. Weighted distinct sampling: Cardinality estimation for spj queries. *Proceedings of the 2021 International Conference on Management of Data*, pages 1465–1477, 2021.
- [33] Pascal Richter et al. Automating localized learning for ce (xgboost-based). *Knowledge and Information Systems (KAIS)*, 2024.

- [34] Matteo Riondato, Mert Akdere, Uğur Çetintemel, Stanley B. Zdonik, and Eli Upfal. The vc-dimension of sql queries and selectivity estimation through sampling. *Lecture Notes in Computer Science*, pages 661–676, 2011.
- [35] P Griffiths Selinger, Morton M Astrahan, Donald D Chamberlin, Raymond A Lorie, and Thomas G Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, pages 23–34, 1979.
- [36] Michael Stillger, Guy M Lohman, Volker Markl, and Mohamed Kandil. Leodb2’s learning optimizer. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB)*, pages 19–28, 2001.
- [37] Jan-Sören Strauch, Ju-Ho Kim, and Carsten Binnig. Selectivity functions of range queries are learnable. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD)*, 2022.
- [38] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 7537–7547, 2020.
- [39] Kostas Tzoumas, Amol Deshpande, and Christian S Jensen. Lightweight graphical models for selectivity estimation without independence assumptions. In *Proceedings of the VLDB Endowment*, volume 4, pages 852–863, 2011.
- [40] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.
- [41] Xiaoying Wang et al. Robust query-driven ce under changing workloads. *Proceedings of the VLDB Endowment*, 2023.
- [42] Lucas Woltmann, Claudio Hartmann, Dirk Habich, and Wolfgang Lehner. An empirical analysis of deep learning for cardinality estimation. *arXiv preprint arXiv:1905.06425*, 2019.
- [43] Lucas Woltmann, Claudio Hartmann, Dirk Habich, and Wolfgang Lehner. Deep learning models for selectivity estimation of multi-attribute queries. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020.
- [44] Pengfei Wu et al. Warper: Efficiently adapting learned ce to data & workload drifts. In *Proceedings of the 2022 ACM SIGMOD International Conference on Management of Data*, 2022.
- [45] Pengfei Wu et al. Ceda: Learned ce with domain adaptation. *Proceedings of the VLDB Endowment*, 2023.

- [46] Bowen Yang, Cong Guo, Zhengyi Guan, Kai Zheng, and Xiaofang Zhou. Iam: Unsupervised selectivity estimation by integrating gaussian mixture models and an autoregressive model. In *Proceedings of the 25th International Conference on Extending Database Technology (EDBT)*, 2022.
- [47] Guoliang Li Jianhua Feng Yaoyu Zhu, Jintao Zhang. A lightweight learned cardinality estimation model (code). *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2025.
- [48] Hongzhi Wang Kaixin Zhang Yu Yan Yiming Guan Yingze Li, Xianglong Liu. Updateable data-driven cardinality estimator with bounded q-error. *arXiv preprint arXiv:2403.xxxxx*, 2024.
- [49] Jun Gao Yirui Zhan, Wen Nie. Sscard: Substring cardinality estimation using suffix tree-guided learned fm-index. *arXiv preprint arXiv:2501.xxxxx*, 2025.
- [50] Yuya Sasaki Takahiro Hara Yuchen Ji, Daichi Amagata. Safe: Sampling-assisted fast learned cardinality estimation for dynamic spatial data. In *Proceedings of the 35th International Conference on Database and Expert Systems Applications (DEXA)*, 2024.
- [51] Yan Zhao Jiandong Xie Kai Zeng Kai Zheng Zibo Liang, Xu Chen. Efficient cardinality and cost estimation with bidirectional compressor-based ensemble learning. In *2023 IEEE International Conference on Data Mining (ICDM)*, 2023.