SynopsisLake: Quality-aware Approximate Spatial Query Processing Using Data Synopses

Xin Zhang
University of California, Riverside
Riverside, USA
xzhan261@ucr.edu

Ahmed Eldawy
University of California, Riverside
Riverside, USA
eldawy@ucr.edu

ABSTRACT

Accurate cost estimation is crucial for optimizing spatial queries and for data exploration. Partition-based spatial synopses, such as histograms and sketches, offer greater accuracy than sampling for the same space budget. In data lake systems, which are increasingly adopted for managing large-scale geospatial data, synopses are stored across immutable files. As data grows, these partition-based synopses increasingly overlap and differ in shape, making them inherently unmergeable. This prevents traditional query optimizers from applying standard estimation techniques. In this paper, we present SynopsisLake, a Lakehouse system that enables geospatial query optimization over data lakes. We introduce the Align-Reshape-Merge framework to combine unmergeable spatial synopses and support quality-aware approximate query processing. We also propose Skewness-Align, a metric to evaluate the quality of merged synopses. Experiments on real-world geospatial datasets show that SynopsisLake incurs **less than 10% overhead** during synopsis construction while **reducing total execution time across ingestion and queries by up to 3×** in mixed workload throughput compared to baseline systems.

CCS CONCEPTS

• **Information systems → Data management systems**.

KEYWORDS

Data Lake, Data Lakehouse, approximate geospatial query processing, data synopses combination

ACM Reference Format:
Xin Zhang and Ahmed Eldawy. 2025. SynopsisLake: Quality-aware Approximate Spatial Query Processing Using Data Synopses. In *The 33rd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '25), November 3–6, 2025, Minneapolis, MN, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3748636.3762714

1 INTRODUCTION

Spatial data synopses are compact data models that extract the vital properties of the original data while using less space [23]. They are the key component to provide approximate answers in the query optimizers of spatial database management systems (DBMSs). Overall, there are four categories of data synopses: samples [4, 49, 70, 78], histograms [6, 8, 18, 59, 63], wavelets [20, 52, 68, 71], and sketches [26, 35, 50, 51, 57, 65, 77]. They can handle cardinality estimation for range selectivity and join selectivity queries for spatial vector data. Recently, data synopses have also become a default block in the Apache Puffin [13], which is a data lake file format designed to store information such as statistics and indexes about data managed in an Apache Iceberg [10] table. Due to its distributed nature, data lake systems and big data DBMSs, partition data in separate files, each having its own data synopsis during the data loading phase. In the query phase, multiple synopses need to be merged together to provide approximate answers to estimate the query cost for the query optimizer.

Agarwal et al. define synopses mergeability [3] as the ability to merge two source synopses into a single synopsis while preserving error and size guarantees similar to the source synopses. They show that all linear function-based sketches, such as the Count-Min sketch [25] and KLL sketch [77], are fully mergeable. In contrast, widely used partition-based synopses, such as histograms [6, 8, 18, 59, 63], and hash function based synopses, such as spatial sketch [26] and Hyper-Log-Log (HLL) sketch [34, 41] are mergeable under the following stringent conditions. Specifically, partition-based synopses must be built using the *same predefined partitioning function*, and hash-based synopses must be created using the *same set of hash functions*. In other words, two histograms can only be merged if they are perfectly aligned. These constraints are often impractical in real-world settings. In geospatial data management, partitioning is fundamental, and data across different regions often follow diverse distributions. Even within the same region, data files may overlap in their value ranges. Using different partitioning functions tailored to each file can better summarize the underlying data. Unfortunately, the **inability to merge spatial partition-based synopses** restricts the seamless integration of geospatial data applications into modern data lake systems and large-scale DBMSs.

Our goal is to address the limitations of combining and querying *unmergeable spatial synopses*. When new datasets are inserted, a straightforward approach is to reload existing data files and construct new, unified synopses. However, in data lake systems and large-scale DBMSs such as LSM-tree-based systems [5], data files are immutable. Frequently reloading and rewriting these files to update synopses blocks can significantly degrade system throughput.

Without rebuilding unified synopses, we cannot support the full mergeability property. To address this, we introduce the concept of **partial mergeability**, which preserves the size guarantee of merged synopses, while explicitly quantifying the loss of error guarantees. We identify two major challenges that must be addressed: First, as data volume grows, how can we merge unmergeable synopses while preserving both quality and size constraints? Second, how can we define and evaluate the quality of approximate answers provided by partition-based synopses? Throughout this paper, we use histograms

ACM ISBN 979-8-4007-2086-4/2025/11.
https://doi.org/10.1145/3748636.3762714

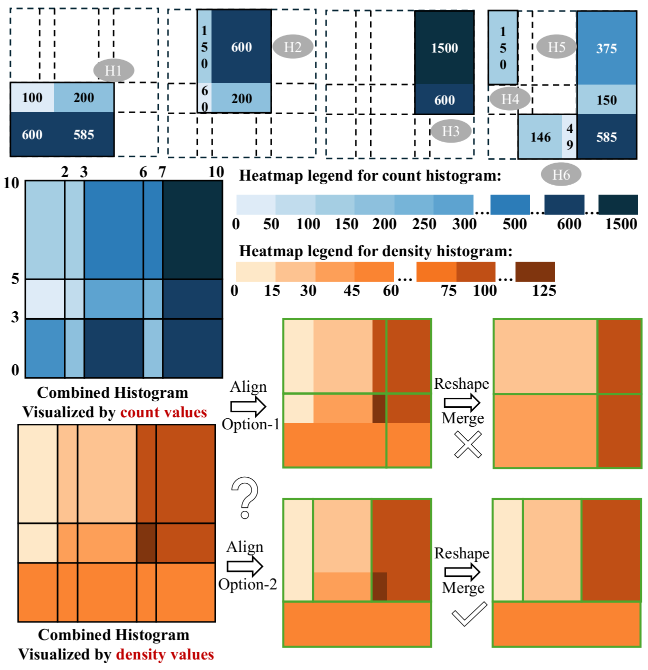


Figure 1: Two options to combine six source histograms into a single merged histogram with four buckets.

as a representative example of partition-based spatial synopses to anchor our discussion.

In this paper, we propose an **Align-Reshape-Merge framework** to address the mergeability challenge. This framework merges partition-based synopses and supports the computation of approximate answers from the merged synopsis. Within this framework, we define optimal alignment problems tailored to different merging objectives and introduce practical strategies for combining various types of statistical information.

To tackle the challenge of quality estimation, we make two key contributions. First, we extend existing histograms to include density statistics, introducing only minimal computational overhead and no additional storage cost. Second, we propose a new metric called **Skewness-Align**, which estimates the error ratio of cardinality estimations produced by histograms. This metric quantifies the skewness of the underlying data distribution using the newly introduced density statistics. For a given range query, the Skewness-Align metric applies a statistical model to estimate the expected error ratio. During synopsis reshaping, we use this metric to track accuracy loss, allowing us to quantify and bound the degradation in error guarantees. This enables our system to support the partial mergeability property in a principled way.

Figure 1 shows how our Align-Reshape-Merge framework tackles the challenge of combining overlapping 2D count histograms. Each source histogram (H_1 to H_6) summarizes a spatial region using count-based buckets. We enhance these buckets with a density statistic (count divided by area) which we show in this paper to be more effective in merging histograms. By aligning all bucket boundaries, we create a unified 15-cell grid and generate a combined histogram, visualized by count (blue) and density (orange), with darker shades indicating higher values. Our goal is to produce a merged histogram with four buckets. If we follow the count histogram (blue), to find the boundaries of the merged histogram, we will have the solution highlighted as Option 1 which results in poor estimation of selectivity queries due to the discrepancy in distribution within cells. On the other hand, the density histogram (orange) reveals the difference in skewness and allows us to find the merged histogram in Option 2 which merges the histogram cells into more uniformly distributed buckets. Option 2 better preserves spatial skew, demonstrating the advantage of using enhanced statistics to improve the quality of the merged histogram.

Extensive experiments on large-scale real data demonstrate that our approach maintains high-quality synopses and delivers accurate approximate query results. In summary, this paper makes four key contributions. First, we propose *Skewness-Align*, a statistical metric for evaluating the quality of approximate answers produced by spatial histograms (Section 3). Second, we design an *Align-Reshape-Merge framework* to combine spatial synopses. We formulate two histogram reshaping problems (data-driven and query-driven) and propose efficient greedy algorithms to solve them. We also extend our approach to support the combination of samples and spatial sketches (Section 4). Third, we present SynopsisLake, a Lakehouse architecture that integrates synopsis storage and indexes into the metadata and indexing layers (Section 2). Finally, we validate the performance of SynopsisLake on real-world datasets through extensive experiments (Section 5).

2 SYNOPSISLAKE

2.1 SynopsisLake Architecture

Figure 2 shows the architecture of SynopsisLake. It extends from an existing data Lakehouse [15, 17] which includes three layers and a query engine. The three layers are the metadata, indexing, and caching layers. The data Lakehouse is built on top of the data lake.

Metadata layer, indexing layer, and caching layer. The metadata layer of Lakehouse stores files with a standard file format, such as Apache Parquet [12]. The goal of this layer is to add management capabilities such as ACID transactions for data lakes [15]. SynopsisLake stores the spatial synopses in the metadata layer to process approximate spatial queries. The data synopses are constructed when the data files are loaded into the data lake. The goal of the indexing and caching layers of Lakehouse is to optimize the query performance. SynopsisLake maintains one map, stored in memory, to track the data range of each synopsis. The merged synopses reside in the caching layer. During query time, if no cached merged synopses exist, SynopsisLake will load unmerged synopses from the metadata layer, compute merged synopses, and store them in memory for future queries. The merged synopses are only computed once if no new datasets are inserted into the data lake.

File Format and Storage Design. Parquet [12] and ORC [11] are two widely used columnar data storage file formats for storing data in data lake systems such as Delta Lake [14], Apache Iceberg [10], and Apache Hudi [9]. Following a similar design, SynopsisLake stores data synopses in Parquet files.

A synopses Parquet file includes two groups of columns: *synopsis information columns* and *synopsis content columns*. Each file can store multiple synopses. The synopsis information columns record metadata about the summarized data and the parameters used to

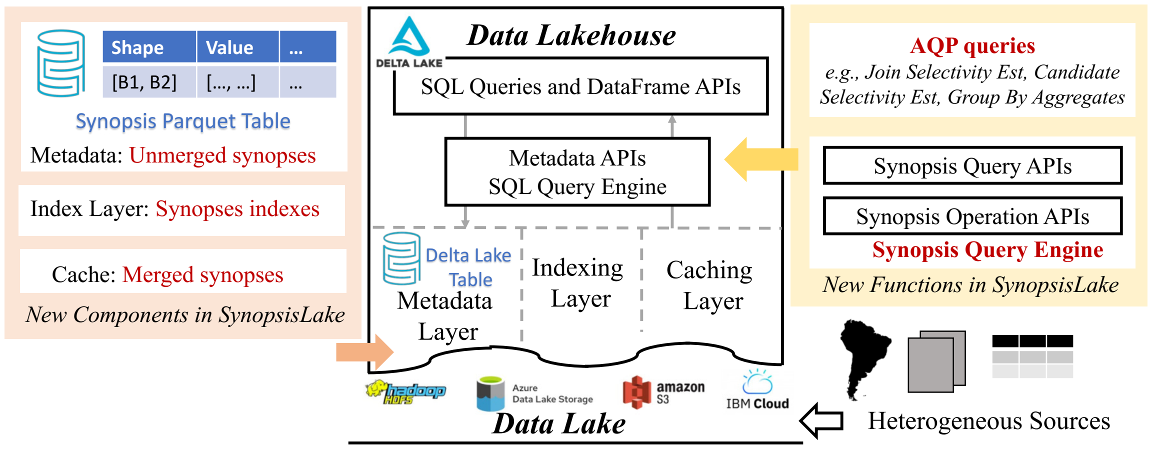


Figure 2: The architecture of SynopsisLake.

construct the synopsis. These include the raw data file path in the data lake, the data range covered, file ID, synopsis type, total number of summarized items, histogram partition function, and sample ratio. The synopsis content columns store the statistical information contained in each synopsis.

2.2 Query Engine

Table 1: The supported data synopses and AQP queries

Synopsis	Supported AQP Query
1D/2D Count Spatial Histogram	RangeScanEst, Clustering
Geometric Histogram[8]	SpatialJoinEst
Spatial Sketches[26]	SpatialJoinEst
Uniform and Stratified Samples	RangeScanEst, SpatialJoinEst, Clustering and GroupByAggregatesEst
1D/2D Wavelets	Store and merge only

Table 1 summarizes the synopses and queries that are supported by the current version of SynopsisLake. It includes both partition-based and function-based synopses. SynopsisLake can process the following types of AQP queries: candidate selectivity estimation, spatial join selectivity estimation, group by aggregates estimation, and approximate clustering. SynopsisLake is not limited to geospatial query processing. For example, it also supports equi-join selectivity estimation using samples. Additional query types can be supported in the future.

Flowchart to Process Queries. Figure 3(a) illustrates how SynopsisLake processes AQP queries. It begins by parsing the query and selecting appropriate synopses. Users can specify the synopsis type within query or allow SynopsisLake to automatically select from the available synopses. SynopsisLake identifies the candidate synopses based on the synopses index and query range. If merged synopses are cached in memory, SynopsisLake uses them to compute approximate answers. Otherwise, it accesses the unmerged synopses from the metadata layer and combines them into merged synopses. Along with approximate answers, SynopsisLake also provides the quality for different estimation types. For samples-based synopses, it returns a confidence interval. For sketches, it uses sketch-specific quality metric. For partition-based synopses, it introduces a novel statistics-based metric described in Section 3.

Synopses Combination Process. Figure 3(b) shows how SynopsisLake combines data synopses. It loads the unmerged synopses from the metadata layer and computes the merged synopses through the three-step synopses combination framework. The three steps are Alignment, Reshaping, and Merging. This framework is called the Align-Reshape-Merge framework.

In the Alignment step, the framework repartitions the entire data range into a set of adjacent smaller ranges. In the Reshaping and Merge steps, the framework computes merged synopses based on each of these smaller ranges. The framework also provides different strategies to combine count-type and ratio-type statistical summaries. The final step involves merging the reshaped synopses within each partition. Since the Alignment step divides the entire data range into adjacent non-overlapping ranges, the reshaping and merging steps can be executed in parallel across all ranges. The detailed descriptions of the Align-Reshape-Merge framework are in Section 4.

3 SKEWNESS-ALIGN METRIC

SynopsisLake combines multiple partition-based synopses computed from different partition functions. The key operation is to reshape them into a common partitioning so they can be merged directly. We call this operation *Alignment*. The main challenge lies in determining the optimal new partitioning that maximizes the accuracy of the merged synopses. In this section, we define a quality metric to evaluate the accuracy of an alignment operation. To simplify the discussion, we use one-dimensional count histograms to describe the definitions and functions.

3.1 Preliminaries

DEFINITION 3.1 (ONE-DIMENSIONAL DATA POINT AND DATASET). *In a one-dimensional data space, a data point p is represented by a numerical value. A dataset P containing N data points is denoted as $P=\{p_1, \cdots, p_N\}$.*

DEFINITION 3.2 (ONE-DIMENSIONAL BUCKET AND ONE-DIMENSIONAL RANGE). *A one-dimensional bucket B is represented by an interval $[l, h)$, where l and h are the lower and upper bounds, and its length is defined as $len_B=h-l$. A one-dimensional range R is*

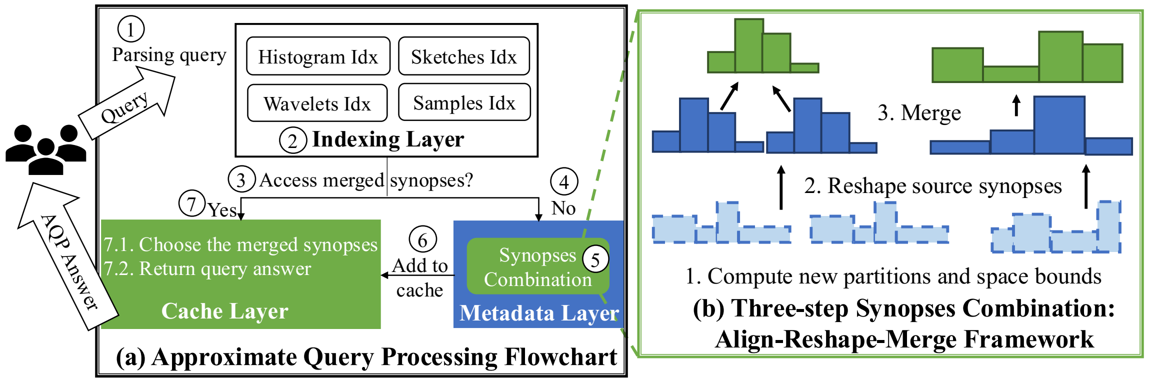

① Parsing query
Query
Histogram Idx
Sketches Idx
Wavelets Idx
Samples Idx
② Indexing Layer
③ Access merged synopses?
⑦ Yes
④ No
7.1. Choose the merged synopses
7.2. Return query answer
Cache Layer
⑥ Add to cache
Synopses Combination ⑤
Metadata Layer
AQP Answer
(a) Approximate Query Processing Flowchart
3. Merge
2. Reshape source synopses
1. Compute new partitions and space bounds
(b) Three-step Synopses Combination: Align-Reshape-Merge Framework

Figure 3: The query process of SynopsisLake.

similarly defined. We use the term bucket *when referring to histogram units and* range *when referring to arbitrary intervals.*

Definition 3.3 (Bucket value). *The bucket value v_B is the number of data points that fall within the interval $[l, h)$.*

Definition 3.4 (Bucket density). *The bucket density d_B is the ratio of bucket value v_B to its length. Given a one-dimensional histogram bucket $B = [l, h)$ with length len_B=$h - l$, the density is d_B=$\frac{v_B}{h-l} = \frac{v_B}{len_B}$.*

3.2 Histogram Querying Process

Definition 3.5 (Histogram). *A histogram H summarizes a dataset using a set of adjacent buckets $\mathcal{B}$. It is defined as H=$(\mathcal{B}, \mathcal{V})$, where $\mathcal{V}$ denotes the buckets values. Assuming H consists of n buckets, $\mathcal{B} = [b_1, \cdots, b_n]$ and $\mathcal{V} = [v_1, \cdots, v_n]$. We can also represent the histogram using bucket densities $[d_1, \cdots, d_n]$, where d_i=v_i/len_{b_i}.*

Given a range query Q that overlaps with a bucket B, the approximate query answer from B is computed as $v_B \cdot \frac{len_{OQ}}{len_B}$=$len_{OQ} \cdot d_B$, where v_B, len_B, and d_B are the bucket value, length, and density, respectively, and len_{OQ} is the overlap length between Q and B.

We extract two observations from the histogram querying process: (1) **Equivalence of Buckets:** Given two buckets, regardless of which histogram they belong to, if they have the same density and the same query overlap length, they produce the same approximate range query answer. (2) **Uniform Density Assumption:** Histogram estimation assumes that the proportion of the bucket's value is equal to the proportion of its length that overlaps with the query. In other words, *all sub-ranges within a bucket are assumed to have uniform density*. A concrete example illustrating this assumption is provided in Appendix 8.1.1.

Example 3.1. *Figure 4 illustrates two 1D spatial regions with the same data range $[0, 45]$ but different data skewness. Each region is summarized using two 1D count histograms: one with 9 equal-width buckets $0, 5, \cdots, 45$, and one with 3 equal-width buckets $0, 15, 30, 45$. For example, the first bucket b_1 of Count Hist 1 contains 180 points: $b_1 = [0, 5)$, $v_{b_1} = 180$, and $d_{b_1} = 180/5 = 36$. Following the color-code in Figure 1, count histograms are shown in blue and density histograms in orange, with darker shades indicating denser regions.*

We evaluate three range queries Q_1, Q_2, and Q_3. Ground truth answers are computed using the 9-bucket histograms, while estimates are obtained from the 3-bucket histograms. In (b), for instance, $Q_3 = [5, 20]$ overlaps with $b_1 = [0, 15)$ and $b_2 = [15, 30)$ in Count Merge A. The estimate answer $ans_3 = 140 \times \frac{10}{15} + 550 \times \frac{5}{15} \approx 277$.

3.3 Histogram Query Accuracy Model

To address the limitations of histogram-based range query estimation, we propose a statistical model that quantifies the expected error of an approximate query answer. Based on motivation experiments (see Figure 10in Appendix 8.1.2) demonstrate how accuracy is affected by two key factors: (1) **Data Skewness:** Histograms yield accurate estimates when summarizing uniformly distributed data, but their accuracy deteriorates in the presence of skewed distributions. (2) **Overlap Ratio:** The more a query overlaps with a bucket, the more accurate the estimate. Even with extreme skewness, a query that perfectly aligns with a bucket returns an exact answer. We defer a detailed discussion and illustrative examples to Appendix 8.1.2.

Definition 3.6 (Data Skewness via Median Absolute Deviation). *Given a histogram H with bucket densities $[d_1, \cdots, d_n]$, let $\tilde{d}$ be the median of the densities. We define the* median absolute deviation (MAD) *as:*

$$MAD_H = median(|d_i - \tilde{d}|). \tag{1}$$

The data skewness DK_H *of the histogram is then defined as:*

$$DK_H = \frac{MAD_H}{\tilde{d} + \varepsilon}, \tag{2}$$

where ε is a small positive constant to prevent division by zero.

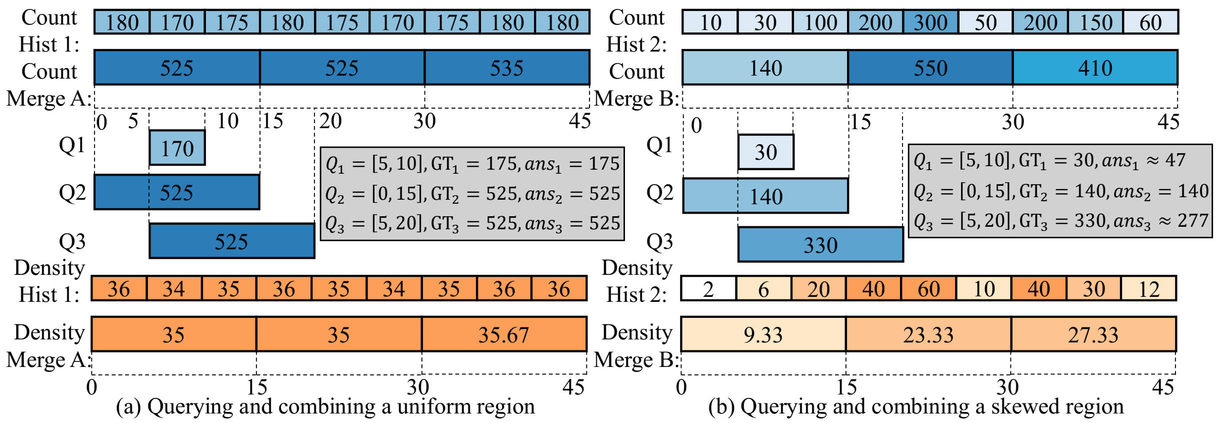
Intuitively, a larger DK_H indicates a more skewed data distribution, while a value close to zero indicates uniformity.

Example 3.2. *Figure 4 illustrates how to compute data skewness from histogram bucket densities.*

For Merged A, the sorted densities are $H_{MA} = \{35, 35, 35.67\}$ with median $\tilde{d}_{MA} = 35$. The absolute deviations are $\{0, 0, 0.67\}$, so $MAD_{MA} = 0$, yielding $DK_{MA} = \frac{0}{35+10^{-2}} = 0$, following Equation 1.

Using the same method: (1) Hist 1: $\tilde{d} = 35$, $MAD = 0$, $DK = 0$ (2) Merged B: $\tilde{d} = 20$, $MAD = 14$, $DK \approx 0.70$ (3) Hist 2: $\tilde{d} = 23.33$, $MAD = 4$, $DK \approx 0.17$

Both Hist 2 and Merged B summarize skewed regions, but Hist 2 better captures skewness due to its finer granularity.


Count Hist 1: 180 170 175 180 175 170 175 180 180
Count Merge A: 525 525 535
0 5 10 15 20 30 45
Q1 170
Q2 525
Q3 525
Q1 = [5, 10], GT1 = 175, ans1 = 175
Q2 = [0, 15], GT2 = 525, ans2 = 525
Q3 = [5, 20], GT3 = 525, ans3 = 525
Density Hist 1: 36 34 35 36 35 34 35 36 36
Density Merge A: 35 35 35.67
0 15 30 45
Count Hist 2: 10 30 100 200 300 50 200 150 60
Count Merge B: 140 550 410
0 15 30 45
Q1 30
Q2 140
Q3 330
Q1 = [5, 10], GT1 = 30, ans1 ≈ 47
Q2 = [0, 15], GT2 = 140, ans2 = 140
Q3 = [5, 20], GT3 = 330, ans3 ≈ 277
Density Hist 2: 2 6 20 40 60 10 40 30 12
Density Merge B: 9.33 23.33 27.33
0 15 30 45

(a) Querying and combining a uniform region (b) Querying and combining a skewed region

Figure 4: Examples of combining and querying uniform and skewed regions.

We aim to model the error of approximate query answers using data skewness and the overlap ratio between the query and histogram bucket. To capture this relationship, we adopt the beta distribution for two reasons: (1) It naturally models proportions in the interval $[0, 1]$, aligning with the overlap ratio $\frac{len_Q}{len_B}$. (2) Its shape can be adjusted using skewness, allowing us to express varying levels of estimation uncertainty.

We assume $\alpha = \beta$ due to lack of prior knowledge of the intra-bucket distribution. A full discussion of this modeling choice is provided in Appendix 8.1.3.

Definition 3.7 (Beta distribution model of a histogram). *Given a histogram H, we define its beta distribution shape parameters:*

$$\alpha = \beta = \frac{k}{DK_H + \varepsilon} + 1, \tag{3}$$

where DK_H is the data skewness, $k > 0$ is a scaling parameter, and ε is a small constant for numerical stability. A detailed discussion of the mapping function's behavior in Equation 3 is available in Appendix 8.1.4.

Since $\alpha = \beta$, the distribution is symmetric with a peak at:

$$p = \frac{\alpha - 1}{\alpha + \beta - 2} = 0.5. \tag{4}$$

The peak p in PDF_H [72] is $p = \frac{\alpha-1}{\alpha+\beta-2}$. Since we let α=β in our model, the peak value is always 0.5.

We now define how this distribution is used to estimate the quality of a histogram-based query answer.

Definition 3.8 (Quality of the approximate query answer). *Let a histogram H have beta distribution $PDF_H = f(x; \alpha, \beta)$ and peak $p = 0.5$. For a range query $Q = [l_Q, h_Q]$ overlapping a bucket $B_i = [l_{B_i}, h_{B_i}]$ with value v_{B_i} and length len_{B_i}, the approximate answer is:*

$$ans_Q = v_{B_i} \cdot \frac{len_{OQ}}{len_{B_i}} = d_{B_i} \cdot len_{OQ}, \tag{5}$$

where len_{OQ} is the overlap between Q and B_i, and d_{B_i} is the bucket density. The estimated error ratio between ans_Q and v_{B_i} as:

$$r_{B_i \to Q} = 1 - \frac{P(a < X < b)}{P(0 \le X \le 1)} = 1 - \frac{\int_a^b f(x; \alpha, \beta)\, dx}{\int_0^1 f(x; \alpha, \beta)\, dx}$$
$$= 1 - \int_a^b f(x; \alpha, \beta)\, dx = 1 - I_b(\alpha, \beta) + I_a(\alpha, \beta) \tag{6}$$

where $I_x(\alpha, \beta)$ is the beta CDF. The overlap interval $[a, b]$ is computed as:

$$a = p - \frac{1}{2} \cdot \frac{len_{OQ}}{len_{B_i}}, \quad b = p + \frac{1}{2} \cdot \frac{len_{OQ}}{len_{B_i}}. \tag{7}$$

If query Q overlaps m buckets $\{b_1, \ldots, b_m\}$, the overall estimated error is a weighted sum:

$$r_{\{b_1,\cdots,b_m\} \to Q} = \sum_{i=1}^{m} o_i \cdot r_{b_i \to Q}, \quad o_i = \frac{len_{oi}}{len_Q}, \tag{8}$$

where len_{oi} is the overlap between Q and bucket b_i.

Example 3.3. *We evaluate queries $Q_1 = [5, 10]$, $Q_2 = [0, 15]$, and $Q_3 = [5, 20]$ on Merge A and Merge B (3-bucket histograms in Figure 4).*

From Example 3.2, Merge A has skewness $DK = 0$, yielding $\alpha = \beta = 52.23$. Merge B has $DK \approx 0.17$, so $\alpha = \beta \approx 6.51$ (assuming $k = 1$, $\varepsilon = 10^{-2}$).

Using Equation 5, Merge B returns estimated answers: - $ans_1 = 47$, $ans_2 = 140$, $ans_3 = 277$.

For Q_1, the overlap is $\frac{5}{15}$, so by Equation 7, $a = 0.5 - 0.5 \cdot \frac{5}{15} \approx 0.33$, $b \approx 0.67$. Using Equation 6, error ratio $\approx 1 - \int_{0.33}^{0.67} f(x; \alpha, \beta) dx \approx 0.22$.

For Q_2, the overlap equals the full bucket → error ratio = 0. For Q_3, overlapping two buckets, we apply Equation 8:

$$r_{Q_3} \approx \frac{10}{15} \cdot r_{b_1 \to Q_3} + \frac{5}{15} \cdot r_{b_2 \to Q_3} \approx 0.08.$$

Merge A's high $\alpha = 52.23$ yields near-zero errors for all three queries: $r_{Q_1} = 0.0005$, $r_{Q_2} = 0$, $r_{Q_3} = 0.0002$.

4 ALIGN-RESHAPE-MERGE FRAMEWORK

This section introduces how each step of the Align-Reshape-Merge framework works in detail.

4.1 Alignment Step

The alignment step splits the one-dimensional data domain into a set of adjacent intervals. Existing methods determine interval boundaries based on either the underlying data distribution (data-driven) or expected query workload (query-driven). Data-driven approaches [2, 27, 44, 45], such as V-optimal histograms, aim to preserve statistical accuracy within each bucket. Query-driven approaches [6, 24, 69] instead optimize boundaries to improve the accuracy of query results. However, neither approach supports the problem of combining precomputed, possibly overlapping synopses into a unified representation.

We consider a generalized setting where the input is a collection of overlapping histograms $\mathcal{H}$ defined on n source ranges $\{s_1, ..., s_n\}$. Given a space constraint m, the goal is to reshape the histograms into m non-overlapping, adjacent buckets. We extend data-driven and query-driven techniques to this merging setting and define new optimization formulations accordingly.

The data domain can be partitioned in infinitely many ways under a space constraint. To bound the search space, we define *canonical ranges*, which are the minimal disjoint intervals formed by source range boundaries. The alignment problem then reduces to optimally grouping these canonical ranges within the space limit.

Definition 4.1 (Canonical Ranges U). *Given overlapping source ranges, the canonical ranges $U = u_1, \ldots, u_s$ are the minimal set of adjacent, non-overlapping intervals that partition the union of all source boundaries.*

To construct U, we union and sort all boundaries from source ranges $s_1, \ldots, s_n$; each adjacent pair defines a canonical range. These ranges subdivide the domain into atomic units that fully preserve boundary transitions. Since each $u_j \subseteq s_i$ for some s_i, the alignment error $r_{s_i \to u_j}$ is zero, ensuring a lossless initialization for the alignment process.

4.1.1 Data-driven Histogram Optimal Reshaping.

Definition 3.8 shows how data skew and overlap affect histogram-based query accuracy. The data-driven reshaping problem considers two objectives: (1) *Statistical accuracy*: preserving values from the original histograms; and (2) *Density preservation*: avoiding the merging of ranges with differing densities.

Definition 4.2 (Data-driven histogram optimal reshaping problem). *Given unmerged source ranges $\{s_1, ..., s_n\}$, space budget m, skewness scale $k > 0$, and error smoothing $\varepsilon > 0$, the goal is to generate m non-overlapping buckets $\{b_1, \cdots, b_m\}$ that minimize the total alignment cost:*

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} \big(\alpha \cdot x_i + (1-\alpha) \cdot y_i\big) \\ \text{s.t.} \quad & x_i = \sum_{b_i \cap s_j \neq \emptyset} r_{s_j \to b_i}, \\ & y_i = \sum_{s_j \cap b_i \neq \emptyset \, \wedge \, s_{j'} \cap b_i \neq \emptyset} \big|Diff(d_j, d_{j'})\big|, \end{aligned} \tag{9}$$

where $r_{s_j \to b_i}$ is the alignment error between source range s_j and reshaped bucket b_i, computed using Definition 6. x_i captures statistical accuracy; y_i measures the density mismatch among overlapping ranges in b_i. The weight $\alpha \in [0, 1]$ balances the two terms.

Algorithm 1: Optimal Alignment

```
Input: Source ranges $\{s_1, \ldots, s_n\}$, target size $m$, objective function $F$
Output: Reshaped buckets $\{b_1, \ldots, b_m\}$
1  Compute canonical ranges $\{u_1, \ldots, u_k\}$ from source range boundaries;
2  Initialize priority queue $Q$ with merge positions $(u_i, u_{i+1})$ scored by $F$;
3  $numBuckets \leftarrow |u|$;
4  while $numBuckets > m$ do
5      Pop position $p$ with minimal score increase;
6      if $p$ is still valid then
7          Merge left $u_l$ and right $u_r$ into $u_{merged}$;
8          Update $Q$: re-score $p$'s left neighbor by $u_{merged}$'s left range and $u_{merged}$;
9          Update $Q$: re-score $p$'s right neighbor by $u_{merged}$ and $u_{merged}$'s right range;
10         $numBuckets \leftarrow numBuckets - 1$;
11     end
12 end
13 Construct $\{b_1, \ldots, b_m\}$ from remaining ranges;
14 return $\{b_1, \ldots, b_m\}$;
```

4.1.2 Query-driven Histogram Optimal Reshaping.

The query-driven reshaping problem incorporates workload information into the alignment process. Given a set of z range queries $Q = \{Q_1, ..., Q_z\}$, we treat the approximate answers $\widehat{ans_Q}$ computed from the source histograms $\mathcal{H}$ as ground truth. The objective is to reshape the histograms into m buckets that minimize the error in query answers.

Definition 4.3 (Query-driven histogram optimal reshaping problem). *Given a set of unmerged source histograms $\mathcal{H}$, a query workload $Q = \{Q_1, ..., Q_z\}$ with corresponding ground-truth answers $\widehat{ans_Q} = \{\widehat{ans_{Q_1}}, ..., \widehat{ans_{Q_z}}\}$, and a space budget m, the goal is to find m adjacent, non-overlapping buckets $\{b_1, ..., b_m\}$ that minimize:*

$$\min \quad \sum_{i=1}^{z} \left| Diff(\widehat{ans_{Q_i}}, ans_{Q_i}) \right|, \tag{10}$$

where $ans_{Q_i} = \sum_{b_j \cap Q_i \neq \emptyset} (d_{b_j} \cdot len_{OQ})$, computed as in Definition 3.8.

4.2 Merging Algorithm

To solve both data-driven and query-driven reshaping problems, Algorithm 1 uses a greedy bottom-up strategy that merges canonical ranges into m buckets minimizing a user-defined objective function F (see Equations 9 and 10).

The algorithm begins by computing the set of canonical ranges $u_1, \ldots, u_k$, which are non-overlapping intervals derived from the boundaries of the source ranges. This transformation ensures a finite and structured search space for merging.

Each adjacent pair (u_i, u_{i+1}) defines a candidate merge position. These are stored in a priority queue Q ordered by the marginal increase in F if merged. Since standard priority queues do not support in-place priority updates, we implement a custom binary-heap-based queue with a hash map for position tracking (details in Appendix 8.2.1). This design enables efficient re-scoring of neighboring positions after each merge step.

At each step, the algorithm pops the merge with the smallest score increase. If the corresponding pair remains unmodified in Q, they are merged, and neighboring positions are re-scored based on the new range. This ensures consistency between Q and the current partitioning state.

The merging process repeats until only m buckets remain. The final output is constructed from the remaining ranges.

Time complexity. Let n be the number of source endpoints. Canonical range construction takes $O(n \log n)$ due to sorting. The queue contains $O(n)$ entries, and each merge and update step runs in $O(\log n)$, giving an overall time complexity of $O(n \log n)$. Updating left/right neighbors incurs constant time per update, amortized over all merge steps.

4.2.1 Multi-dimensional Alignment. In multi-dimensional settings, maintaining neighbor relationships during merging or splitting is more complex, and the extensions are discussed in Appendix 8.2.3. To simplify the process, we partition each dimension independently and construct the final multi-dimensional partitions as the Cartesian product of the one-dimensional partitions. This design enables the direct application of our data-driven and query-driven reshaping formulations to each dimension. We leave the development of a fully multi-dimensional merging algorithm for future work.

4.3 Reshaping and Merging Steps.

To compute reshaped histograms, we extend the transformation function proposed by Singla et al. [61], originally designed for count-based summaries. It computes the reshaped value v_{reshaped} by scaling the source value v_{source} using the ratio of the overlapping length to the source range: $v_{\text{reshaped}} = v_{\text{source}} \times (\text{overlap}/\text{source range})$.

However, certain histograms, such as the Geometric Histogram [8] used in spatial join estimation, store ratio-based summaries. For these, we introduce a transformation that scales by the inverse ratio: $v_{\text{reshaped}} = v_{\text{source}} \times (\text{source range}/\text{reshaped range})$.

Aligning non-uniform Geometric Histograms is expensive. To address this, we reshape them into adjacent, non-overlapping histograms using uniform grids. Rather than preserving original bucket boundaries, we derive canonical ranges from their covered data domains and apply the same alignment strategy used for count-based histograms. Space budgets are allocated proportionally to data coverage, and each merged histogram is rebuilt using a uniform grid. Additional reshaping strategies for Spatial Sketches and samples are described in Appendix 8.2.2.

5 EXPERIMENT

5.1 Experiment Setting

We compare SynopsisLake with two baselines: EagerCalc and LazyCalc. Experiments run on a cluster with 1 head node: Xeon E5-2609 v4 @ 1.70GHz and 12 workers nodes: Xeon E5-2603 v4 @ 1.70GHz, 12 cores each; 144 cores total) using HDFS. SynopsisLake is implemented in Java on Spark 3.0.1, and the source code is available at: https://github.com/xin-aurora/SynopsisLake.

Table 2: Scalability: loading thousands of datasets.

#OfDatasets	#OfPoints	NoSynopsis	WithSynopsis	Overhead
1089	1.6 B	0.484 h	0.510 h	6.372 %
2182	3.3 B	0.917 h	0.956 h	4.25 %
3253	4.9 B	1.374 h	1.425 h	3.371 %
4385	6.6 B	1.761 h	1.912 h	8.364 %

Table 3: Overhead: creating different synopses files.

Synopsis Type	Space Overhead		
	0.131 MB	0.524 MB	2.097 MB
2D Count Histogram	5.150 %	6.122 %	12.880 %
2D Wavelets	5.798 %	14.609 %	14.858 %
Geometric Histogram	7.281 %	12.927 %	12.927 %
Spatial Sketches	20.054 %	25.149 %	27.751 %
Uniform Samples	2.825 %	3.111 %	8.883 %

Datasets. We use four real-world spatial datasets, grouped into two categories: (1) Point datasets: OSM21/pois [75] with 147 million 2D records and OSM15/All [31] with 2.6 billion 2D records; (2) Polygon datasets: OSM15/lakes [32] with 7.5 million records and OSM15/parks [33] with approximately 10 million records. Points test histograms, wavelets, samples, and sketches; polygons test Geometric Histograms and spatial sketches. We simulate data lake ingestion workloads by splitting each dataset into thousands of files; for reading, we split the tested datasets into 20–100 parts, following prior benchmarks [28, 30, 56, 62, 74].

Evaluation metric. We report the latency as the query efficiency measurements, which is the average time to answer the AQP queries. We also report the total processing time to process read&write mixed workloads.

Query accuracy is measured by average relative error (ARE), which is also considered as the quality metric in other AQP works [21]. We also report average absolute error (ABS) to measure the accuracy of the merged synopses.

Experiment Plan. In the following experiments, we compute **2D data synopses** to evaluate the performance of SynopsisLake.

In Section 5.2, we evaluate the performance of SynopsisLake through three experiments: (1) scalability in loading thousands of datasets and the overhead of writing synopses; (2) synopsis construction time under a fixed space budget; and (3) performance under mixed read&write workloads, compared to EagerCalc and LazyCalc. In Section 5.3, we measure the AQP performance of SynopsisLake using merged histograms, focusing on query accuracy, latency, and the impact of different alignment strategies. In Section 5.4, we compare query-driven and data-driven alignment solutions under varying parameters, analyzing their effects on synopsis accuracy and query results.

5.2 Performance of Data Ingestion

Scalability. SynopsisLake stores raw data in binary format and synopses in Parquet, both on HDFS. We test scalability by loading 1,089, 2,182, 3,253, and 4,385 datasets (1.6B–6.6B records). Each setting is tested with and without synopsis generation using 128^2 2D count histograms. We generate 2D count histograms with 128^2

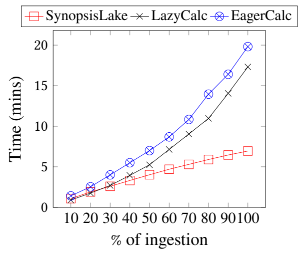
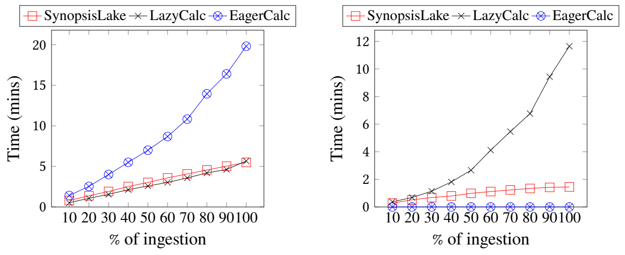


Figure 5: Total time (mins): processing the mixed workload.



(a) Total ingestion time (mins) (b) Total query response time (mins)

Figure 6: Total time breakdown: ingestion vs. query.

buckets as the data synopses. Table 2 reports the total ingestion times and the overhead from synopsis creation. SynopsisLake only 3.4%–8.4% overhead, demonstrating good scalability with minimal synopsis creation cost.

Overhead to compute and write synopsis files. As described in Section 2, each Parquet file contains shared metadata columns and type-specific content columns. Table 3 shows the overhead of creating 50 synopsis files under different space budgets.

Uniform samples have the lowest overhead due to the efficient random sampling process. Count histograms only store bucket counts, resulting in lower overhead compared to other histogram types. Wavelets add transformation overhead on top of histograms. Geometric histograms operate on polygon datasets, where each polygon may overlap with multiple cells. They collect four types of cell-level statistics, increasing processing time. Spatial sketches have the highest overhead: (1) They maintain multi-level counters (point and interval covers), which increases computation during ingestion; (2) They also process polygon datasets, which require expensive geometric processing.

Overall performance on mixed workloads. SynopsisLake slightly increases ingestion time but greatly improves query efficiency. Figure 5 and Figure 6 compare SynopsisLake with LazyCalc and EagerCalc on a mixed workload consisting of batched insertions followed by query rounds.

We create 2D count histograms as the data synopses. We ingest 20M records in 10 files with 20,000 queries per round. LazyCalc ingests raw data and constructs synopses at query time. EagerCalc recomputes synopses for the entire data lake after each insertion. SynopsisLake builds partial synopses per dataset during ingestion and merges them at query time. All systems store the synopses in memory during query execution. After new datasets are inserted, the three systems will recompute the new synopses for querying.

Figure 5 shows the total time of three systems to process the mixed workload. SynopsisLake is 3x faster than LazyCalc and EagerCalc systems. Figure 6(a) shows the total ingestion times. Compared with LazyCalc, the overhead of SynopsisLake to create partial synopses for each dataset is very small. Since EagerCalc creates the synopsis for all the datasets in the data lake, its ingestion performance is the worst. Figure 6(b) reports the total querying processing time of the three systems. The synopses combination process of SynopsisLake is lightweight. Therefore, the query performance of SynopsisLake is close to EagerCalc. LazyCalc accesses all the datasets to create the synopsis during the query time and its query performance is the worst among the three systems.

Table 4: Latency (ms): spatial join selectivity estimation.

Resolution	64^2	128^2	256^2
Query Latency	0.322	1.281	6.527

Summary. SynopsisLake achieves strong scalability and outperforms baseline systems on mixed workloads. It incurs less than 10% ingestion overhead while delivering up to 3× speedup over LazyCalc and EagerCalc.

5.3 Performance of Query Processing

Precision. We evaluate the accuracy of SynopsisLake using merged synopses and report the average relative error (ARE) against exact results. We compare with QueryOpt, which performs partial search over unmerged synopses built from original data. Note that QueryOpt only applies to range scan estimation and not to tasks like spatial join selectivity. To study the impact of merging, we vary the *merging rate*, defined as the ratio of merged resolution to the maximum resolution (the canonical ranges). A 100% merging rate retains full detail (same as QueryOpt), while lower rates reduce space via bucket compaction (e.g., 5% uses only 5% of the original buckets). For all data-driven alignment results, we set the weighting factor $\alpha = 0.5$, $\varepsilon = 10^{-6}$, and $k = 0.5$.

We also compare different alignment strategies for merging: (1) Uniform (equal-sized grid cells), (2) Random (random-sized cells, averaged over 10 runs), and (3) vMeasure (based on V-Optimal histograms [2], applied to canonical ranges). Implementation details for baselines are in Appendix 8.3.

Figure 7 shows the ARE over 32,000 uniformly distributed range queries on the OSM21/pois dataset. The y-axis shows ARE, where shorter bars indicate higher accuracy. Note that the y-axis does not start at zero or extend to the maximum error—this intentional scaling highlights the gap between SynopsisLake (or baseline methods) and QueryOpt, making differences more visually discernible. The x-axis shows 12 merging rates, each represented as a group of bars. In each group, the first bar corresponds to the QueryOpt solution (unmerged synopses), while the red bar shows SynopsisLake with data-driven alignment. Across all merging rates, SynopsisLake consistently achieves lower error than the three baseline strategies: Uniform, Random, and vMeasure.

We also evaluate SynopsisLake's query-driven alignment by comparing it to QueryOpt and the three baselines. Based on prior access patterns, we extract a hot region and 13,000 range queries confined to this area. The alignment is computed using 300 training queries ($\approx 2\%$ of the test set). Figure 8 shows the ARE results. Black bars represent SynopsisLake with query-driven alignment, which consistently achieves lower error than all three baselines.

Efficiency. Table 4 presents the latency of spatial join selectivity estimation using merged Geometric Histograms at resolutions 64^2,

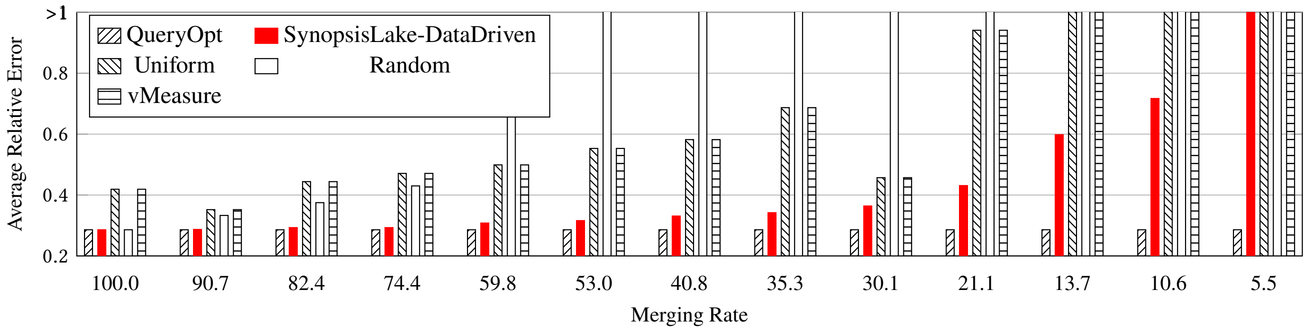


Figure 7: Range selectivity estimation: Compare QueryOpt with data-driven alignment and baseline alignment solutions.

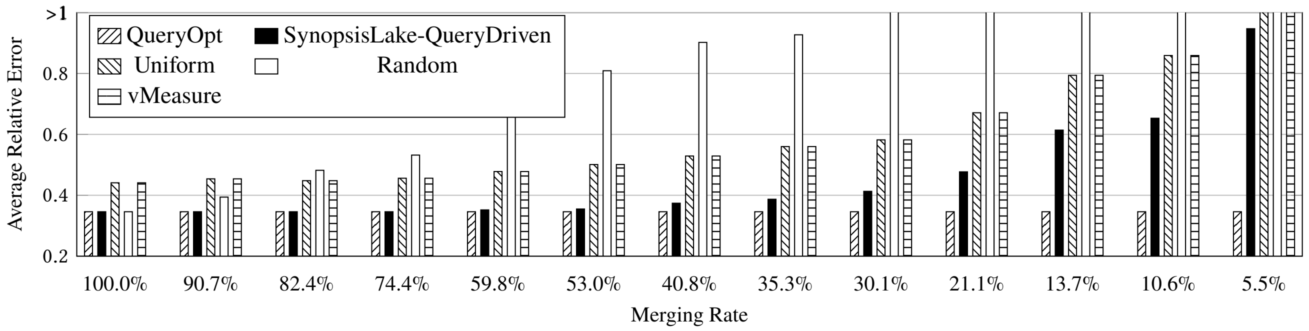


Figure 8: Range selectivity estimation: Compare QueryOpt with query-driven alignment and baseline alignment solutions..

128^2, and 256^2. Each experiment computes five merged histograms generated via data-driven alignment. On average, each query joins 25 histogram pairs. Latency increases with resolution due to the cost of evaluating all cross-bucket pairs. For instance, 128^2 resolution with five histograms requires aggregating $128^2 \times 5$ bucket pairs. Higher space budgets result in significantly longer query times.

Summary. SynopsisLake delivers high query accuracy with much lower storage cost than QueryOpt. Both data-driven and query-driven alignment strategies consistently outperform baseline methods in accuracy and maintain reasonable query latency.

5.4 Comparing Different Alignment Strategies

We have demonstrated that both the query-driven and data-driven alignment strategies enable SynopsisLake to produce merged histograms with higher query accuracy than baseline methods. Section 4 introduced the data-driven strategy, which optimizes a weighted objective function balancing statistical accuracy and density preservation, controlled by the parameter α.

Here, we analyze how varying α affects performance and compare results against the query-driven strategy and two baselines: Uniform alignment and QueryOpt. The following data-driven variants are evaluated: (1) alignOnly (red bars with crosshatch): data-driven solution optimizing only statistical accuracy; (2) densityOnly (black bars with horizontal lines): data-driven solution optimizing only density similarity; (3) Data-driven (combine25, combine50, combine75): data-driven solution with α values of 0.25, 0.5, and 0.75; (4) queryDriven (black bars): query-driven solution; (5) Uniform; (6) QueryOpt.

All methods are evaluated on the same query workload. Figure 9(a) shows the merged histograms accuracy, while Figure 9(b) reports query accuracy. The data-driven alignOnly strategy achieves the most accurate histograms due to its explicit optimization of the statistical accuracy of merging. However, it performs poorly on queries, as the merged buckets layout does not align well with query ranges. Among the data-driven strategies, $\alpha = 0.25$ (combine25) and $\alpha = 0.50$ (combine50) achieve a strong balance, delivering competitive query accuracy while maintaining reasonable histogram accuracy. Overall, the data-driven solutions generally outperform Uniform alignment, confirming the benefits of balancing histogram shape with data distribution. The query-driven approach achieves the best query accuracy (second only to QueryOpt), demonstrating the effectiveness of leveraging workload information.

Summary. Optimizing for histogram accuracy reduces merge error but may hurt query accuracy. A balanced data-driven strategy offers better trade-offs, while query-driven alignment yields the best results when workload knowledge is available.

6 RELATED WORK

This section highlights the contributions of SynopsisLake from three perspectives: (1) the quality metric for partition-based synopses and queries, (2) the synopses management, (3) the contribution of SynopsisLake from a system perspective.

6.1 AQP by using data synopses

Data synopses. A data synopsis extracts statistical information from large data and compactly stores it. Overall, there are four categories: histograms [6, 8, 18, 59, 63] and wavelets [20, 52, 67, 68, 71],samples [4, 70, 78], and sketches [26, 50, 57, 65]. Data synopsese are widely used to range query selectivity estimation [1], join query selectivity estimation [8], detect the heavy hitters [3], compute quantiles [50], and provide approximate query answers for aggregate queries [4] and even more complex queries (e.g., join, clustering, etc.) [20, 60]. There are also non-synopses-based AQP methods, such as Bayesian networks [66] and learning models [39].

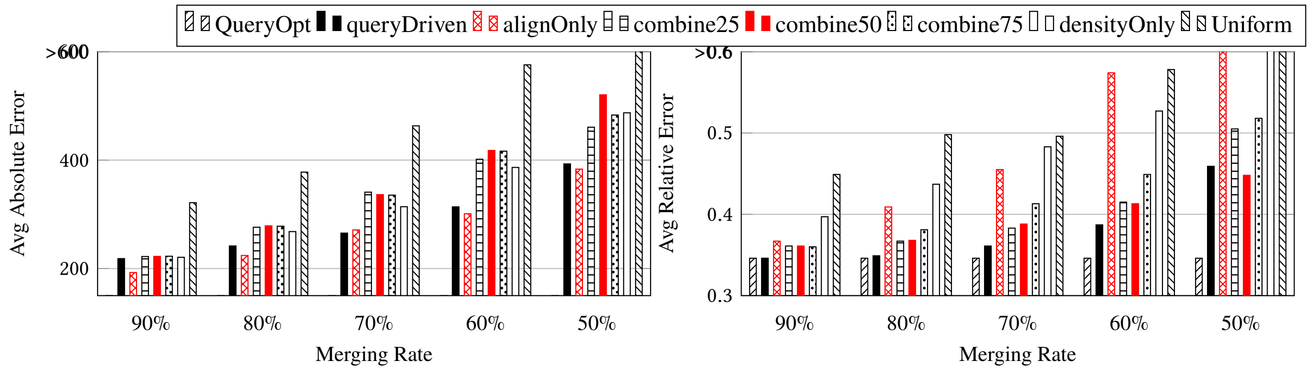


Figure 9: Histograms combination accuracy (left) and query accuracy (right): Query-Driven vs. Read-Opt and Baselines.

SynopsisLake manages and processes the data synopses stored as metadata in Lakehouse. The learning and purely statistical solutions are not considered in our scenario.

Statistical-based quality control. Researchers [16, 21, 37, 40, 47, 54] consider statistical methods to evaluate the quality of range cardinality estimation, join selectivity estimation, and approximate query processing. Most works [16, 21, 40, 47] use conditional probability to define the selectivity of the query over the whole dataset under the selectivity of the samples. The statistical models are mostly in the form of binomial distribution and beta distribution. Ripple join algorithm [37] adaptively adjusts its behavior by using the statistical properties of the input data. The goal is to minimize the time until an acceptably precise estimation which is measured by the length of a confidence interval. They compute the confidence interval following the central limit theorem(CLT). INCVISAGE [54] applies Hoeffding's inequality [42] to compute the sample size. The approximate aggregation is computed based on samples. In these sampling-based methods, the quality of the result is computed based on the confidence threshold of the statistical model. However, there is no solution to evaluate the quality of approximate answers returned by histograms. In SynopsisLake, we fill this gap by defining a statistical model for histogram-based AQP.

Synopses mergeability and management. In Lakehouse, synopses management is related to managing the overlapping synopses and using them to finish AQP tasks efficiently. These goals can be achieved by merging the overlapping synopses. Agarwal et al. [3] define that mergeability requires the target merged data synopsis to preserve the error and size guarantees of the source data synopses. They claim that all sketches sample synopses are mergeable. The partition-based data synopsis families contain equi-width histograms, equi-height histograms, V-optimal histograms, and wavelets. Existing works [1, 53] discussed why equi-height histograms, V-optimal histograms, and wavelets are not mergeable.

The reason is that overlapping partition-based data synopses have varying bucket boundaries. SynopsisLake merges the overlapping data synopses based on a three-step framework that consists of alignment, reshaping, and merging. Researchers propose query-driven partitioning strategies [24], greedy solution [27], and V-optimal solutions [2, 44, 45] to partitions the data space or compute partition-based synopses. These solutions work for single-type synopsis and are not general to manage different types of data synopses in Lakehouse. Additionally, they [2, 27, 44, 45] returned non-uniform histograms which are inefficient for AQP. SynopsisLake partitions the data space following the quality metrics (data-driven and query-driven) for histograms and provides the approximate answers with good quality.

6.2 Data lakes and Lakehouse

The concept of the data lake was first proposed in 2010 by James Dixon [29]. It receives significant attention from both academia and industry. Researchers are still working on proposing new architectures and definitions for data lake [22, 38, 46, 58]. Researchers in the industry [7, 14, 43, 55] develop their own data lakes. Since data lakes store heterogeneous resources, finding useful information among the large amount the datasets is very important. Lots of works focus on query-driven dataset discovery and dataset generation [19, 30, 48, 62, 64, 76, 79]. Researchers are also working on enhancing the capabilities of data lakes by incorporating features traditionally associated with data warehouses. They focus on the metadata management in data lakes [36, 58] and designing data Lakehouse [9, 10, 14, 17, 43]. Delta Lake is one of the popular open-source data Lakehouse. It compacts data logs of data lakes into Apache Parquet format [12] to provide ACID properties for data lakes. The Parquet files are stored in Lakehouse. With this design, data scientists can use SQL queries to process the data stored in data lakes. In this work, we propose SynopsisLake by extending the architecture of Delta Lake. We compact the data synopses into Parquet files to support AQP for data lakes.

7 CONCLUSION AND FUTURE WORK

We present SynopsisLake, a new Lakehouse system that compacts spatial data synopses and leverages them for approximate query processing (AQP) in data lakes. SynopsisLake introduces a unified framework for combining diverse spatial synopses and defines a quality metric to guide the merging process. Experiments show that SynopsisLake can efficiently merge hundreds of synopses and deliver high query accuracy. Several improvements will be explored in future work. First, we plan to design more effective methods for combining non-count-based histograms to better support accurate query cost estimation. Second, we will extend our framework to define quality-aware combination strategies for spatial sketches and samples, enabling robust support for more complex AQP workloads.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation (NSF) under grants IIS-1954644 and IIS-2046236.

REFERENCES

[1] Ildar Absalyamov, Michael J Carey, and Vassilis J Tsotras. 2018. Lightweight cardinality estimation in LSM-based systems. In *Proceedings of the 2018 International Conference on Management of Data*. 841–855.

[2] Jayadev Acharya, Ilias Diakonikolas, Chinmay Hegde, Jerry Zheng Li, and Ludwig Schmidt. 2015. Fast and near-optimal algorithms for approximating distributions by histograms. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 249–263.

[3] Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. 2013. Mergeable summaries. *ACM Transactions on Database Systems (TODS)* 38, 4 (2013), 1–28.

[4] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*. 29–42.

[5] Sattam Alsubaiee et al. 2014. AsterixDB: a scalable, open source BDMS. *Proceedings of the VLDB Endowment* 7, 14 (2014), 1905–1916.

[6] Ahmed M Aly, Ahmed R Mahmood, Mohamed S Hassan, Walid G Aref, Mourad Ouzzani, Hazem Elmeleegy, and Thamir Qadah. 2015. Aqwa: adaptive query workload aware partitioning of big spatial data. *Proceedings of the VLDB Endowment* 8, 13 (2015), 2062–2073.

[7] Amazon. [n. d.]. Amazon AWS Data Lake. https://aws.amazon.com/big-data/datalakes-and-analytics/

[8] Ning An, Zhen-Yu Yang, and Anand Sivasubramaniam. 2001. Selectivity estimation for spatial joins. In *Proceedings 17th International Conference on Data Engineering*. IEEE, 368–375.

[9] Apache Hudi. 2021. Apache Hudi. https://hudi.apache.org

[10] Apache Iceberg. 2017. Apache Iceberg. https://iceberg.apache.org

[11] Apache ORC. 2013. Apache ORC. https://orc.apache.org

[12] Apache Parquet. 2013. Apache Parquet. https://parquet.apache.org

[13] Apache Puffin. 2022. Apache Puffin. https://iceberg.apache.org/puffin-spec/#blobmetadata

[14] Michael Armbrust, Tathagata Das, Liwen Sun, Burak Yavuz, Shixiong Zhu, Mukul Murthy, Joseph Torres, Herman van Hovell, Adrian Ionescu, Alicja Łuszczak, et al. 2020. Delta lake: high-performance ACID table storage over cloud object stores. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3411–3424.

[15] Michael Armbrust, Ali Ghodsi, Reynold Xin, and Matei Zaharia. 2021. Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In *Proceedings of CIDR*, Vol. 8.

[16] Brian Babcock and Surajit Chaudhuri. 2005. Towards a robust query optimizer: a principled and practical approach. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 119–130.

[17] Alexander Behm, Shoumik Palkar, Utkarsh Agarwal, Timothy Armstrong, David Cashman, Ankur Dave, Todd Greenstein, Shant Hovsepian, Ryan Johnson, Arvind Sai Krishnan, et al. 2022. Photon: A fast query engine for lakehouse systems. In *Proceedings of the 2022 International Conference on Management of Data*. 2326–2339.

[18] Richard Beigel and Egemen Tanin. 1998. The geometry of browsing. In *Latin American Symposium on Theoretical Informatics*. Springer, 331–340.

[19] Alex Bogatu, Alvaro AA Fernandes, Norman W Paton, and Nikolaos Konstantinou. 2020. Dataset discovery in data lakes. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 709–720.

[20] Kaushik Chakrabarti, Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. 2001. Approximate query processing using wavelets. *The VLDB Journal* 10, 2 (2001), 199–223.

[21] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. 2007. Optimized stratified sampling for approximate query processing. *ACM Transactions on Database Systems (TODS)* 32, 2 (2007), 9–es.

[22] Mohamed Cherradi and Anass EL Haddadi. 2022. Data Lakes: A Survey Paper. In *Innovations in Smart Cities Applications Volume 5: The Proceedings of the 6th International Conference on Smart City Applications*. Springer, 823–835.

[23] Graham Cormode, Minos Garofalakis, Peter J Haas, Chris Jermaine, et al. 2011. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends® in Databases* 4, 1–3 (2011), 1–294.

[24] Graham Cormode, Minos Garofalakis, and Michael Shekelyan. 2021. Data-Independent Space Partitionings for Summaries. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 285–298.

[25] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* 55, 1 (2005), 58–75.

[26] Abhinandan Das, Johannes Gehrke, and Mirek Riedewald. 2004. Approximation techniques for spatial data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. 695–706.

[27] Ilias Diakonikolas, Jerry Li, and Ludwig Schmidt. 2018. Fast and sample near-optimal algorithms for learning multidimensional histograms. In *Conference On Learning Theory*. PMLR, 819–842.

[28] Claudia Diamantini, Domenico Potena, and Emanuele Storti. 2021. A semantic data lake model for analytic query-driven discovery. In *The 23rd International Conference on Information Integration and Web Intelligence*. 183–186.

[29] James Dixon. 2010. Pentaho, Hadoop, and Data Lakes. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes

[30] Yuyang Dong, Kunihiro Takeoka, Chuan Xiao, and Masafumi Oyamada. 2021. Efficient joinable table discovery in data lakes: A high-dimensional similarity-based approach. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 456–467.

[31] Ahmed Eldawy and Mohamed F. Mokbel. 2019. All points on the map as extracted from OpenStreetMap. doi:10.6086/N100004J

[32] Ahmed Eldawy and Mohamed F. Mokbel. 2019. All water areas in the world from OpenStreetMap. This includes coastal lines, lakes, rivers, pools, and others. doi:10.6086/N1668B70

[33] Ahmed Eldawy and Mohamed F. Mokbel. 2019. Boundaries of parks and green areas from all over the world as extracted from OpenStreetMap. doi:10.6086/N1RX994T

[34] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete mathematics & theoretical computer science* Proceedings (2007).

[35] Edward Gan, Peter Bailis, and Moses Charikar. 2020. Coopstore: Optimizing precomputed summaries for aggregation. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2174–2187.

[36] Yihan Gao, Silu Huang, and Aditya Parameswaran. 2018. Navigating the data lake with datamaran: Automatically extracting structure from log datasets. In *Proceedings of the 2018 International Conference on Management of Data*. 943–958.

[37] Peter J Haas and Joseph M Hellerstein. 1999. Ripple joins for online aggregation. *ACM SIGMOD Record* 28, 2 (1999), 287–298.

[38] Rihan Hai, Christos Koutras, Christoph Quix, and Matthias Jarke. 2023. Data Lakes: A Survey of Functions and Systems. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[39] Shohedul Hasan, Saravanan Thirumuruganathan, Jees Augustine, Nick Koudas, and Gautam Das. 2020. Deep learning models for selectivity estimation of multi-attribute queries. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1035–1050.

[40] Axel Hertzschuch, Guido Moerkotte, Wolfgang Lehner, Norman May, Florian Wolf, and Lars Fricke. 2021. Small selectivities matter: Lifting the burden of empty samples. In *Proceedings of the 2021 International Conference on Management of Data*. 697–709.

[41] Stefan Heule, Marc Nunkesser, and Alexander Hall. 2013. Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology*. 683–692.

[42] Wassily Hoeffding. 1994. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding* (1994), 409–426.

[43] IBM. [n. d.]. IBM Data Lake and Lakehouse. https://www.ibm.com/data-lake

[44] Yannis E Ioannidis and Viswanath Poosala. 1995. Balancing histogram optimality and practicality for query result size estimation. *Acm Sigmod Record* 24, 2 (1995), 233–244.

[45] Hosagrahar Visvesvaraya Jagadish, Nick Koudas, S Muthukrishnan, Viswanath Poosala, Kenneth C Sevcik, and Torsten Suel. 1998. Optimal histograms with quality guarantees. In *VLDB*, Vol. 98. 24–27.

[46] Pwint Phyu Khine and Zhao Shun Wang. 2018. Data lake: a new ideology in big data era. In *ITM web of conferences*, Vol. 17. EDP Sciences, 03025.

[47] Per-Ake Larson, Wolfgang Lehner, Jingren Zhou, and Peter Zabback. 2007. Cardinality estimation using sample views with quality assurance. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 175–186.

[48] Aristotelis Leventidis, Laura Di Rocco, Wolfgang Gatterbauer, Renée J Miller, and Mirek Riedewald. 2021. DomainNet: Homograph Detection for Data Lake Disambiguation. *arXiv preprint arXiv:2103.09940* (2021).

[49] Xi Liang, Stavros Sintos, and Sanjay Krishnan. 2023. JanusAQP: Efficient partition tree maintenance for dynamic approximate query processing. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 572–584.

[50] Ge Luo, Lu Wang, Ke Yi, and Graham Cormode. 2016. Quantiles over data streams: experimental comparisons, new analyses, and further improvements. *The VLDB Journal* 25, 4 (2016), 449–472.

[51] Charles Masson, Jee E Rim, and Homin K Lee. 2019. DDSketch: a fast and fully-mergeable quantile sketch with relative-error guarantees. *Proceedings of the VLDB Endowment* 12, 12 (2019), 2195–2205.

[52] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. 1998. Wavelet-based histograms for selectivity estimation. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*. 448–459.

[53] Rudi Poepsel-Lemaitre, Martin Kiefer, Joscha Von Hein, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. 2021. In the land of data streams where synopses are missing, one framework to bring them all. *Proceedings of the VLDB Endowment* 14, 10 (2021), 1818–1831.

[54] Sajjadur Rahman, Maryam Aliakbarpour, Ha Kyung Kong, Eric Blais, Karrie Karahalios, Aditya Parameswaran, and Ronitt Rubinfield. 2017. I've seen" enough" incrementally improving visualizations to support rapid decision making. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1262–1273.

[55] Raghu Ramakrishnan, Baskar Sridharan, John R Douceur, Pavan Kasturi, Balaji Krishnamachari-Sampath, Karthick Krishnamoorthy, Peng Li, Mitica Manu, Spiro Michaylov, Rogério Ramos, et al. 2017. Azure data lake store: a hyperscale distributed file service for big data analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 51–63.

[56] El Kindi Rezig, Anshul Bhandari, Anna Fariha, Benjamin Price, Allan Vanterpool, Vijay Gadepally, and Michael Stonebraker. 2021. DICE: data discovery by example. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2819–2822.

[57] Florin Rusu and Alin Dobra. 2008. Sketches for size of join estimation. *ACM Transactions on Database Systems (TODS)* 33, 3 (2008), 1–46.

[58] Pegdwendé Sawadogo and Jérôme Darmont. 2021. On data lake architectures and metadata management. *Journal of Intelligent Information Systems* 56 (2021), 97–120.

[59] AB Siddique, Ahmed Eldawy, and Vagelis Hristidis. 2019. Euler++: Improved Selectivity Estimation for Rectangular Spatial Records. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 4129–4133.

[60] Abu Bakar Siddique, Ahmed Eldawy, and Vagelis Hristidis. 2019. Comparing synopsis techniques for approximate spatial data analysis. *Proceedings of the VLDB Endowment* 12, 11 (2019).

[61] Samriddhi Singla and Ahmed Eldawy. 2020. Flexible computation of multidimensional histograms. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Spatial Gems (SpatialGems 2020)(Seattle, Washington, USA). ACM*.

[62] Jie Song and Yeye He. 2021. Auto-Validate: Unsupervised Data Validation Using Data-Domain Patterns Inferred from Data Lakes. In *Proceedings of the 2021 International Conference on Management of Data*. 1678–1691.

[63] Chengyu Sun, Divyakant Agrawal, and Amr El Abbadi. 2002. Selectivity estimation for spatial joins with geometric selections. In *International Conference on Extending Database Technology*. Springer, 609–626.

[64] Wenbo Tao, Adam Sah, Leilani Battle, Remco Chang, and Michael Stonebraker. 2021. Kyrix-J: Visual Discovery of Connected Datasets in a Data Lake. (2021).

[65] Yufei Tao, George Kollios, Jeffrey Considine, Feifei Li, and Dimitris Papadias. 2004. Spatio-temporal aggregation using sketches. In *Proceedings. 20th International Conference on Data Engineering*. IEEE, 214–225.

[66] Kostas Tzoumas, Amol Deshpande, and Christian S Jensen. 2013. Efficiently adapting graphical models for selectivity estimation. *The VLDB Journal* 22 (2013), 3–27.

[67] Jeffrey Scott Vitter and Min Wang. 1999. Approximate computation of multidimensional aggregates of sparse data using wavelets. *Acm Sigmod Record* 28, 2 (1999), 193–204.

[68] Jeffrey Scott Vitter, Min Wang, and Bala Iyer. 1998. Data cube approximation and histograms via wavelets. In *Proceedings of the seventh international conference on Information and knowledge management*. 96–104.

[69] Tin Vu, Ahmed Eldawy, Vagelis Hristidis, and Vassilis Tsotras. 2021. Incremental partitioning for efficient spatial data analytics. *Proceedings of the VLDB Endowment* 15, 3 (2021), 713–726.

[70] Jin-Feng Wang, A Stein, Bin-Bo Gao, and Yong Ge. 2012. A review of spatial sampling. *Spatial Statistics* 2 (2012), 1–14.

[71] Min Wang, Jeffrey Scott Vitter, Lipyeow Lim, and Sriram Padmanabhan. 2001. Wavelet-based cost estimation for spatial queries. In *International Symposium on Spatial and Temporal Databases*. Springer, 175–193.

[72] Wikipedia contributors. 2024. Beta distribution, Practical Implementations: Alpha and Beta Calculations Summary– Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Beta_distribution&oldid=1210316554. [Online; accessed 28-February-2024].

[73] Wikipedia contributors. 2025. Skewness — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Skewness&oldid=1272988155 [Online; accessed 24-February-2025].

[74] Qin Yuan, Ye Yuan, Zhenyu Wen, He Wang, Chen Chen, and Guoren Wang. 2022. Exploring Heterogeneous Data Lake based on Unified Canonical Graphs. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1834–1838.

[75] Yaming Zhang and Ahmed Eldawy. [n. d.]. OpenStreetMap Points of Interest.

[76] Yi Zhang and Zachary G Ives. 2020. Finding related tables in data lakes for interactive data science. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1951–1966.

[77] Fuheng Zhao, Sujaya Maiyya, Ryan Wiener, Divyakant Agrawal, and Amr El Abbadi. 2021. Kll±approximate quantile sketches over dynamic datasets. *Proceedings of the VLDB Endowment* 14, 7 (2021), 1215–1227.

[78] Zhuoyue Zhao, Feifei Li, and Yuxi Liu. 2020. Efficient join synopsis maintenance for data warehouse. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2027–2042.

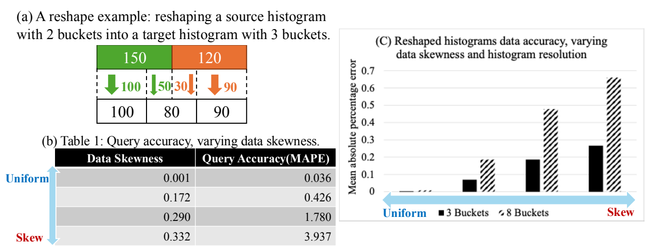
[79] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J Miller. 2019. Josie: Overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the 2019 International Conference on Management of Data*. 847–864.

8 APPENDIX

8.1 Additional Discussion for Section 3

8.1.1 Example: Histogram Estimation Under Uniform Density Assumption. Consider a range query Q that overlaps with 10% of a histogram bucket. The estimated query answer in this case is $0.1 \times v_B$, where v_B is the bucket's total count. This estimation assumes that 10% of the data points are uniformly distributed within the first 10% of the bucket's range, regardless of where Q begins.

Such an assumption is valid only if data points are uniformly distributed within the bucket. However, this uniformity rarely holds in real-world datasets, potentially introducing estimation error.



(b) Table 1: Query accuracy, varying data skewness.

Data Skewness	Query Accuracy(MAPE)
0.001	0.036
0.172	0.426
0.290	1.780
0.332	3.937

Figure 10: Impact of data skewness and overlap ratio on histogram estimation error.

8.1.2 Illustration of Skewness and Overlap Effects. Figure 10 demonstrates how two key factors—**data skewness** and **overlap ratio**—affect histogram-based query estimation:

(1) **Data Skewness**: Histograms are more accurate when summarizing uniform data. As skewness increases, estimation error rises. Note that our skewness metric (DK_H) differs from traditional statistical skewness [73]; it captures density variability across buckets using median absolute deviation.
(2) **Overlap Ratio**: Estimation improves with larger overlap between the query range and the bucket. Even under extreme skew, full overlap yields an exact answer.

Example 3.2 and Figure 4 illustrate how to compute DK_H using bucket densities and how it impacts accuracy.

8.1.3 Justification for Using the Beta Distribution. Our model estimates histogram query accuracy by modeling the random behavior of the overlap ratio, $\frac{len_Q}{len_B}$, as a probability distribution. The **beta distribution** is a natural fit because:

- It models continuous proportions over the interval $[0, 1]$, matching the range of overlap ratios [72].
- Its shape is flexible and can express uncertainty by adjusting two parameters (α, β). We set $\alpha = \beta$ for symmetry, simplifying the model while capturing skewness effects.

The beta distribution's probability density function (PDF) is:

$$\mathrm{PDF}_H = f(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}, \tag{11}$$

where $x \in [0, 1]$ and $B(\alpha, \beta)$ is the beta function.

As skewness increases, we lower α and flatten the distribution. For uniform data, a higher α yields a sharper peak, indicating higher confidence in query accuracy.

8.1.4 Mapping Function 3 Properties. The mapping function Equation 3 has following properties:

- As $\text{DK}_H \to \infty$, the underlying data is very skewed so that α (and β) $\to 1$.
- As $\text{DK}_H \to 0$, the underlying data is uniformly distributed so that α (and β) $\to \frac{k}{\epsilon} + 1$, which is the maximum value of α and β.

This relationship allows the model to adapt its confidence based on the skewness of the underlying data, linking statistical properties of the histogram to expected estimation quality.

8.1.5 Derivation of Error Ratio Metric. This subsection provides the step-by-step derivation of the estimated error ratio defined in Equation 6, which quantifies the uncertainty of a histogram bucket B_i when answering a range query Q.

The error ratio $r_{B_i \to Q}$ is the probability mass outside the aligned overlap interval $[a, b]$ under the beta distribution $\text{PDF}_H = f(x; \alpha, \beta)$:

$$r_{B_i \to Q} = 1 - \frac{P(a < X < b)}{P(0 \leq X \leq 1)} = 1 - \frac{\int_a^b f(x;\alpha,\beta)\,dx}{\int_0^1 f(x;\alpha,\beta)\,dx}$$

$$= 1 - \int_a^b f(x;\alpha,\beta)\,dx = 1 - I_b(\alpha,\beta) + I_a(\alpha,\beta) \qquad (12)$$

where $I_x(\alpha, \beta)$ is the cumulative distribution function of the beta distribution, and $[a, b]$ is the overlap-centered interval defined in Equation 7:

$$a = 0.5 - \frac{1}{2} \cdot \frac{len_{OQ}}{len_{B_i}}, \quad b = 0.5 + \frac{1}{2} \cdot \frac{len_{OQ}}{len_{B_i}}.$$

This formulation captures how the overlap ratio contributes to uncertainty, with higher density concentration within $[a, b]$ corresponding to more accurate query estimates.

8.1.6 The Quality of Multi-dimensional Approximate Query Answer. Equation 6/12 can be naturally extended to estimate the quality of approximate answers returned by multi-dimensional buckets. The only difference lies in how the aligned overlap interval $[a, b]$ is computed. In the one-dimensional case, the value is given by $\frac{len_{OQ}}{len_{B_i}}$ as shown above. In the multi-dimensional setting, the overall overlap ratio is obtained as the product of the overlap ratios along each dimension. For example, in two dimensions the overlap ratio corresponds to the 'area', while in three dimensions it corresponds to the 'volume'. The aligned overlap interval $[a, b]$ is then computed as follows:

$$a = 0.5 - \tfrac{1}{2} \cdot \prod_{dim=1}^{D} \frac{len_{OQ_{dim}}}{len_{B_{i_{dim}}}}, \quad b = 0.5 + \tfrac{1}{2} \cdot \prod_{dim=1}^{D} \frac{len_{OQ_{dim}}}{len_{B_{i_{dim}}}}$$

8.2 Additional Discussion for Section 4

8.2.1 Priority Queue with Updatable Priorities. To support efficient priority updates during merging, we implement a custom priority queue based on a binary min-heap. Standard priority queues do not support in-place updates, which are necessary for adjusting merge scores after each operation.

Our implementation combines an array-based heap with a hash map that tracks each element's index in the heap. When a merge score changes, the hash map allows constant-time lookup of its position, followed by a heapify operation to restore heap order. If the score increases, the element is pushed down; if it decreases, it is bubbled up.

This design ensures $O(\log n)$ time for both extraction and update, allowing fast re-scoring of neighboring merge positions throughout the algorithm.

8.2.2 Merging Sketches and Samples. We extend the Align-Reshape-Merge framework to support Spatial Sketches and sample-based synopses.

SynopsisLake employs Spatial Sketches [26] for approximate spatial join selectivity estimation. Each sketch maintains multi-level counters over dyadic intervals and their endpoints, hierarchically partitioning the data space. To merge sketches defined over different ranges, we align them to a common set of dyadic intervals. Counter values are reshaped using a weighted scheme based on overlap ratios between source and merged intervals. While effective, this method currently lacks a formal quality metric to assess merge accuracy.

To merge samples, SynopsisLake supports both uniform and stratified samples. We first unify all input samples, then resample under a global space budget. The unified range is partitioned using either uniform splitting or k-means clustering. Within each partition, we perform uniform sampling without replacement. Sample counts are proportionally allocated as $n_h = n \cdot N_h / N$, where n is the total sample budget, N_h the size of partition h, and N the total input size.

8.2.3 Multi-dimensional Alignment. Definition 4.2 and Definition 10 and easily be extended to multiple settings. For data-driven reshaping, we should replace the alignment error 'x' by multi-dimensional equations shown in Appendix 8.1.6. For query-driven reshaping, the query answers should be computed by multi-dimensional buckets.

8.3 Additional Discussion for Section 5

We provide details on the baseline alignment methods:

(1) The **Uniform alignment** splits the entire data range into equal-sized grid cells.

(2) The **Random solution** generates grid cells with random sizes. We run this method for 10 rounds and report the average results.

(3) The **vMeasure solution** extends the V-Optimal histogram method [2], which groups data items with similar frequencies. The original algorithm takes a sorted tuple list (i, y_i), where i is a data value and y_i its frequency. It partitions adjacent values into buckets, assigning each bucket a mean value $\bar{v}_j$, and minimizes the L_2 error $\sum(y_i - \bar{v}_j)^2$ using a greedy merging strategy. To apply this in our setting, we treat each canonical range as a bucket and use its value as the frequency, forming an item frequency vector. We then apply the original greedy algorithm to compute merged partitions.