PilotDB: A Step Toward Minimalist Approximate Query Processing with A Priori Error Guarantees

Yuxuan Zhu*
University of Illinois (UIUC)
Champaign-Urbana, USA
yxx404@illinois.edu

Tengjun Jin*
University of Illinois (UIUC)
Champaign-Urbana, USA
tengjun2@illinois.edu

Stefanos Baziotis
University of Illinois (UIUC)
Champaign-Urbana, USA
sb54@illinois.edu

Chengsong Zhang
University of Illinois (UIUC)
Champaign-Urbana, USA
cz81@illinois.edu

Charith Mendis
University of Illinois (UIUC)
Champaign-Urbana, USA
charithm@illinois.edu

Daniel Kang
University of Illinois (UIUC)
Champaign-Urbana, USA
ddkang@illinois.edu

ABSTRACT

Although approximate query processing (AQP) has been studied for decades, its adoption remains limited. Existing methods fail to balance the need for minimal maintenance or modifications on database management systems (DBMSs) and a priori error guarantees, discouraging practitioners from adopting AQP. In this work, we propose PilotDB, an online AQP method that eliminates maintenance and modifications while providing a priori error guarantees efficiently. PilotDB leverages two novel techniques to achieve the goal. First, to reduce sampling costs without modifying the underlying DBMS, we develop a provable error analysis framework for aggregation queries with page-based sampling. Second, we introduce the two-phase sampling scheme as a novel solution to provide a priori error guarantees without knowledge of the workload. We investigate the statistical properties of page-based sampling and prove the validity of two-phase sampling. We evaluate PilotDB on four workloads, demonstrating 0.92 ~ 126× speedups on transactional and analytical databases with a 5% guaranteed error.

PVLDB Reference Format:
Yuxuan Zhu*, Tengjun Jin*, Stefanos Baziotis, Chengsong Zhang, Charith Mendis, and Daniel Kang. PilotDB: A Step Toward Minimalist Approximate Query Processing with A Priori Error Guarantees. PVLDB, 18(1): XXX-XXX, 2025.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/uiuc-kang-lab/PilotDB/tree/vldb25.

1 INTRODUCTION

Approximate query processing (AQP) is widely studied to reduce the cost of query processing in modern data analytics [1, 5, 9, 10, 18, 21, 36, 41, 45, 48, 51, 52, 58, 64, 66]. Despite its long history in research, AQP is rarely deployed in practice due to the limitations of existing methods. AQP methods based on sketches [10, 23, 51], histograms [53, 55, 58], and wavelets [24, 45] are limited to known queries and numeric workloads. Sampling-based AQP can process general analytical queries, but prior systems fail to achieve three practically significant requirements simultaneously: (1) guaranteed small errors, (2) minimal modifications or maintenance, and (3) significant cost reductions [43, 44].

One of the primary challenges in achieving efficient sampling-based AQP is the high cost of uniform sampling, which can be as slow as a full scan for both row and columnar databases [57]. Existing approaches create and select offline samples for query processing to mitigate the cost [1, 5, 18, 48, 52]. However, this offline sampling approach requires additional efforts from database administrators (DBAs) to maintain samples for statistical correctness and adaptation to data changes. To avoid these complexities, recent work in online AQP suggests implementing samplers with lazy execution, which integrates samplers into the query optimizer [35, 36]. However, this method requires re-implementing various samplers in the DBMS and modifying the query optimizer. Furthermore, without knowledge about the workload, the state-of-the-art online AQP system may lead to errors as high as 100% for data with a skewed distribution [36].

In this work, we propose PilotDB, a step toward satisfying three practical requirements simultaneously. First, PilotDB eliminates the need for special maintenance and modifications to the DBMS through online sampling and query rewriting. Second, PilotDB provides a priori error guarantees through a two-phase sampling scheme. Finally, PilotDB achieves significant cost reductions compared to executing exact queries and reduces the sampling costs through block sampling.

Unlike prior work using offline or database-embedded samplers, PilotDB reduces the overhead of sampling by adopting block sampling, which offers two advantages. First, block sampling is significantly more efficient than uniform sampling since it only accesses the sampled physical pages. Empirically, block sampling can reduce costs by up to 219× compared to uniform sampling in query processing (§7.4). Second, since SQL:2003, `TABLESAMPLE SYSTEM` clause is included as a system-dependent sampling method [38] and widely implemented as block sampling [13, 16, 17, 20, 27, 30, 56]. This allows us to rewrite queries to use block sampling without modifying the underlying DBMS.

*Yuxuan Zhu and Tengjun Jing contributed equally to this work.

Proceedings of the VLDB Endowment, Vol. 18, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

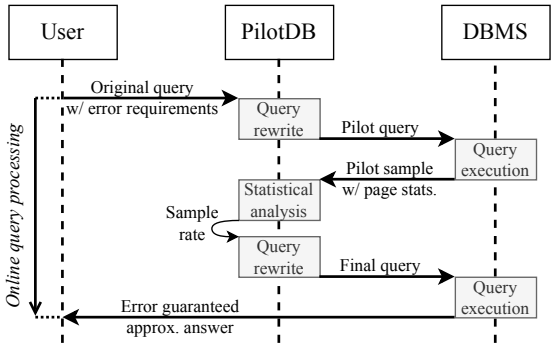


Figure 1: Workflow of PilotDB that avoids modifications to underlying DBMSs and provides a priori error guarantees.

Despite the advantages of block sampling, achieving error guarantees with it for general queries is challenging and not addressed in prior work [11, 29]. The non-uniformity of block sampling introduces two issues that prior work does not overcome: correlated samples and interactions between the sampling and relational operations. The correlated samples produced by block sampling invalidate the confidence intervals calculated by the standard Central Limit Theorem (CLT). Empirically, using CLT for uniform samples to calculate a 95% confidence interval can result in a coverage probability as low as 20% for the following simple query:[1]

```
SELECT SUM(price) FROM orders TABLESAMPLE SYSTEM(1%)
WHERE comment LIKE '%special%'
```

We address this issue by transforming the traditional row-oriented estimators into equivalent page-oriented estimators. Since block sampling results in uniform samples of pages, we can apply CLT to analyze the errors of estimations calculated at the page level.

Second, to process queries with complex relational operations, such as nested queries and `JOIN-GROUP-BY` queries, we need to analyze the interactions between block sampling and common relational operations. We prove that block sampling commutes with projection, selection, join, and bag union. The commutative property enables us to analyze queries with block sampling on the input tables as if the block sampling were applied as the last operation, thereby simplifying the analysis of error guarantees.

A key challenge in guaranteeing errors a priori is determining the sample rate for a requested error without prior knowledge of the data distribution or page layout. To address this, we develop the ***T**wo-**P**hase **S**ampling* scheme (TPS) through dynamic query rewriting. We show the workflow of TPS semantically in Figure 1. In the first phase, we rewrite the original query into a pilot query that estimates the parameters necessary for our error analysis, such as the variance of a column. Next, we calculate the sufficient sample rate to guarantee the requested error. Finally, we rewrite and execute the original query again with the calculated sample rate. We derive the closed-form formula of the sample rate required for a requested error in Section 5. Notably, TPS is completely online, requiring no modifications to the underlying DBMS.

[1]We evaluate the query on DuckDB with the 1,000-scaled TPC-H workload 1,000 times.

We implemented and evaluated PilotDB on four workloads: TPC-H [14], ClickBench [12], Star Schema Benchmark [49], and Instacart [31, 52], which simulate decision-making with big data analytics over online transactions, machine-generated data, and e-commerce. Compared to transactional databases: PostgreSQL [61] and Microsoft SQL Server [42], PilotDB achieves 0.92 $\sim$ 126× speedups. Compared with the analytic database: DuckDB [37], PilotDB achieves 1.0 $\sim$ 13× speedups. Across various settings, PilotDB consistently achieves error guarantees and cost reductions.

We summarize our contribution as follows:

(1) We propose PilotDB, an efficient and error-bounded AQP method without modifying the existing DBMS or extra effort of the DBA in maintenance.
(2) We provide theoretical analysis and results for (1) the block sampling for efficient AQP and (2) a two-phase sampling scheme to provide a priori guarantees for AQP errors.
(3) The evaluation of PilotDB on four workloads for big data analytics shows up to two orders of magnitude speedups.

2 RELATED WORK

In this section, we briefly review the related work on AQP from the perspective of online AQP, offline AQP, and online aggregation (OLA). We highlight that none of the existing methods with a priori error guarantees can avoid extra maintenance efforts or modifications to DBMSs.

Online AQP. Generating and analyzing samples of large tables upon query arrival is widely studied in prior AQP systems and algorithms [21, 36, 46, 47, 63, 66, 67]. Prior work has considered random sampling as a standard operation in query processing to estimate aggregates and use analytical confidence intervals to measure the error of estimations [46, 47, 63]. As an alternative method to construct confidence intervals, bootstrap techniques were studied for AQP, where many resamples are constructed by sampling from the initial sample [54, 66, 67]. However, since bootstrap is simulation-based, it cannot be applied to provide a priori error guarantees. As a step further to accelerate complex queries, Quickr injects sampling operations in the query plan level and supports lazy execution [36]. Additionally, Idea explored reusing previous query results to accelerate future approximate queries [21]. More recently, Taster explored the combination of online and offline samples by caching the online samples and reusing them for future queries to achieve faster execution [48].

Although existing online AQP systems can provide reliable answers by measuring post-hoc errors, they fail to provide a priori guarantees of errors or allow users to specify estimation error constraints in query processing. Moreover, recent advances in reducing query costs involve injecting samplers into the query plan for lazy execution. This requires significant modifications to the query optimization and execution engine of existing DBMSs. Despite these modifications, state-of-the-art methods still slow down a part of queries compared with exact execution [36, 48] or lead to errors as big as 100% [36]. Consequently, practitioners may be discouraged from deploying those advancements [43].

Offline AQP. There are essentially two types of offline AQP methods developed in prior work. First, prior work investigates non-sampling methods, such as sketches [10, 23, 51], histograms [53, 55, 58], and wavelets [24, 45]. Their primary idea is compressing or summarizing columns through numeric transformations. Therefore, they cannot process queries with non-numeric columns, such as categorical columns, or with complex relational operations, such as join and grouping. In this case, these offline AQP methods are not suitable for processing general analytical queries.

Second, prior work developed the method of generating offline samples to answer online queries [1–6, 8, 9, 18, 22, 41, 48, 52]. Aqua first developed the method of rewriting queries with pre-computed samples to answer approximate queries [1–4]. Subsequently, various optimizations in offline sample creation have been proposed, such as weighted sampling [6, 22], stratified sampling [9], and outlier index [8]. Prior work has explored guaranteeing errors a priori by generating specialized samples for non-nested queries [41], sparse data distribution [64], queries over specific columns [5], and queries with specific selectivities [18]. Furthermore, VerdictDB proposed to develop offline AQP as a middleware to avoid modifications to DBMSs [52].

Offline sampling-based AQP methods have two fundamental limitations. First, a priori error guarantees are inherently limited to predictable workloads [5, 18, 41, 64]. For example, BlinkDB requires that incoming queries output columns in a pre-defined column set; Sample+Seek relies on the pre-knowledge of the query selectivity to select the right processing policy (i.e., sample or seek); Baq only supports non-nested queries. Moreover, maintaining offline samples requires special effort and costs, including regularly refreshing samples to ensure statistical correctness and regenerating samples when the database changes [1, 5, 52]. Nevertheless, these offline methods can be applied to complement PilotDB to achieve a more comprehensive query acceleration.

Online Aggregation. Prior work has studied processing aggregation queries interactively by providing answers as soon as the query is issued and making the answer more accurate as more data is sampled [7, 19, 25, 26, 32, 59, 62, 65]. The notion of OLA was first proposed by Hellerstein et al. [26], and has since been improved to support join queries [25, 39], scalable processing on large databases [19, 32], processing multiple queries simultaneously [62], and complex aggregates [65]. Furthermore, ProgressiveDB explored online aggregation as an extension to existing DBMSs using progressive views [7]. More recently, DeepOLA tackled nested queries for online aggregation [59].

Although OLA techniques keep updating the confidence interval, it is invalid to consider the monitored confidence interval as an error guarantee due to the problem of peeking at early results [34]. Nevertheless, OLA can be integrated with the second phase of PilotDB to provide constantly updating results, thereby improving the interactivity and user experience.

3 OVERVIEW

In this section, we present a high-level overview. We explain the following aspects about PilotDB:

(1) What are the semantics of a prior error guarantees (§7.1)?
(2) How to deal with multiple error events in the query (§3.2)?
(3) Which queries do PilotDB support (§3.3)?
(4) How do PilotDB operate at the high level (§3.4)?

3.1 Error Semantics

We first describe two types of errors that PilotDB guarantees, which is independent of the sampling method or AQP algorithm. Given a general aggregation query, PilotDB considers two error events: (1) Aggregate Error: the estimation error of each aggregate being larger than a user-specified value e, and (2) Group Error: missing groups of size larger than a user-specified value g. Aggregate error is a common error type considered by existing offline AQP methods that guarantee errors [5, 18, 41, 64]. Offline AQP systems eliminate group errors by scanning the entire data and storing all group-wise stratified samples, which is infeasible for online AQP. On the other hand, most DBMSs only support random samplers that inevitably miss groups when the group size is small.[1] In this case, PilotDB supports a constrained group error event, which occurs when there is a missed group of size larger than a user-specified value g. Overall, PilotDB guarantees that the probability of either error occurring is less than a user-specified value p.

Formal Description. Now, we describe our error events formally. Let G, G' be the groups returned by the exact query and the approximate query, respectively, C be the number of aggregate columns, and $A_{i,j}, \hat{A}_{i,j}$ be the exact aggregate and approximate aggregate at group G_i and column j. The group error event is defined as

$$\textit{Group Error} := \{G_i | G_i \in G, |G_i| > g\} \neq \{G_i | G_i \in G', |G_i| > g\}$$

The aggregate error event is defined as

$$\textit{Aggregate Error} := \exists_{G_i \in G', |G_i| > g},\ \exists_{1 \leq j \leq C},\ \left| \frac{A_{G_i,j} - \hat{A}_{G_i,j}}{A_{G_i,j}} \right| > e$$

For the user-specified maximum aggregate error e, minimum group size g, and maximum failure probability p, we guarantee that

$$\mathbb{P}\left[\textit{Group Error} \bigvee \textit{Aggregate Error}\right] \leq p \tag{1}$$

3.2 Error Decomposition

Here, we introduce the method of error decomposition to analyze multiple error events simultaneously. We find that the group error and aggregate error can be correlated, making the analysis of Inequality 1 complex. For example, increasing the sample size will decrease the probability of both group error and aggregate error. To address the correlation, we introduce Lemma 3.1 to decompose the overall failure probability into the failure probabilities of individual error events. Lemma 3.1 shows that, to ensure the total failure probability, it is sufficient to guarantee the sum of probabilities of the group error and individual aggregate error. Therefore, it is sufficient to analyze the group error and individual aggregate error separately and ensure the sum of probability is less than p.

Lemma 3.1. *(Error Decomposition) Let $p^{(g)}$ be the probability of missing groups with size larger than g (Eq. 2), and $p_{i,j}^{(a)}$ be the*

[1] Quickr is an online AQP system that eliminates group errors by distinct sampler, which is not supported by existing DBMSs [36].

probability of the relative error of the estimated aggregate $\hat{A}_{i,j}$ being larger than e (Eq. 3).

$$p^{(g)} := \mathbb{P}\left[\left\{G_i | G_i \in G, G_i \notin G', |G_i| > g\right\} \neq \emptyset\right] \quad (2)$$

$$p^{(e)}_{i,j} := \mathbb{P}\left[\left|\frac{A_{i,j} - \hat{A}_{i,j}}{A_{i,j}}\right| > e\right] \quad (3)$$

The overall probability of the group error or aggregate error occurring is upper bounded by the sum of $p^{(g)}$ and all $p^{(a)}_{i,j}$ s. Namely,

$$\mathbb{P}\left[\textit{Group Error} \bigvee \textit{Aggregate Error}\right] \leq \sum_{\substack{G_i \in G' \\ |G_i| > g}} \sum_{j=1}^{C} p^{(a)}_{i.j} + p^{(g)} \quad (4)$$

Proof. We first rewrite the probability of Aggregate Error or Group Error with a union of events.

$$\begin{aligned}
&\mathbb{P}\left[\text{Group Error} \bigvee \text{Aggregate Error}\right] \\
&= \mathbb{P}\left[\left(\left\{G_i | G_i \in G, G_i \notin G', |G_i| > g\right\} \neq \emptyset\right)\right. \\
&\quad \left.\bigcup \left(\exists_{G_i \in G', |G_i| > g},\ \exists_{j \in [1,C]},\ \left|\frac{A_{G_i,j} - \hat{A}_{G_i,j}}{A_{G_i,j}}\right| > e\right)\right] \\
&= \mathbb{P}\left[\left(\left\{G_i | G_i \in G, G_i \notin G', |G_i| > g\right\} \neq \emptyset\right)\right. \\
&\quad \left.\bigcup \left(\bigcup_{G_i \in G', |G_i| > g} \bigcup_{j \in [1,C]} \left(\left|\frac{A_{G_i,j} - \hat{A}_{G_i,j}}{A_{G_i,j}}\right| > e\right)\right)\right]
\end{aligned}$$

Next, we apply Boole's inequality to rewrite the probability of unions of events with the sum of probabilities of events.

$$\begin{aligned}
&\mathbb{P}\left[\text{Group Error} \bigvee \text{Aggregate Error}\right] \\
&\leq \mathbb{P}\left[\left(\left\{G_i | G_i \in G, G_i \notin G', |G_i| > g\right\} \neq \emptyset\right)\right] \\
&\quad + \sum_{\substack{G_i \in G' \\ |G_i| > g}} \sum_{j=1}^{C} \mathbb{P}\left[\left|\frac{A_{i,j} - \hat{A}_{i,j}}{A_{i,j}}\right| > e\right] \\
&= p^{(g)} + \sum_{\substack{G_i \in G' \\ |G_i| > g}} \sum_{j=1}^{C} p^{(e)}_{i.j}
\end{aligned}$$

□

3.3 Supported Queries

Now, we describe what queries are supported by PilotDB. Figure 2 describes the syntax of supported queries in the BNF form. As shown, PilotDB supports general aggregation queries with optional `GROUP BY`, `HAVING`, `ORDER BY`, and `LIMIT` clauses. Currently, PilotDB supports `COUNT`, `SUM`, and `AVG` aggregates, as well as their arithmetic combinations, including additions, multiplications, and divisions. PilotDB does not support `COUNT DISTINCT` whose error is challenging to bound even with offline samples [5]. PilotDB does not support extreme aggregates, including `MAX` and `MIN`. If there is an unsupported aggregate in the query, PilotDB will fall back to execute the exact query.

Query	Q	::= **SELECT** L **FROM** T **WHERE** ψ
		| **SELECT** L **FROM** T **GROUP BY** $cols$ **HAVING** ψ
		| **SELECT** L **FROM** T **GROUP BY** $cols$ **HAVING** ψ **ORDER BY** $cols$
		| **SELECT** L **FROM** T **GROUP BY** $cols$ **HAVING** ψ **ORDER BY** $cols$ **LIMIT** *an positive integer*
Table	T	::= *an input table*
		| **SELECT** $cols$ **FROM** T **WHERE** ψ
		| T **INNER JOIN** T **ON** ψ | T **CROSS JOIN** T **ON** ψ
		| T **LEFT JOIN** T **ON** ψ | T **RIGHT JOIN** T **ON** ψ
		| T **UNION ALL** T
Target List	L	::= $[col$ **AS** $alias, \cdots, col$ **AS** $alias]$, $[t$ **AS** $alias, \cdots, t$ **AS** $alias]$
Aggregate	t	::= $\alpha(col)$ | t aop t
Col. List	$cols$	::= $[col, \cdots, col]$
Column	col	::= $alias$ | *a column from an input table*
Condition	ψ	::= E op E | $\psi \wedge \psi$ | $\psi \vee \psi$
Expression	E	::= col | $\alpha(col)$ | $const$ | T | Q
Agg. Func.	α	::= **AVG** | **SUM** | **COUNT**
Operator	op	::= $\neq$ | $=$ | $>$ | $<$ | $\geq$ | $\leq$ | **IN** | **NOT IN**
Arith. Op.	aop	::= $+$ | $\times$ | $\div$

Figure 2: Syntax of supported queries.

Algorithm 1: TPS Algorithm of PilotDB.

Input : An input query Q_{in}, the maximum relative error e, the minimum group size g, and the maximum failure probability p

Output : An approximate result R

1 $Q_{pilot} \leftarrow$ PilotRewriter$(Q_{in}, g, p/2)$ `/* Phase-1 */`
2 $R_{pilot} \leftarrow$ Execute(Q_{pilot})
3 $\theta \leftarrow$ EstimateSampleRate$(Q_{in}, R_{pilot}, e, p/2)$
4 $Q_{final} \leftarrow$ FinalRewriter(Q_{in}, θ) `/* Phase-2 */`
5 $R \leftarrow$ Execute(Q_{final})
6 **return** R

Users can specify most of the valid SQL clauses in the query. In the `FROM` clause, users may specify a single table or multiple tables merged with `JOIN` and/or `UNION ALL` operations. In the `WHERE` clause, users may specify arbitrary selection predicates, comparison subqueries (e.g., `quantity > (SELECT AVG(quantity) FROM T)`), and set subqueries (e.g., `T1.name IN (SELECT name FROM T2)`). In the `HAVING` clause, users may specify subqueries (e.g., `SUM(quantity) > (SELECT SUM(quantity)*0.01 FROM T)`) and arbitrary aggregation predicates except for equality predicates. Equality predicates, such as `COUNT(*)=1000`, have a probability of 0, which is impossible to estimate. For the same reason, PilotDB does not support aggregates as `GROUP BY` columns. Furthermore, if any of the subqueries is correlated with the outer-query, PilotDB tries to rewrite the correlated subquery with `JOIN` with pre-defined rules. If PilotDB fails to apply rules, we fall back to execute the exact query, since correlated subqueries are known to be slow even with sampling [52].

3.4 Algorithm Overview

We present the two-phase sampling scheme of PilotDB: TPS, in Algorithm 1. TPS accepts an input query Q_{in} with three error specifications: e specifies the maximum relative error for the aggregate error event; g specifies the minimum group size for the group error

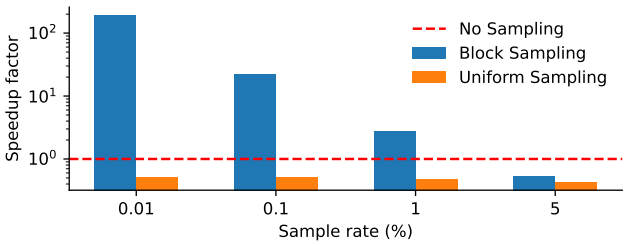


Figure 3: Block sampling can be up to 370× faster than uniform sampling while uniform sampling can be slower than no sampling.

event; p specifies the maximum probability of either event occurring. TPS returns an approximate result R to Q_{in}, which guarantees that the error requirements are satisfied.

Next, we introduce the details of the algorithm. On receiving the query, PilotDB rewrites the original query Q_{in} into a pilot query Q_{pilot} that calculates the necessary statistics to guarantee aggregate errors. Based on the specified g, PilotDB automatically calculates the sample rate such that Q_{pilot} would encounter a group error with a probability less than half of the probability budget (i.e., $p/2$). Next, PilotDB sends Q_{pilot} to the DBMS for execution. After the DBMS returns the pilot query result R_{pilot}, PilotDB automatically calculates the sample rate θ such that Q_{final} would encounter an aggregate error with a probability less than half of the probability budget (i.e., $p/2$). Then, PilotDB rewrites Q_{in} into the final query Q_{final} with sampling, based on θ. Finally, PilotDB sends Q_{final} to the DBMS for execution and returns the final query result R to the user. Additionally, PilotDB always rewrites queries with block sampling to reduce the overhead of sampling on the fly.

The rest of the paper expands on the designs and theories of TPS in the following way. Section 4 investigates using block sampling for AQP with error guarantees. Section 5 presents the theories of sample rate estimation for a requested relative error rate of common aggregates with a guaranteed probability. Section 6 describes the rules to rewrite Q_{in} into Q_{pilot} and Q_{final} that are supported by most DBMSs.

4 BLOCK SAMPLING FOR EFFICIENT ONLINE QUERY PROCESSING

In this section, we investigate block sampling to tackle the challenge of efficient sampling-based AQP without offline samples or modifications to DBMSs.

We first describe our motivations for using block sampling instead of the standard method: uniform sampling. Intuitively, uniform sampling is expensive in modern DBMSs that store data in pages, since it requires random access to data. For example, sampling 5% data from a database with 100 data per page would require accessing 99% pages ($1-0.95^{100}$) [57]. Empirically, we evaluated the runtime of TPC-H query 6 on PostgreSQL with different sampling methods as a motivating example. We evaluated each sample rate 10 times. As shown in Figure 3, uniform sampling can be slower than no sampling by up to 2.3×. In contrast, block sampling can be up to 191× faster than no sampling and 370× faster than uniform sampling. This is because block sampling uses physical pages as the sampling unit and only accesses sampled pages. Considering the significant benefit in efficiency, we use block sampling in PilotDB.

However, block sampling brings challenges to error analysis since it does not produce identically and independently distributed (i.i.d.) samples. In the rest of this section, we develop page-oriented estimators and error propagation schemes to analyze block sampling in simple aggregation queries (§4.1). For complex queries, we introduce and prove commutative properties to analyze the interactions between block sampling and relational operations (§4.2). Finally, we analyze the group error in block sampling (§4.3).

4.1 Simple Aggregation Queries

We first investigate how to deal with the following simple aggregation queries that compute aggregates directly on the input table.

```
SELECT AGG(x1), ..., AGG(xn) FROM T
```

The standard way of analyzing the error of such queries is to calculate confidence intervals of aggregates using CLT. However, block sampling results in non-i.i.d. samples of rows, which invalidates the confidence interval computed by standard CLT. Intuitively, block sampling introduces greater variability of the sampling results when the correlation of the data from the same page is stronger. To address this, we develop page-oriented estimators for common aggregates, leveraging the property that block sampling produces i.i.d. samples of pages. We show that our page-oriented estimators are equivalent to widely used row-oriented estimators [25, 26].

Notation. We first summarize the notation we will use for derivation. Suppose we apply the block sampling with sample rate θ over a table T with N pages, resulting in a sample of n pages: $P_1, \ldots, P_n$. We note that N is usually unknown if T is a derived table in the query. Suppose page P_i has Q_i rows: $X_{i,1}, \ldots, X_{i,Q_i}$, where each row represents a value for aggregation. Additionally, we denote the sum of values on page P_i as S_i (i.e., page sum). We denote the sample mean and the expected value of page sizes as $\bar{Q}$ and μ_Q, and the sample mean and the expected value of page sums as $\bar{S}$ and μ_S. Q_i and S_i are the key values that we will use to estimate aggregates.

Page-Oriented Estimators. Now, we introduce our page-oriented estimators for COUNT, SUM, and AVG. Table 1 presents and compares the calculations of page-oriented estimators and their equivalent row-oriented estimators. Specifically, to estimate COUNT, we divide the COUNT of the sample by the sample rate, which can be rewritten as the product of page statistics: n/θ and $\bar{Q}$. Similarly, to estimate SUM, we divide the SUM of the sample by the sample rate, which can be rewritten as the product of page statistics: n/θ and $\bar{S}$. Finally, to estimate AVG, we divide the sample sum and the sample size, which can be rewritten as the ratio of page statistics: $\bar{S}$ and $\bar{Q}$.

Error Propagation. Next, we explain how to analyze the errors of our page-oriented estimators. We observe that our page-oriented estimators are products or ratios of statistics whose confidence intervals can be calculated with CLT. For example, AVG is estimated as the ratio between $\bar{S}$ and $\bar{Q}$. Both of them represent the mean of an i.i.d. sample, whose error can be analyzed with CLT. Therefore, to estimate the sample rate for a given relative error of aggregates, the following two steps are sufficient.

Table 1: Transformation of common aggregates from row-oriented estimators to page-oriented estimators equivalently.

Aggregate	Exact value		Estimator	
	Row	Page	Row	Page
COUNT	$\sum_{i=1}^{N} Q_i$	$N \cdot \mu_Q$	$\left(\sum_{i=1}^{n} Q_i\right)/\theta$	$(n/\theta)\cdot \bar{Q}$
SUM	$\sum_{i=1}^{N}\sum_{j=1}^{Q_i} X_{i,j}$	$N \cdot \mu_S$	$\left(\sum_{i=1}^{n}\sum_{j=1}^{Q_i} X_{i,j}\right)/\theta$	$(n/\theta)\cdot \bar{S}$
AVG	μ_X	μ_S/μ_Q	$\left(\sum_{i=1}^{n}\sum_{j=1}^{Q_i} X_{i,j}\right)/\left(\sum_{i=1}^{N} Q_i\right)$	$\bar{S}/\bar{Q}$

Table 2: Summary of upper bounds of relative errors of composite estimators with multiplication, division, and addition.

Composite estimator	Upper bound of relative error
$\hat{x}\cdot\hat{y}$	$e_x + e_y + e_x \cdot e_y$
$\hat{x}/\hat{y}$	$(e_x+e_y)/(1+\min(e_x,e_y))$
$\hat{x}+\hat{y}$	$\max(e_x, e_y)$

(1) We calculate the relative errors of page statistics that achieve the required relative errors of aggregates.
(2) We then estimate the sample rate for the calculated relative error of page statistics.

The first step tries to propagate the error of aggregates to the corresponding statistics that are computed from an i.i.d. sample. Then, the second step will leverage the i.i.d. property to calculate the sample rate for the requested error, which we will introduce in Section 5. Based on prior work in uncertainty propagation [33, 50, 60], we can derive relative error propagation rules for positive estimators through multiplication, division, and addition, as shown in Table 2, where e_x is the relative error of the positive estimator $\hat{x}$, and e_y is the relative error of the positive estimator $\hat{y}$. We only derive sufficient upper bounds, while tighter upper bounds can be obtained by considering the correlation of estimators. The following lemmas explain the propagation rules in detail.

Lemma 4.1. *(Multiplication) Let z be a quantity calculated as the product of two positive quantities x and y (i.e., $z = xy$). We estimate x with $\hat{x}$ and y with $\hat{y}$. Let $e_x(<1)$ be the relative error between $\hat{x}$ and x, and $e_y(<1)$ be the relative error between $\hat{y}$ and y. The relative error between z and $\hat{x}\hat{y}$ has an upper bound of $e_x + e_y + e_x \cdot e_y$.*

Proof. By definition of the relative error and the positiveness of x, we have

$$\left|\frac{\hat{x}-x}{x}\right| \le e_x \quad \Leftrightarrow \quad (1-e_x)x \le \hat{x} \le (1+e_x)x$$

Similarly, we have

$$\left|\frac{\hat{y}-y}{y}\right| \le e_y \quad \Leftrightarrow \quad (1-e_y)y \le \hat{y} \le (1+e_y)y$$

Then, we have

$$\begin{aligned}&(1-e_x)(1-e_y)xy \le \hat{x}\hat{y} \le (1+e_x)(1+e_y)xy\\ \Leftrightarrow\ &(e_xe_y - e_x - e_y)xy \le \hat{x}\hat{y} - xy \le (e_xe_y + e_x + e_y)xy\\ \Leftrightarrow\ &\left|\frac{\hat{x}\hat{y}-xy}{xy}\right| \le e_x + e_y + e_x\cdot e_y\end{aligned}$$

□

Lemma 4.2. *(Division) Let z be a quantity calculated as the ratio of two positive quantities x and y (i.e., $z = x/y$). We estimate the x with $\hat{x}$ and y with $\hat{y}$. Let $e_x(<1)$ be the relative error between $\hat{x}$ and x, and $e_y(<1)$ be the relative error between $\hat{y}$ and y. The relative error between z and $\hat{x}/\hat{y}$ has an upper bound of $\frac{e_x+e_y}{1+\min(e_x,e_y)}$.*

Proof. By the definition of the relative error and the positiveness of x and y, we have

$$\begin{aligned}&\frac{1-e_x}{1+e_y}\frac{x}{y} \le \frac{\hat{x}}{\hat{y}} \le \frac{1+e_x}{1-e_y}\frac{x}{y}\\ \Leftrightarrow\ &-\frac{e_x+e_y}{1+e_x}\frac{x}{y} \le \frac{\hat{x}}{\hat{y}} - \frac{x}{y} \le \frac{e_x+e_y}{1+e_y}\frac{x}{y}\\ \Rightarrow\ &\left|\frac{\hat{x}/\hat{y}-x/y}{x/y}\right| \le \frac{e_x+e_y}{1+\min(e_x,e_y)}\end{aligned}$$

□

Lemma 4.3. *(Addition) Let z be a quantity calculated as the linear combination of two positive quantities x and y. Namely, $z = \lambda_1 x + \lambda_2 y$, where λ_1 and λ_2 are positive. We estimate x with $\hat{x}$ and y with $\hat{y}$. Let $e_x(<1)$ be the relative error between $\hat{x}$ and x, and $e_y(<1)$ be the relative error between $\hat{y}$ and y. The relative error between z and $\lambda_1\hat{x} + \lambda_2\hat{y}$ has an upper bound of $\max(e_x, e_y)$.*

Proof. By the definition of the relative error and the positiveness of x and y, we have

$$\begin{aligned}&(1-e_x)\lambda_1 x + (1-e_y)\lambda_2 y \le \lambda_1\hat{x} + \lambda_2\hat{y} \le (1+e_x)\lambda_1 x + (1+e_y)\lambda_2 y\\ \Leftrightarrow\ &-\lambda_1 e_x x - \lambda_2 e_y y \le \lambda_1\hat{x} + \lambda_2\hat{y} - (\lambda_1 x + \lambda_2 y) \le \lambda_1 e_x x + \lambda_2 e_y y\\ \Rightarrow\ &\left|\frac{\lambda_1\hat{x} + \lambda_2\hat{y} - (\lambda_1 x + \lambda_2 y)}{\lambda_1 x + \lambda_2 y}\right| \le \max(e_x, e_y)\end{aligned}$$

□

Example 4.4 shows an example that propagates the error of the AVG aggregate.

Example 4.4. Consider the AVG aggregate, which can be estimated as $\bar{S}/\bar{Q}$. Suppose we need to guarantee that AVG has a maximum error of 5% with 2% failure probability. According to Table 2, we have the following error propagation.

$$e_{\bar{S}} \le 0.025, e_{\bar{Q}} \le 0.025 \ \Rightarrow\ e_{\mathsf{AVG}} \le \frac{0.025+0.025}{1+0.025} < 0.05$$

Namely, it is sufficient to ensure that $\bar{S}$ and $\bar{Q}$ have maximum errors of 2.5% with 1% failure probability.

4.2 Complex Queries

Unlike simple aggregation queries, complex queries usually compute aggregates over an intermediate table derived by multiple relational operations on the input tables, such as selection, projection, join, and union. However, block sampling is executed when scanning the input tables. Therefore, the possible interactions of

block sampling and relational operations make the analysis challenging. In this section, we address this challenge theoretically.

To illustrate, we take the TPC-H query 6 as an example. In order to save the cost of disk access, we would like to execute block sampling on the input table:

```
SELECT SUM(l_extendedprice * l_discount)
FROM lineitem TABLESAMPLE SYSTEM (0.5%)
WHERE l_shipdate >= DATE '1994-01-01' AND ...
```

However, the error analysis becomes challenging due to the selection between the sampling and aggregation. The following query simplifies the error analysis by executing sampling after the selection, while the query cost is not significantly reduced due to the full scan of the table `lineitem`.

```
SELECT SUM(l_extendedprice * l_discount)
FROM (  SELECT * FROM lineitem
        WHERE l_shipdate >= DATE '1994-01-01' AND ...
    ) AS cte TABLESAMPLE SYSTEM (0.5%)
```

To address the challenge, we introduce the commutative properties that allow us to analyze the above two queries equivalently. We describe the intuition of the theory and defer all detailed proofs to our extended report [69].

We first introduce the notion of the equivalence of sampling designs in Definition 4.5. Intuitively, we consider sampling designs with identical distribution as equivalent sampling designs.

Definition 4.5. Two sampling designs, $\mathcal{S}_1$ and $\mathcal{S}_2$, for a set of k relations $\{R_1, \ldots, R_k\}$, where $k \geq 1$, are said to be equivalent, denoted as

$$\mathcal{S}_1\left(\{R_1, \ldots, R_k\}\right) \Leftrightarrow \mathcal{S}_2\left(\{R_1, \ldots, R_k\}\right)$$

if, for any possible sample result R, the probability of obtaining R is the same under both sampling designs S_1 and S_2, i.e.,

$$\forall R,\ \mathbb{P}\left[\mathcal{S}_1\left(\{R_1, \ldots, R_k\}\right) = R\right] = \mathbb{P}\left[\mathcal{S}_2\left(\{R_1, \ldots, R_k\}\right) = R\right].$$

The equivalence of sampling designs leads to an important property that the statistical distribution of the aggregate is identical for equivalent sampling designs, as shown in Proposition 4.6.

Proposition 4.6. *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two equivalent sampling designs, as defined in Definition 4.5. For any aggregate function f that maps a table to a real value, the probability distribution of the f applied to samples from $\mathcal{S}_1$ is identical to the probability distribution of f applied to the samples from $\mathcal{S}_2$. Namely, for any real value x,*

$$\mathbb{P}\left[f\left(\mathcal{S}_1(\{R_1, \ldots, R_k\})\right) = x\right] = \mathbb{P}\left[f\left(\mathcal{S}_2(\{R_1, \ldots, R_k\})\right) = x\right]$$

Proof. For any value $x \in \mathbb{R}$, let $T = \{R_1^*, \ldots, R_n^*\}$ be the set of all tables on which the aggregate function f results in x. Then, we can rewrite:

$$\begin{aligned}
\mathbb{P}\left[f(\mathcal{S}_1(R_1, \ldots, R_k)) = x\right] &= \mathbb{P}\left[\bigcup_{i=1}^{k} \left(\mathcal{S}_1(R_1, \ldots, R_k) = R_i^*\right)\right] \\
&= \sum_{i=1}^{k} \mathbb{P}\left[RO\left(BS(R_1, \ldots, R_k)\right) = R_i^*\right] \\
\mathbb{P}\left[f(\mathcal{S}_2(R_1, \ldots, R_k)) = x\right] &= \mathbb{P}\left[\bigcup_{i=1}^{k} \left(\mathcal{S}_2(R_1, \ldots, R_k) = R_i^*\right)\right] \\
&= \sum_{i=1}^{k} \mathbb{P}\left[RO\left(BS(R_1, \ldots, R_k)\right) = R_i^*\right]
\end{aligned}$$

Given definition 4.5, for equivalent sampling designs $\mathcal{S}_1$ and $\mathcal{S}_2$, Namely, we have the following equation holds for any $R_i^* \in T$.

$$\mathbb{P}\left[\mathcal{S}_1(R_1, \ldots, R_n) = R_i^*\right] = \mathbb{P}\left[\mathcal{S}_2(R_1, \ldots, R_n) = R_i^*\right]$$

Therefore, for any $x \in \mathbb{R}$, we have

$$\mathbb{P}\left[f\left(\mathcal{S}_1(R_1, \ldots, R_n)\right) = x\right] = \mathbb{P}\left[f\left(\mathcal{S}_1(R_1, \ldots, R_n)\right) = x\right]$$

□

Proposition 4.6 implies that we can analyze the error of the aggregates computed over the sampling design A as if it would be computed over the sampling design B, as long as A and B are equivalent. This allows us to execute the query on a more efficient sampling design while analyzing the error on an equivalent sampling design that is easier to analyze. In our case, we prove that exchanging the order of block sampling and relational operations, including projection, selection, join, bag union, results in equivalent sampling designs. We call that block sampling *commutes* with those relational operations. The case of projection is trivial since projection, which selects specific columns, is an orthogonal operation with block sampling, which samples blocks of rows. We show the commutativity between block sampling and other relational operations in Proposition 4.7, 4.8, and 4.9. Additionally, grouping operation can be considered as a special case of selection.

Proposition 4.7. *(Selection-BS Commutativity) For any table R, selection σ_ϕ with a predicate ϕ, and block sampling $\mathcal{B}_\theta$ with a sample rate θ,*

$$\sigma_\phi(\mathcal{B}_\theta(R)) \Leftrightarrow \mathcal{B}_\theta(\sigma_\phi(R))$$

Proof. First, we describe two events whose probability is zero and independent of the order of block sampling and selection. Suppose the input table R has N pages: $P_1, \ldots, P_N$. Let R^* be the result table of applying block sampling and selection on R.

$E_1 := \exists_{r \in R^*},\ \phi(r) = 0$

$E_2 := \forall_{i \in [1,N]}\ \exists_{r_1, r_2 \in P_i},\ \phi(r_1) = \phi(r_2) = 1$ AND $r_1 \in R^*$ AND $r_2 \notin R^*$

As long as the selection operation is applied, every row in the result table must evaluate the predicate ϕ to 1. Therefore, the probability of E_1 is 0. Furthermore, block sampling ensures that if one row is sampled, the whole page is sampled. Therefore, it is not likely to have two rows from the same page satisfying the predicate, but only one of them is in the result table. Namely, the probability of E_2 is 0.

Next, we analyze the probability distributions excluding events E_1 and E_2. Let $R' = \{P'_1, \ldots, P'_{N'}\}$ be the subset of R satisfying ϕ, where each page P'_i contains non-zero rows satisfying ϕ. If we exclude events E_1 and E_2, the result table must be a subset of pages in R', regardless of the order of operations. We can calculate the inclusion probability of P'_i in R^* for different orders of operations. It turns out that the inclusion probability is independent of the operation order.

$$\mathbb{P}\left[P'_i \in \sigma_\phi(\mathcal{B}_\theta(R))\right] = \mathbb{P}\left[P_i \in \mathcal{B}_\theta(R)\right] = \theta$$
$$\mathbb{P}\left[P'_i \in \mathcal{B}_\theta(\sigma_\phi(R))\right] = \mathbb{P}\left[P'_i \in \mathcal{B}_\theta(R')\right] = \theta$$

Since pages are independent of each other in the process of selection and block sampling, for the result table $R^* = \{P'_1, \ldots, P'_n\}$, we have

$$\begin{aligned}
&\mathbb{P}\left[\sigma_\phi(\mathcal{B}_\theta(R)) = R^*\right] \\
&= \mathbb{P}\left[\left(\bigcap_{i=1}^{n} P'_i \in \sigma_\phi(\mathcal{B}_\theta(R))\right) \bigcap \left(\bigcap_{i=n+1}^{N'} P'_i \notin \sigma_\phi(\mathcal{B}_\theta(R))\right)\right] \\
&= \theta^n (1-\theta)^{N'-n}; \\
&\mathbb{P}\left[\mathcal{B}_\theta(\sigma_\phi(R)) = R^*\right] \\
&= \mathbb{P}\left[\left(\bigcap_{i=1}^{n} P'_i \in \mathcal{B}_\theta(\sigma_\phi(R))\right) \bigcap \left(\bigcap_{i=n+1}^{N'} P'_i \notin \mathcal{B}_\theta(\sigma_\phi(R))\right)\right] \\
&= \theta^n (1-\theta)^{N'-n}
\end{aligned}$$

Therefore, for any result table, the probability is independent of the order of selection and block sampling. Namely, selection and block sampling commutes. □

Proposition 4.8. *(Join-BS Commutativity) For any tables R_1 and R_2, join $\bowtie_\phi$ with a predicate ϕ, and block sampling $\mathcal{B}_\theta$ with a sample rate θ,*

$$\mathcal{B}_\theta(R_1) \bowtie_\theta R_2 \Leftrightarrow \mathcal{B}_\theta(R_1 \bowtie_\phi R_2)$$

Proof. Since block sampling commutes with selection, it is sufficient to prove that block sampling also commutes with cross-product. Without loss of generality, we assume block sampling is executed on R_1. We first describe an event whose probability is zero and is independent of the order of block sampling and cross-product. Suppose R_1 has N pages: $R_1 = \{P_1 \ldots, P_N\}$. Let R^* be the result table after applying block sampling and cross-product over R_1 and R_2.

$$E := \quad \forall_{i\in[1,N]}\, \exists_{r_1,r_2\in P_i},\ (r_1 \bowtie R_2) \in R^* \text{ AND } (r_2 \bowtie R_2) \notin R^*$$

Block sampling ensures that if one row is sampled, the whole page is sampled. Therefore, it is not likely to have one row in the result table while other rows from the same page are not. Namely, the probability of E is 0.

Next, we analyze the probability distributions excluding the event E. Let $R' = \{P'_1, \ldots, P'_N\}$ be the full cross-product where $P'_i = P_i \bowtie R_2$ (i.e., the cross-product of page P_i and table R_2). If we exclude event E, the result table R must be a subset of pages in R'. We can calculate the inclusion probability of P'_i in R. It turns out that the inclusion probability is independent of the operation order.

$$\mathbb{P}\left[P'_i \in (\mathcal{B}_\theta(R_1) \bowtie R_2)\right] = \mathbb{P}\left[P_i \in \mathcal{B}_\theta(R_1)\right] = \theta$$
$$\mathbb{P}\left[P'_i \in \mathcal{B}_\theta(R_1 \bowtie R_2)\right] = \mathbb{P}\left[P'_i \in \mathcal{B}_\theta(R')\right] = \theta$$

Since pages are independent of each other in the process of cross-product and block sampling, for the arbitrary result table $R^* = \{P'_1, \ldots, P'_n\}$, we have

$$\begin{aligned}
&\mathbb{P}\left[(\mathcal{B}_\theta(R_1) \bowtie R_2) = R^*\right] \\
&= \mathbb{P}\left[\left(\bigcap_{i=1}^{n} P'_i \in (\mathcal{B}_\theta(R_1) \bowtie R_2)\right) \bigcap \left(\bigcap_{i=n+1}^{N} P'_i \notin (\mathcal{B}_\theta(R_1) \bowtie R_2)\right)\right] \\
&= \theta^n (1-\theta)^{N-n}; \\
&\mathbb{P}\left[\mathcal{B}_\theta(R_1 \bowtie R_2) = R^*\right] \\
&= \mathbb{P}\left[\left(\bigcap_{i=1}^{n} P'_i \in \mathcal{B}_\theta(R_1 \bowtie R_2)\right) \bigcap \left(\bigcap_{i=n+1}^{N} P'_i \notin \mathcal{B}_\theta(R_1 \bowtie R_2)\right)\right] \\
&= \theta^n (1-\theta)^{N-n}
\end{aligned}$$

Therefore, for any result table, the probability is independent of the order of cross-product and block sampling. Namely, cross-product and block sampling commute. Since selection and block sampling commute, join and block sampling commute. □

Proposition 4.9. *(Union-BS Commutativity) Let $\cup$ be a bag union operation. For any tables $R_1, \ldots, R_k$ with $k \geq 2$, and block sampling $\mathcal{B}_\theta$ with a sample rate θ,*

$$\bigcup_{i=1}^{k} \mathcal{B}_\theta(R_i) \Leftrightarrow \mathcal{B}_\theta\left(\bigcup_{i=1}^{k} R_i\right)$$

Proof. First, we describe an event whose probability is zero and independent of the order of union and block sampling. Suppose R_1 has N_1 pages: $R_1 = \{P_1^{(1)}, \ldots, P_{N_1}^{(1)}\}$, and R_2 has N_2 pages: $R_2 = \{P_1^{(2)}, \ldots, P_{N_2}^{(2)}\}$. Let R^* be the result table after applying block sampling and union over R_1 and R_2.

$$E_1 := \quad \forall_{i\in[1,N_1]}\, \exists_{r_1,r_2\in P_i^{(1)}},\ r_1 \in R^* \text{ AND } r_2 \notin R^*$$
$$E_2 := \quad \forall_{i\in[1,N_2]}\, \exists_{r_1,r_2\in P_i^{(2)}},\ r_1 \in R^* \text{ AND } r_2 \notin R^*$$

Block sampling ensures that if one row is sampled, the whole page is sampled. Therefore, it is not likely to have one row in the result table while other rows from the same page are not. Namely, both the probability of E_1 and the probability of E_2 are zero.

Next, we analyze the probability distribution excluding E_1 and E_2. Let $R' = \{P_1^{(1)}, \ldots, P_{N_1}^{(1)}, P_1^{(2)}, \ldots, P_{N_2}^{(2)}\}$ be the union of R_1 and R_2. If we exclude E_1 and E_2, the result table R must be a subset of pages in R'. We can calculate the inclusion probability of $P_i^{(1)}$ and $P_i^{(2)}$ in R. It turns out that the inclusion probability is independent of the operation order. Without loss of generality, we only show the calculation of the inclusion probability of $P_i^{(1)}$.

$$\mathbb{P}\left[P_i^{(1)} \in (\mathcal{B}_\theta(R_1) \cup \mathcal{B}_\theta(R_2))\right] = \mathbb{P}\left[P_i^{(1)} \in \mathcal{B}_\theta(R_1)\right] = \theta$$
$$\mathbb{P}\left[P_i^{(1)} \in \mathcal{B}_\theta(R_1 \cup R_2)\right] = \theta$$

Since pages are independent of each other in the process of union and block sampling, for the arbitrary result table:

$$R^* = \{P_1^{(1)}, \ldots, P_{n_1}^{(1)}, P_1^{(2)}, \ldots, P_{n_2}^{(2)}\}$$

we have

$$\begin{aligned}
&\mathbb{P}\left[(\mathcal{B}_\theta(R_1) \cup \mathcal{B}_\theta(R_2)) = R^*\right] \\
&= \mathbb{P}\Bigg[\left(\bigcup_{i=1}^{n_1} P_i^{(1)} \in \mathcal{B}_\theta(R_1)\right) \bigcup \left(\bigcup_{i=n_1+1}^{N_1} P_i^{(1)} \notin \mathcal{B}_\theta(R_1)\right) \\
&\qquad \bigcup \left(\bigcup_{i=1}^{n_2} P_i^{(2)} \in \mathcal{B}_\theta(R_2)\right) \bigcup \left(\bigcup_{i=n_2+1}^{N_2} P_i^{(2)} \notin \mathcal{B}_\theta(R_2)\right)\Bigg] \\
&= \theta^{n_1} \cdot (1-\theta)^{N_1-n_1} \cdot \theta^{n_2} \cdot (1-\theta)^{N_2-n_2} \\
&= \theta^{n_1+n_2} \cdot (1-\theta)^{N_1+N_2-n_1-n_2}; \\
&\mathbb{P}\left[\mathcal{B}_\theta(R_1 \cup R_2) = R^*\right] \\
&= \mathbb{P}\Bigg[\left(\bigcup_{k=1}^{2}\bigcup_{i=1}^{n_k} P_i^{(k)} \in \mathcal{B}_\theta(R_1 \cup R_2)\right) \\
&\qquad \bigcup \left(\bigcup_{k=1}^{2}\bigcup_{i=n_k+1}^{N_k} P_i^{(k)} \notin \mathcal{B}_\theta(R_1 \cup R_2)\right)\Bigg] \\
&= \theta^{n_1+n_2}(1-\theta)^{N_1-n_1+N_2-n_2}
\end{aligned}$$

Therefore, for any result table, the probability is independent of the order of union and block sampling. Namely, union and block sampling commute. □

4.3 Group Error Analysis

Finally, we introduce the analysis of the group error in block sampling. Group error quantifies the probability of missing groups when using block sampling to approximately process queries with `GROUP BY` clauses. We include the probability of missing groups as part of the failure probability budget, as decomposed in Lemma 3.1.

Scenario. We may miss groups only if the table executing block sampling contains grouping keys. If the sampled pages do not cover a group, we miss the group. Due to the randomness of sampling, we cannot guarantee that no groups are missed. In the extreme case, we have to make a full table scan to guarantee groups with sizes equal to 1 are not missed. Therefore, we provide a conditioned probabilistic guarantee that the probability of missing groups with sizes larger than a user-specified value is small. We achieve the guarantee by configuring the sampling rate of the pilot sampling.

Notation. Let R be the table executing block sampling with a column being the group-by key, m be the size of each page, θ be the sampling rate, p be the maximum failure probability, and g be the minimum group size. Page size can be retrieved from the metadata of the DBMS. We introduce Lemma 4.10 that estimates the required sampling rate of the pilot sampling to ensure that the block sampling does not miss groups of the size greater than g with a maximum failure probability p. We defer the proof to our extended report [69].

Lemma 4.10. *For a table R with page size m, block sampling with a sampling rate θ satisfying the condition below ensures that the probability of missing a group of size greater than g is less than p.*

$$\theta \geq 1 - \left(1 - (1-p)^{\lceil g/m \rceil / |R|}\right)^{1/\lceil g/m \rceil}$$

Proof. Based on the meta-information, we calculate the number of pages in R as $N = |R|/m$. Because each group has at least g rows, each group takes at least $n_0 = \lceil g/m \rceil$ pages. Suppose there are t groups with size larger than g. Let n_i be the number of pages taken by the i-th group. We then have the following constraints

$$t \leq \frac{|R|}{n_0} \tag{5}$$

$$\forall_{1 \leq i \leq t},\ n_i \geq n_0 \tag{6}$$

Based on the process of block sampling, we can calculate the probability of including i-th group as following

$$\mathbb{P}\left[\text{include group i}\right] = 1 - \mathbb{P}\left[\text{miss group i}\right] = 1 - (1-\theta)^{n_i}$$

Next, we calculate the probability of including groups i and j ($i \neq j$). Suppose group i and group j share k pages ($k \geq 0$), then the probability of including two groups has the following lower bound.

$$\begin{aligned}
&\mathbb{P}\left[(\text{include group i}) \wedge (\text{include group j})\right] \\
&= \mathbb{P}\left[\text{include group i} \mid \text{include group j}\right] \cdot \mathbb{P}\left[\text{include group j}\right] \\
&= \big(\mathbb{P}\left[\text{include group i}\right] \cdot \mathbb{P}\left[\text{not include shared pages} \mid \text{include group j}\right] + \\
&\qquad \mathbb{P}\left[\text{include shared pages} \mid \text{include group j}\right]\big) \cdot \mathbb{P}\left[\text{include group j}\right] \\
&= \left(\left(1-(1-\theta)^{n_i}\right) \cdot \frac{n_j - k}{n_j} + \frac{k}{n_j}\right) \cdot \left(1-(1-\theta)^{n_j}\right) \\
&\geq \left(1-(1-\theta)^{n_i}\right) \cdot \left(1-(1-\theta)^{n_j}\right)
\end{aligned}$$

We extend the results to all groups and calculate lower bound of the the probability of including all groups

$$\mathbb{P}\left[\text{include all groups}\right] \geq \prod_{i=1}^{t} \left(1-(1-\theta)^{n_i}\right)$$

Applying the constraints in 5 and 6, we can have the following lower bound for the probability of including all groups

$$\begin{aligned}
&\mathbb{P}\left[\text{include all groups}\right] \geq \prod_{i=1}^{t} \left(1-(1-\theta)^{n_0}\right) \geq \prod_{i=1}^{|R|/n_0} \left(1-(1-\theta)^{n_0}\right) \\
&= \left(1-(1-\theta)^{\lceil g/m \rceil}\right)^{|R|/\lceil g/m \rceil}
\end{aligned}$$

align, if the sampling rate θ satisfies

$$\theta \geq 1 - \left(1 - (1-p)^{\lceil g/m \rceil / |R|}\right)^{1/\lceil g/m \rceil}$$

then

$$\mathbb{P}\left[\text{miss a group}\right] = 1 - \mathbb{P}\left[\text{include all groups}\right] \leq p$$

□

5 A PRIORI ERROR GUARANTEES WITH PILOT SAMPLING

In this section, we present the theoretical guarantees of our two-phase sampling scheme (TPS), to provide a priori error guarantees without offline samples or extra maintenance by DBAs. We start with statistical intuitions behind TPS. Next, based on the page-oriented estimators derived in Section 4, we formulate two problems of sample rate estimations. Finally, we present theoretical solutions to those problems.

Statistical Intuitions. First, when we try to estimate a population parameter with an i.i.d. sample statistic, we can calculate the upper bound of the estimation error before we take the sample if we know the data distribution. This is because the estimation error can be quantified through the width of a confidence interval, which is computable before sampling. For example, the width of a 95% confidence interval of an i.i.d. sample mean is calculated as $2 \cdot 1.96 \cdot \sqrt{\sigma^2/n}$, where σ^2 is the variance and n is the sample size. If we know σ^2, we can calculate the sample size for a given width of the confidence interval, which in turn guarantees the error.

Second, we can obtain a tight interval estimation of parameters of real-world data with a small loss of failure probability. This property is due to fast decaying tails of statistics of real-world data. Following prior work, we assume real-world data is bounded [40]. The sum of bounded random variables follows a sub-Gaussian distribution, whose tail decays at least exponentially fast. This directly follows the generic Chernoff bound, where $M_X(t)$ is the moment generating function:

$$\forall t > 0, \quad \mathbb{P}[X \geq a] \leq M_X(t)e^{-ta}, \ \mathbb{P}[X \leq a] \leq M_X(t)e^{ta}$$

For instance, the sample variance follows a chi-squared distribution, which is upper-bounded by a Gaussian distribution [15]. Based on this, we can obtain a tight upper or lower bound of the variance by sacrificing a small failure probability. Then, we can plug in the interval estimation of the variance into the calculation of the width of the confidence interval to compute an upper bound of the estimation error. Finally, the general scheme of TPS is as follows.

(1) Based on Q_{in}, we take a pilot sample to estimate the upper or lower bound of parameters to calculate the sample rate for a requested error.
(2) We take a final sample with the calculated sample rate to estimate aggregates.

Example 5.1 illustrates the intuitions by deriving the a priori error guarantee for the sample mean.

Example 5.1. Consider the error analysis when estimating the expected value μ with a sample mean $\bar{X}$. A widely used approach to measure the estimation error is to calculate the width of the confidence interval of μ, as shown in Equation 7, where $z_{1-\delta/2}$ is $1-\delta/2$ percentile of a standard normal distribution, n is the sample size, and σ^2 is the variance.

$$\mathbb{P}\left[\left|\mu - \bar{X}\right| \leq z_{1-\delta/2}\sqrt{\sigma^2/n}\right] \geq 1 - \delta \tag{7}$$

When σ^2 is known, we can calculate the sample size for a required error and failure probability. However, in an online scenario, we do not know σ^2. Nevertheless, we can compute the upper bound of σ^2 using the sample variance $\hat{\sigma}^2$ of a pilot sample. Specifically, $\hat{\sigma}^2$ follows a chi-squared distribution centered at σ^2 with a fast decaying tail. Through a pilot sample of sufficient size (e.g., 200), we can obtain an upper bound of σ^2 by scaling up $\hat{\sigma}^2$ and sacrificing a corresponding amount of failure probability. For example, σ^2 is upper-bounded by $1.5 \cdot \hat{\sigma}^2$ with 98% probability. Plugging in the upper bound of σ^2 and considering the corresponding failure probability, we can obtain the following error bound.

$$\mathbb{P}\left[\left|\mu - \bar{X}\right| \leq z_{1-\delta/2}\sqrt{1.5\hat{\sigma}^2/n}\right] \geq 1 - \delta - (1 - 0.98) \tag{8}$$

Finally, we can calculate the sample size for any required error and failure probability.

5.1 Problem Setup

We now describe the concrete problems for sample rate estimations to guarantee aggregate errors in AQP. Based on the page-oriented estimators we introduced in Section 4, we estimate common aggregates through two types of statistics: sample mean and sample size. The sample mean is used to estimate the population mean, while the sample size is used to estimate the population size. We would obtain a non-deterministic sample size because block sampling is implemented based on Bernoulli sampling in existing DBMSs. With Bernoulli sampling, the sample size n follows a binomial distribution $B(N, \theta)$, where N is the population size and θ is the sample rate. In many cases when aggregates are computed over a derived table, N is unknown but necessary to compute `COUNT` and `SUM`. Therefore, we need to estimate N with n/θ.

Given the parameters to estimate, we now describe two sample rate estimation problems for a priori error guarantees with TPS.

Problem 1. *Given a pilot sample with size n_1, sample mean $\bar{X}$, and sample variance $\hat{\sigma}^2$, we draw a final sample with a sample rate θ, which will result in a final sample mean $\bar{Y}$. To guarantee that the relative error of estimating the mean μ with $\bar{Y}$ is less than e with a maximum failure probability p, what is the sufficient condition for θ?*

Problem 2. *Given a pilot sample with size n_1 and sample rate θ_1, we draw a final sample with a sample rate θ, which will result in a sample size n. To guarantee that the relative error of estimating the cardinality N with n/θ is less than e with a maximum failure probability p, what is the sufficient condition for θ?*

5.2 Preliminaries

In this section, we introduce the preliminaries of simple random sampling that results in independently and identically distributed (i.i.d.) samples. We present two lemmas that derive the concentration inequalities for the sample mean and standard deviation of an i.i.d. sample and the number of successes in a sequence of independent Bernoulli experiments.

Lemma 5.2. *Let $X_1, \ldots, X_n$ be an i.i.d. random sample from a distribution with the expected value given by μ and finite variance given by σ^2. Suppose $\bar{X}$ is the sample mean (Eq. 9) and $\hat{\sigma}^2$ is the unbiased sample variance (Eq. 10).*

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{9}$$

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} \left(X_i - \bar{X}\right)}{n-1} \tag{10}$$

We have the following probabilistic bounds for μ and σ.

$$\lim_{n\to\infty} \mathbb{P}\left[L_\mu^{(t)}(\bar{X},\hat{\sigma},n,\delta/2) \le \mu \le U_\mu^{(t)}(\bar{X},\hat{\sigma},n,\delta/2)\right] = 1-\delta$$
$$\lim_{n\to\infty} \mathbb{P}\left[L_\sigma(\hat{\sigma},n,\delta/2) \le \sigma \le U_\sigma(\hat{\sigma},n,\delta/2)\right] = 1-\delta$$

where

$$L_\mu^{(t)}(\bar{X},\hat{\sigma},n,\delta) = \bar{X} - t_{n-1,1-\delta}\frac{\hat{\sigma}}{\sqrt{n}}$$
$$U_\mu^{(t)}(\bar{X},\hat{\sigma},n,\delta) = \bar{X} + t_{n-1,1-\delta}\frac{\hat{\sigma}}{\sqrt{n}} \tag{11}$$
$$L_\sigma(\hat{\sigma},n,\delta) = \sqrt{\frac{n-1}{\chi^2_{n-1,1-\delta}}}\hat{\sigma}$$
$$U_\sigma(\hat{\sigma},n,\delta) = \sqrt{\frac{n-1}{\chi^2_{n-1,\delta}}}\hat{\sigma} \tag{12}$$

$t_{n-1,1-\delta/2}$ is the $1-\delta/2$ percentile of student's t distribution with $n-1$ degrees of freedom. Namely, $\delta/2 = \mathbb{P}\left[T > t_{n-1,1-\delta/2}\right]$, where T follows student's t distribution with $n-1$ degrees of freedom. $\chi^2_{n-1,\delta/2}$ is the $\delta/2$ percentile of chi-squared distribution with $n-1$ degrees.

If we know the variance σ^2, we can have the following probabilistic bound for μ.

$$\lim_{n\to\infty} \mathbb{P}\left[L_\mu^{(z)}(\bar{X},\sigma,n,\delta/2) \le \mu \le U_\mu^{(z)}(\bar{X},\sigma,n,\delta/2)\right] = 1-\delta$$

where

$$L_\mu^{(z)}(\bar{X},\sigma,n,\delta) = \bar{X} - z_{1-\delta}\frac{\sigma}{\sqrt{n}}$$
$$U_\mu^{(z)}(\bar{X},\sigma,n,\delta) = \bar{X} + z_{1-\delta}\frac{\sigma}{\sqrt{n}}, \tag{13}$$

and $z_{1-\delta/2}$ is the $1-\delta/2$ percentile of standard normal distribution.

Lemma 5.3. *Suppose we have a sequence of N independent experiments. Each experiment is either successful or not successful with a fixed success probability θ. Suppose $\hat{n}$ is the number of successes. We have the following probabilistic bounds for $\hat{n}$.*

$$\lim_{N\to\infty} \mathbb{P}\Big[N\theta - z_{1-\delta/2}\sqrt{N\theta(1-\theta)} \le \hat{n} \le N\theta + z_{1-\delta/2}\sqrt{N\theta(1-\theta)}\Big] = 1-\delta \tag{14}$$

or equivalently

$$\lim_{N\to\infty} \mathbb{P}\left[N - z_{1-\delta/2}\sqrt{\frac{N(1-\theta)}{\theta}} \le \frac{\hat{n}}{\theta} \le N + z_{1-\delta/2}\sqrt{\frac{N(1-\theta)}{\theta}}\right] = 1-\delta \tag{15}$$

where $z_{1-\delta/2}$ is the $1-\delta/2$ percentile of standard normal distribution.

Lemma 5.2 defines the confidence interval of the expected value and variance with an i.i.d. sample. It follows from the Central Limit Theorem [28]. Lemma 5.3 defines the confidence interval of the number of successes in a binomial distribution. It follows the Central Limit Theorem with a standard normal approximation [28].

Based on probabilistic bounds derived in Lemma 5.2 and Lemma 5.3, we observe that the error of using an i.i.d. sample to estimate mean (Eq. 11 and 13) or total number of Bernoulli experiments (Eq. 15) monotonically decreases at the order of $-1/2$ with respect to the sample size n or sample rate θ. Therefore, the key to guaranteeing errors is to estimate a sufficient sample size or sample rate to ensure the error is lower than the user's specification, which will be introduced in the next section.

5.3 Sample Rate Estimation for Mean

In this subsection, we present the theoretical result that solves Problem 1. We describe the intuition and defer the detailed proof to our extended report [69].

We can obtain an upper bound of the relative error using the confidence interval of the mean μ, as shown in Equation 16, where δ_1 is an adjustable failure probability, and $z_{1-\delta_1/2}$ is the $1-\delta_1/2$ percentile of the standard normal distribution.

$$\mathbb{P}\left[\left|\frac{\bar{Y}-\mu}{\mu}\right| \le \frac{z_{1-\delta_1/2}}{\sqrt{n}}\frac{\sigma}{|\mu|}\right] \ge 1-\delta_1 \tag{16}$$

Based on Equation 16, we observe that we can calculate the sample size for a given relative error bound and failure probability with the estimation of σ and μ.

Therefore, there are two steps to derive the condition of the sample rate for guaranteed error of the mean. First, we obtain a sufficient condition of the sample size, as shown in Theorem 5.4.

Theorem 5.4. *Consider a two-phase sampling scheme that estimates the mean μ. The pilot i.i.d. sample has a sample size n_1, a sample mean $\bar{X}$, and a sample std $\hat{\sigma}$, while the final i.i.d. sample of size n will have a sample mean $\bar{Y}$. To guarantee that the relative error between μ and $\bar{Y}$ is less than e with a maximum failure probability p, it is sufficient to ensure the sample size n of the final sample satisfies*

$$\exists_{\delta_1,\delta_2,\delta_3>0}:\ \delta_1+\delta_2+\delta_3 \le p,\ n \ge \left(\frac{z_{1-\delta_1/2}}{e}\frac{U_\sigma(\hat{\sigma},n_1,\delta_2)}{L_\mu(\bar{X},\hat{\sigma},n_1,\delta_3)}\right)^2$$

where U_σ is the upper bound of the std given by Equation 17, L_μ is the lower bound estimation of the μ given by Equation 18, and z_k is the k percentile of the standard normal function.

$$U_\sigma(\hat{\sigma},n_1,\delta_2) = \sqrt{\frac{n_1-1}{\chi^2_{n_1-1,\delta_2}}}\hat{\sigma} \tag{17}$$
$$L_\mu(\bar{X},\hat{\sigma},n_1,\delta_3) = \hat{\mu} - t_{n_1-1,1-\delta_3}\frac{\hat{\sigma}}{\sqrt{n}} \tag{18}$$

Similarly, $t_{d,k}$ is the k percentile of the student's t distribution with d degrees of freedom, while $\chi^2_{d,k}$ is the k percentile of the chi-squared distribution with d degrees of freedom.

Proof. We apply the one-sided version of the probabilistic bounds derived in Lemma 5.2 on the pilot sample result, resulting in

$$\mathbb{P}\left[\sigma \le U_\sigma(\hat{\sigma},n_1,\delta_1)\right] = 1-\delta_1 \tag{19}$$
$$\mathbb{P}\left[\mu \ge L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)\right] = 1-\delta_2 \tag{20}$$

Next, we apply the probabilistic bounds derived in Lemma 5.2 on the final sample result, resulting in

$$\mathbb{P}\left[\bar{Y} - z_{1-\delta_3/2}\frac{\sigma}{\sqrt{n}} \le \mu \le \bar{Y} + z_{1-\delta_3/2}\frac{\sigma}{\sqrt{n}}\right] = 1-\delta_3$$
$$\Leftrightarrow \mathbb{P}\left[-z_{1-\delta_3/2}\frac{\sigma}{\sqrt{n}} \le \bar{Y} - \mu \le z_{1-\delta_3/2}\frac{\sigma}{\sqrt{n}}\right] = 1-\delta_3$$
$$\Leftrightarrow \mathbb{P}\left[\left|\bar{Y}-\mu\right| \le z_{1-\delta_3/2}\frac{\sigma}{\sqrt{n}}\right] = 1-\delta_3 \tag{21}$$

Based on Equation 28, we then derive the probabilistic bounds for the relative error between the sample mean and population mean.

$$\mathbb{P}\left[\left|\frac{\bar{Y}-\mu}{\mu}\right| \le \frac{z_{1-\delta_3/2}}{\sqrt{n}}\frac{\sigma}{\mu}\right] \ge 1-\delta_3 \tag{22}$$

Suppose the sample size of the final sample satisfies

$$n \ge \left(\frac{z_{1-\delta_3/2}}{e}\frac{U_\sigma(\hat{\sigma},n_1,\delta_1)}{L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)}\right)^2 \tag{23}$$

Then, we have

$$\sigma \le U_\sigma(\hat{\sigma},n_1,\delta_1) \quad \text{AND} \quad \mu \ge L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)$$
$$\text{AND} \quad \left|\frac{\bar{Y}-\mu}{\mu}\right| \le \frac{z_{1-\delta_3/2}}{\sqrt{n}}\frac{\sigma}{\mu} \tag{24}$$
$$\Rightarrow$$
$$\left|\frac{\bar{Y}-\mu}{\mu}\right| \le z_{1-\delta_3/2}\left(\frac{z_{1-\delta_3/2}}{e}\frac{U_\sigma(\hat{\sigma},n_1,\delta_1)}{L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)}\right)^{-1}\frac{U_\sigma(\hat{\sigma},n_1,\delta_1)}{L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)}$$
$$= e$$

Namely, given a sample size satisfying Equation 30, if the relative error between $\bar{Y}$ and μ is larger than e, then one of the inequalities in 31 does not hold. Given the failure probability of inequalities in Equation 26, 27, and 29, we can derive the upper bound of the failure probability of the relative error

$$\mathbb{P}\left[\left|\frac{\bar{Y}-\mu}{\mu}\right| \ge e\right] \le \mathbb{P}\left[\left(\sigma \ge U_\sigma(\hat{\sigma},n_1,\delta_1)\right) \vee \left(\mu \le L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)\right) \vee \left(\left|\frac{\bar{Y}-\mu}{\mu}\right| \ge \frac{z_{1-\delta_3/2}}{\sqrt{n}}\frac{\sigma}{\mu}\right)\right]$$
$$\le \mathbb{P}\left[\sigma \ge U_\sigma(\hat{\sigma},n_1,\delta_1)\right] + \mathbb{P}\left[\mu \le L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)\right] + \mathbb{P}\left[\left|\frac{\bar{Y}-\mu}{\mu}\right| \ge \frac{z_{1-\delta_3/2}}{\sqrt{n}}\frac{\sigma}{\mu}\right]$$
$$= \delta_1+\delta_2+\delta_3$$

If we configure $\delta_1+\delta_2+\delta_3 \le p$, we have

$$\mathbb{P}\left[\left|\frac{\bar{Y}-\mu}{\mu}\right| \ge e\right] < p$$

□

Second, we estimate the sufficient sample rate to guarantee the sample size, as shown in Theorem 5.5.

Theorem 5.5. *Consider two sequences of Bernoulli trials, each with an unknown number of trials N. The first sequence (i.e., pilot sample) has a success probability θ_1 and results in n_1 successes. The second sequence has a success probability θ and results in $\hat{n}$ successes. To guarantee that $\hat{n}$ is greater than a desired value n with a maximum failure probability p, it is sufficient to ensure that θ satisfies*

$$\exists_{\delta_1,\delta_2>0}:\ \delta_1+\delta_2 \le p,\ \theta \ge \left(\frac{z_{1-\delta_2}+\sqrt{z_{1-\delta_2}^2+4n}}{2\sqrt{L_N(n_1,\theta_1,\delta_1)}}\right)^2$$

where L_N is the lower bound of N given by Equation 25, and z_k is the k percentile of the standard normal distribution.

$$L_N(n,\theta,\delta) = \left(\sqrt{\frac{n}{\theta}+z_{1-\delta}^2\frac{1-\theta}{2\theta}} - \sqrt{z_{1-\delta}^2\frac{1-\theta}{2\theta}}\right)^2 \tag{25}$$

Proof. We apply the one-sided version of the probabilistic bounds derived in Lemma 5.2 on the pilot sample result, resulting in

$$\mathbb{P}\left[\sigma \le U_\sigma(\hat{\sigma},n_1,\delta_1)\right] = 1-\delta_1 \tag{26}$$
$$\mathbb{P}\left[\mu \ge L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)\right] = 1-\delta_2 \tag{27}$$

Next, we apply the probabilistic bounds derived in Lemma 5.2 on the final sample result, resulting in

$$\mathbb{P}\left[\bar{Y} - z_{1-\delta_3/2}\frac{\sigma}{\sqrt{n}} \le \mu \le \bar{Y} + z_{1-\delta_3/2}\frac{\sigma}{\sqrt{n}}\right] = 1-\delta_3$$
$$\Leftrightarrow \mathbb{P}\left[-z_{1-\delta_3/2}\frac{\sigma}{\sqrt{n}} \le \bar{Y} - \mu \le z_{1-\delta_3/2}\frac{\sigma}{\sqrt{n}}\right] = 1-\delta_3$$
$$\Leftrightarrow \mathbb{P}\left[\left|\bar{Y}-\mu\right| \le z_{1-\delta_3/2}\frac{\sigma}{\sqrt{n}}\right] = 1-\delta_3 \tag{28}$$

Based on Equation 28, we then derive the probabilistic bounds for the relative error between the sample mean and population mean.

$$\mathbb{P}\left[\left|\frac{\bar{Y}-\mu}{\mu}\right| \le \frac{z_{1-\delta_3/2}}{\sqrt{n}}\frac{\sigma}{\mu}\right] \ge 1-\delta_3 \tag{29}$$

Suppose the sample size of the final sample satisfies

$$n \ge \left(\frac{z_{1-\delta_3/2}}{e}\frac{U_\sigma(\hat{\sigma},n_1,\delta_1)}{L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)}\right)^2 \tag{30}$$

Then, we have

$$\sigma \le U_\sigma(\hat{\sigma},n_1,\delta_1) \quad \text{AND} \quad \mu \ge L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)$$
$$\text{AND} \quad \left|\frac{\bar{Y}-\mu}{\mu}\right| \le \frac{z_{1-\delta_3/2}}{\sqrt{n}}\frac{\sigma}{\mu} \tag{31}$$
$$\Rightarrow$$
$$\left|\frac{\bar{Y}-\mu}{\mu}\right| \le z_{1-\delta_3/2}\left(\frac{z_{1-\delta_3/2}}{e}\frac{U_\sigma(\hat{\sigma},n_1,\delta_1)}{L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)}\right)^{-1}\frac{U_\sigma(\hat{\sigma},n_1,\delta_1)}{L_\mu^{(t)}(\bar{X},\hat{\sigma},n_1,\delta_2)}$$
$$= e$$

Namely, given a sample size satisfying Equation 30, if the relative error between $\bar{Y}$ and μ is larger than e, then one of the inequalities in 31 does not hold. Given the failure probability of inequalities in Equation 26, 27, and 29, we can derive the upper bound of the

failure probability of the relative error

$$\begin{aligned}
\mathbb{P}\left[\left|\frac{\bar{Y}-\mu}{\mu}\right| \geq e\right] &\leq \mathbb{P}\Bigg[(\sigma \geq U_\sigma(\hat{\sigma}, n_1, \delta_1)) \vee \left(\mu \leq L_\mu^{(t)}(\bar{X}, \hat{\sigma}, n_1, \delta_2)\right) \\
&\qquad \vee \left(\left|\frac{\bar{Y}-\mu}{\mu}\right| \geq \frac{z_{1-\delta_3/2}}{\sqrt{n}}\frac{\sigma}{\mu}\right)\Bigg] \\
&\leq \mathbb{P}\left[\sigma \geq U_\sigma(\hat{\sigma}, n_1, \delta_1)\right] + \mathbb{P}\left[\mu \leq L_\mu^{(t)}(\bar{X}, \hat{\sigma}, n_1, \delta_2)\right] \\
&\quad + \mathbb{P}\left[\left|\frac{\bar{Y}-\mu}{\mu}\right| \geq \frac{z_{1-\delta_3/2}}{\sqrt{n}}\frac{\sigma}{\mu}\right] \\
&= \delta_1 + \delta_2 + \delta_3
\end{aligned}$$

If we configure $\delta_1 + \delta_2 + \delta_3 \leq p$, we have

$$\mathbb{P}\left[\left|\frac{\bar{Y}-\mu}{\mu}\right| \geq e\right] < p$$

□

We observe that Theorem 5.4 and 5.5 give sufficient conditions with adjustable positive parameters: δ_1, δ_2, and δ_3. The only constraint for them is the sum being less than the total failure probability p. Different allocation strategies may result in different sample rates. However, we find that the difference in final performance is marginal for different allocations. Therefore, in this work, we split the failure probability evenly and leave the optimization of the allocation to future work.

5.4 Sample Rate Estimation for Cardinality

In this subsection, we present the theoretical result that solves Problem 2. Similarly, we describe the intuition and defer the detailed proof to our extended report [69].

We recall that to estimate a cardinality-related aggregate (e.g., COUNT), we draw a Bernoulli sample with success probability θ (i.e., sample rate) and count the number of successes n (i.e., smaple size). The cardinality of the table N is estimated by n/θ. We observe that n follows a binomial distribution $B(N, \theta)$. Therefore, we can exploit the confidence interval of $N \cdot \theta$ to obtain an upper bound of the relative error between n/θ and N. Since binomial distribution is a discrete distribution whose percentile is computationally complex, we use a normal distribution $\mathcal{N}(N\cdot\theta, N\cdot\theta\cdot(1-\theta))$ to approximate, as widely used in database and statistics literature [28, 52]. Based on that, we can derive the upper bound of the relative error in Equation 32, where δ_1 is an adjustable failure probability, and $z_{1-\delta_1}$ is the $1-\delta_1$ percentile of the standard normal distribution:

$$\mathbb{P}\left[\left|\frac{n/\theta - N}{N}\right| \leq \frac{z_{1-\delta_1}}{\sqrt{N}}\sqrt{\frac{1}{\theta}-1}\right] \geq 1-\delta_1 \tag{32}$$

According to Equation 32, we need to estimate the lower bound of N to obtain the upper bound of the relative error. Similar to Theorem 5.5, the lower bound of N can be estimated with a pilot sample. Theorem 5.6 presents the sufficient condition for θ to guarantee the relative error between n/θ and N. As discussed in Section 5.3, we allocate the overall failure probability p evenly to δ_1 and δ_2.

Theorem 5.6. *Consider two sequences of Bernoulli trials, each with an unknown number of trials N. The first sequence (i.e., pilot sample) has a success probability θ_1 and results in n_1 successes. The second sequence has a success probability θ and results in n successes. To guarantee that the relative error between n/θ and N is less than e with a maximum failure probability p, it is sufficient to ensure that θ satisfies*

$$\exists_{\delta_1,\delta_2>0}:\ \delta_1+\delta_2 \leq p,\ \theta \geq \left(1+\frac{e^2\cdot L_N(n_1,\theta_1,\delta_1)}{z^2_{1-\delta_2/2}}\right)^{-1}$$

where L_N is the lower bound of N given by Equation 25, and z_k is the k percentile of the standard normal distribution.

Proof. Similar to the proof of Theorem 5.5, we obtain a lower bound of the N by applying Lemma 5.3 on the first sequence of Bernoulli trials.

$$\begin{aligned}
&\mathbb{P}\left[n_1 \leq N\theta_1 + z_{1-\delta_1}\sqrt{N\theta_1(1-\theta_1)}\right] = 1-\delta_1 \\
\Leftrightarrow &\mathbb{P}\left[N \geq L_N(n_1,\theta_1,\delta_1)\right] = 1-\delta_1
\end{aligned} \tag{33}$$

where

$$L_N(n,\theta,\delta) = \left(\sqrt{\frac{n}{\theta}+z^2_{1-\delta}\frac{1-\theta}{2\theta}} - \sqrt{z^2_{1-\delta}\frac{1-\theta}{2\theta}}\right)^2$$

Nest, we apply the probabilistic bounds derived in Lemma 5.3 on the second sequence of Bernoulli trails, resulting in the relative error bounds of estimating N.

$$\begin{aligned}
&\mathbb{P}\left[N\theta - z_{1-\delta_2/2}\sqrt{N\theta(1-\theta)} \leq n \leq N\theta + z_{1-\delta_2/2}\sqrt{N\theta(1-\theta)}\right] \\
&= 1-\delta_2 \\
\Leftrightarrow &\mathbb{P}\left[\left|\frac{n}{\theta}-N\right| \leq z_{1-\delta_2/2}\sqrt{\frac{N(1-\theta)}{\theta}}\right] = 1-\delta_2 \\
\Leftrightarrow &\mathbb{P}\left[\left|\frac{n/\theta-N}{N}\right| \leq \frac{z_{1-\delta_2/2}}{\sqrt{N}}\sqrt{\frac{1}{\theta}-1}\right] = 1-\delta_2
\end{aligned} \tag{34}$$

Suppose the success probability of the second sequence of Bernoulli trials satisfies

$$\theta \geq \left(1+\frac{e^2\cdot L_N(n_1,\theta_1,\delta_1)}{z_{1-\delta_2/2}}\right)^{-1} \tag{35}$$

We have

$$\begin{aligned}
&N \geq L_N(n_1,\theta_1,\delta_1) \quad \text{AND} \quad \left|\frac{n/\theta-N}{N}\right| \leq \frac{z_{1-\delta_2/2}}{\sqrt{N}}\sqrt{\frac{1}{\theta}-1} \\
\Rightarrow &\left|\frac{n/\theta-N}{N}\right| \leq \frac{z_{1-\delta_2/2}}{\sqrt{L_N(n_1,\theta_1,\delta_1)}}\sqrt{1+\frac{e^2\cdot L_N(n_1,\theta_1,\delta_1)}{z_{1-\delta_2/2}}-1} = e
\end{aligned} \tag{36}$$

Namely, given a sample rate satisfying 35, if the estimation of the number of Bernoulli trials has a relative error larger than e, then one of the inequalities in 36 does not hold. Given the failure probability of those inequalities in 33 and 34, we can derive the upper bound

$$\begin{aligned} \textit{Table}\quad T &::= \ldots \mid T\ \texttt{TABLESAMPLE}(\theta) \\ \textit{Column}\ col &::= \ldots \mid \texttt{PAGELOC}(T) \end{aligned}$$

Figure 4: Additional syntax for rewritten queries.

$$(1)\ \frac{T \text{ is an input table} \quad T \text{ is largest in } Q_{in} \quad \theta \text{ is the pilot sample rate}}{T\ \texttt{TABLESAMPLE}(\theta) \in Q_{pilot}}$$

$$(2)\ \frac{T \text{ is an input table} \quad T \text{ is largest in } Q_{in}}{\texttt{PAGELOC}(T) \in \texttt{GroupingCols}(Q_{pilot})}$$

$$(3)\ \frac{(t_1\ aop\ t_2) \in Q_{in}}{[t_1, t_2] \in Q_{pilot}} \qquad (4)\ \frac{\texttt{AVG}(col) \in Q_{in}}{[\texttt{SUM}(col), \texttt{COUNT}(col)] \in Q_{pilot}}$$

Figure 5: Inference rules for PilotRewriter

of the overall failure probability.

$$\begin{aligned} \mathbb{P}\left[\left|\frac{n/\theta - N}{N}\right| > e\right] &\le \mathbb{P}\left[\left(N \le L_N(n_1, \theta_1, \delta_1)\right)\right. \\ &\qquad \left. \vee \left(\left|\frac{n/\theta - N}{N}\right| \ge \frac{z_{1-\delta_2/2}}{\sqrt{N}}\sqrt{\frac{1}{\theta} - 1}\right)\right] \\ &\le \mathbb{P}\left[N \le L_N(n_1, \theta_1, \delta_1)\right] + \mathbb{P}\left[\left|\frac{n/\theta - N}{N}\right| \ge \frac{z_{1-\delta_2/2}}{\sqrt{N}}\sqrt{\frac{1}{\theta} - 1}\right] \\ &= \delta_1 + \delta_2 \end{aligned}$$

If we configure $\delta_1 + \delta_2 \le p$, we have

$$\mathbb{P}\left[\left|\frac{n/\theta - N}{N}\right| > e\right] \le p$$

□

6 QUERY REWRITING

In this section, we introduce the query rewriting method that realizes block sampling and TPS introduced in Section 4 and 5. As mentioned in Algorithm 1, our query rewriter has two tasks: (1) transforming the input query Q_{in} into a pilot query Q_{pilot} that executes block sampling and calculates the page-oriented statistics for error analysis; (2) transforming Q_{in} into a final query Q_{final} that executes block sampling with the calculated sample rate and estimated the original aggregates. We describe each in turn.

Pilot Query Rewriting. The pilot query Q_{pilot} calculates aggregates for each sampled page, which are used to calculate the required sample rate later. To do that, we first extend the syntax of the table and column for the query rewriting in Figure 4. Specifically, we include the `TABLESAMPLE` clause as a table expression and include the location of the physical page as a column expression called `PAGELOC`. `PAGELOC` can be expressed differently in different DBMSs. For example, in DuckDB, we divide the row ID by the page size, while we use the system column `ctid` in PostgreSQL. Based on the syntax in Figure 2 and 4, we show the rewriting rules that make changes to Q_{in} using the rules of inference in Figure 5, represented in the standard Modus ponens form. We omit other rules that directly copy Q_{in} to Q_{pilot} for simplicity. We explain each rule as follows.

(1) We add a `TABLESAMPLE` clause to the largest table in Q_{in}.

$$(1)\ \frac{T \text{ is an input table} \quad T \text{ is largest in } Q_{in} \quad \theta \text{ is the final sample rate}}{T\ \texttt{TABLESAMPLE}(\theta) \in Q_{final}}$$

$$(2)\ \frac{\texttt{SUM}(col) \in Q_{in} \quad \theta \text{ is the sample rate}}{\texttt{SUM}(col)/\theta \in Q_{final}}$$

$$(3)\ \frac{\texttt{COUNT}(col) \in Q_{in} \quad \theta \text{ is the sample rate}}{\texttt{COUNT}(col)/\theta \in Q_{final}}$$

Figure 6: Inference rules for FinalRewriter

Table 3: Characteristics of workloads.

Workload	#Queries	#Queries w/ joins	Max/Avg. #groups
TPC-H	9	7	175/22
ClickBench	7	0	17/3
Star Schema	10	10	150/38
Instacart	9	7	146/22

(2) We incorporate the page ID column of the largest table into `GROUP BY` clauses to compute page-oriented estimators later.
(3) We decompose composite aggregates (e.g., `SUM(x)/SUM(y)`) into separate components to analyze errors.
(4) We replace `AVG` with `SUM` and `COUNT` to calculate the page-oriented estimator for `AVG` later.

Final Query Rewriting. The final query Q_{final} estimates the aggregates in the input query Q_{in} with block sampling. We present the rewriting rules that make changes to Q_{in} in Figure 5, while omitting other rules that directly copy Q_{in} to Q_{pilot}. We explain each rule as follows. Similar to the PilotRewriter, we add a `TABLESAMPLE` clause to the largest table. In addition, we divide `SUM` and `COUNT` by the sample rate θ, considering the down-scaling effect of sampling.

7 EVALUATION

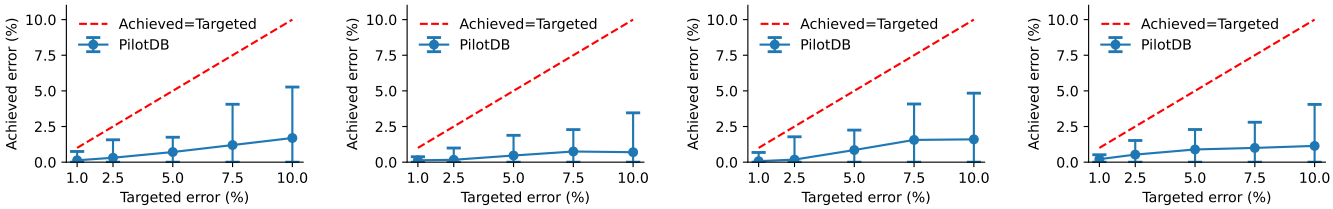
In this section, we evaluate PilotDB with extensive experiments. We aim to answer the following research questions.

- **Correctness Analysis**. Does PilotDB achieve the error guarantees on the results?
- **Performance Gains**. How many speedups can PilotDB achieve on different DBMSs and errors?
- **Ablation Study**. How do components in PilotDB contribute to the final performance?
- **Sensitivity Study**. How does the performance of PilotDB change in different settings?

7.1 Experiment Settings

Workloads. We evaluate PilotDB on four widely used workloads.

- **TPC-H** is a standard workload that simulates ad-hoc queries for decision-making [14]. We use a scale factor of 1,000.
- **ClickBench** simulates ad-hoc analysis over machine-generated data, such as structured logs, with one table [12]. We scale the data by 5×, resulting in a pre-processed size of 200GB.
- **Star Schema Benchmark (SSB)** is a standard workload for star schema data warehouse queries [49]. We use 1,000 scale factor.



(a) TPC-H. (b) ClickBench. (c) SSB. (d) Instacart.

Figure 7: PilotDB achieves the targeted errors for all workloads.

- **Instacart** is a micro-benchmark with real-world data from the Instacart [31] and queries from TPC-H. We scale up the original data by 100× using the same method as VerdictDB [52].

For each benchmark, we evaluate aggregation queries that have non-empty results, no correlated subqueries, and fewer than 200 groups. Queries with more than 200 groups typically contain small-sized groups, which can lead to missing groups. Such queries are not suitable for middleware AQP and are not supported in prior work with offline samples [52]. As summarized in Table 3, we evaluated 35 queries in total, including queries with joins and various numbers of groups.

DBMSs. We evaluate PilotDB on three DBMSs: PostgreSQL 16.3, SQL Server 2022, and DuckDB 1.0.0. DuckDB is an open-source column-oriented DBMS designed for online analytical processing, known for its efficiency in processing analytical queries that fit in memory [37]. The default sampler of DuckDB scans the entire column before block sampling, which is not efficient. We implemented a 30-line extension that realizes block sampling without full scans.[1]

Metrics. We consider two types of metrics: performance metrics and error metrics. For performance metrics, we measure the speedup factor, calculated as the ratio of the runtime of the exact query to that of PilotDB. For error metrics, we consider both error events introduced in Section :

- **Group error**: The number of missed groups that have sizes larger than the requested value g (i.e., minimum group size).
- **Aggregate error**: The maximum relative error of all aggregates in a query.

Testbed. Our experiments are conducted on CloudLab r6525 nodes, each equipped with 256GB RAM, 1.6TB NVMe SSD, and two 32-core AMD 7543 CPUs. The nodes run Ubuntu 22.04 and Python 3.11. Before executing each query, we clear both the operating system cache and the query plan cache.

7.2 PilotDB Guarantees Errors

We first analyze whether PilotDB achieves guaranteed errors.

Setup. We evaluated the error guarantee property of PilotDB on all workloads. We executed each query on PostgreSQL with various targeted errors 10 times and measured the achieved errors for all queries. For each execution, we set the failure probability to 5% and the minimum group size to 400 pages.

[1]We are working on merging our changes into DuckDB [68].

Table 4: PilotDB accelerates over 80% of queries and achieves speedups by up to two orders of magnitude.

	PostgreSQL	SQL Server	DuckDB
Query coverage	80.0%	82.9%	82.9%
G.M. speedup	5.90×	5.63×	1.84×
Max. speedup	126×	56.5×	13.2×

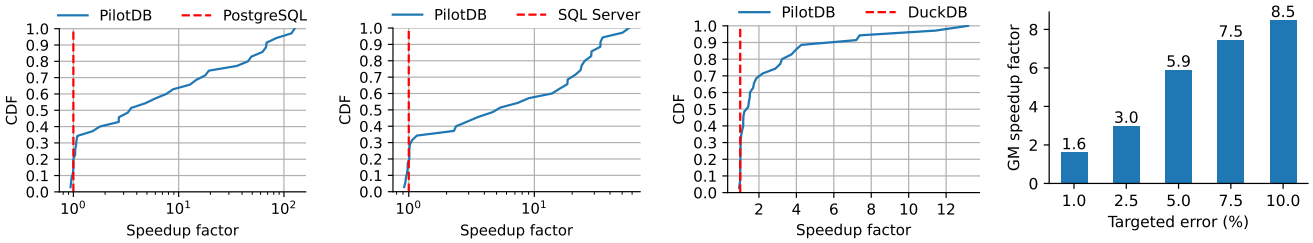
Results. In terms of aggregate error, PilotDB never miss groups across all the queries we evaluated, resulting in a group error probability of 0%. In terms of aggregate error, Figure 7 shows the achieved error for each workload with various targeted errors. The bars in the figure represent the minimum and maximum achieved errors across all queries and executions, while the dots indicate the average achieved errors. As we can see, the achieved errors of PilotDB are consistently less than the targeted errors.

Discussion. We observe that PilotDB conservatively guarantees errors, where the maximum of achieved errors is approximately half of the targeted error. This occurs because the sample rate that PilotDB calculates is a sufficient condition for the requested error. The sample rate may not be necessary because of several relaxations in the calculation. For example, we applied Boole's Inequality to simplify the probability of a union of events to the sum of the probabilities of individual events. Boole's Inequality can be loose when the events are positively correlated. To ensure the sample rate is sufficient and necessary, it is important to analyze the correlations between events, which could be a valuable direction for future research.

7.3 PilotDB Accelerates Query Execution

In this section, we analyze the performance of PilotDB by evaluating it on different DBMSs and different targeted aggregate errors.

Setup. We first execute PilotDB on different DBMSs, targeting 5% aggregate error, 400 minimum group size, and 5% failure probability. Next, we execute PilotDB with different targeted aggregate errors on PostgreSQL with 400 minimum group size and 5% failure probability. The performance of PilotDB is compared with directly executing queries on the corresponding DBMS. We execute each query in each setting five times and calculate the arithmetic mean of speedups for each query. The overall speedup on each DBMS is calculated as the geometric mean of speedups of all queries.



(a) PostgreSQL (log scale). (b) SQL Server (log scale). (c) DuckDB.

Figure 8: PilotDB achieves 0.92~126× speedups across three DBMSs.

Figure 9: Speedups of PilotDB across various errors.

PilotDB Accelerates Queries on All DBMSs. Table 4 summarizes the performance of PilotDB on different DBMSs. As shown, PilotDB consistently accelerates 80% queries on each DBMS. Moreover, PilotDB achieves up to 126× speedups on transactional databases and up to 13× speedups on an analytical database, DuckDB. Figure 8 provides a detailed view of performance on each database, illustrating the cumulative probability function of speedups. For example, as shown in Figure 8b, PilotDB achieves at least 10× speedups for approximately 40% of queries. In the worst case, PilotDB slows down the execution by at most 8%.

PilotDB Accelerates Queries on Various Errors. Figure 9 shows the geometric mean of the speedups of PilotDB with various targeted errors. We observe that PilotDB achieves speedups for all evaluated targeted errors. Even with a small targeted error of 1%, PilotDB achieves 1.8× speedups on geometric mean. Furthermore, we find that PilotDB achieves higher speedups at larger targeted errors. This is because the required sample size decreases with larger targeted errors.

Discussion. We first discuss the two types of queries that PilotDB fails to accelerate in our evaluation:

(1) **Queries with aggregates of high variance:** According to our sample size formula, the required sample size for a requested error is proportional to the variance. Therefore, when the variance is large, PilotDB requires a high sample rate to guarantee the error.
(2) **Queries with low selectivity or small groups:** When the selectivity or group size of the query is small, PilotDB requires a high sample rate to achieve the required sample size for a guaranteed error.

In both cases, when the sample rate exceeds the pre-determined threshold, PilotDB falls back to executing the exact query, resulting in no speedup. One of the advantages of PilotDB is that for queries it fails to accelerate, the overhead is minimal, at most 6%, while still guaranteeing errors by executing the exact query.

Additionally, we observe that PilotDB performs relatively better on PostgreSQL and SQL Server compared to DuckDB. We believe there are two reasons for this:

(1) **Optimization for in-memory processing**: DuckDB is highly optimized for in-memory analytical processing. Since our workload can fit in memory after compression, the cost reduction by sampling is smaller compared to transactional databases.

Table 5: Geometric mean of slowdowns of PilotDB compared to PilotDB (Oracle).

	PostgreSQL	SQL Server	DuckDB
PilotDB (overall)	1.61×	1.21×	1.27×
PilotDB (2nd phase)	1.04×	1.08×	1.19×

(2) **Page size differences**: DuckDB has a default page size significantly larger than other DBMSs. DuckDB's page size is 2048, while PostgreSQL's and SQL Server's page sizes range from 40 to 50. A larger page size introduces stronger correlations in block sampling, requiring a larger sample size for the same requested error.

Finally, we observe that the speedup of PilotDB is sub-linear with respect to the targeted error rate. However, in our theoretical results, the required sample size is proportional to e^{-2}, where e is the targeted error rate. This is valid since the runtime is not necessarily linear with respect to the sample size, since sampling may introduce overheads in query processing, such as the generation of random numbers.

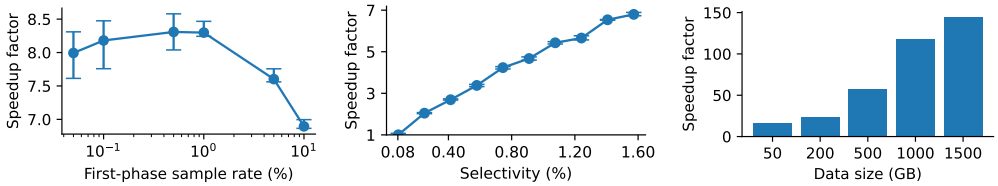
7.4 Ablation Study

In this section, we evaluate the effectiveness of the design choices of PilotDB by comparing speedups with alternative configurations.

(1) PilotDB (Oracle): We replace the first phase sampling of PilotDB with pre-determined statistics.
(2) PilotDB (Uniform): We replace the block sampling of PilotDB with uniform sampling.

We use the same setting as Section 7.3 to measure the performance.

PilotDB Achieves Near-Optimal Performance. In the first phase of PilotDB, we use interval estimations to calculate the sample rate for a requested error. To understand the impact of these estimations on the performance of PilotDB, we compare it with PilotDB (Oracle), which demonstrates the upper-bound performance achievable with online sampling and query rewriting for AQP with a priori error guarantees. For each query, we compare the overall runtime and the second phase runtime of PilotDB with the runtime of PilotDB. The execution of each query follows the same setting as described in Section 7.3. We evaluated all queries in our workloads.



(a) Varying 1st-phase sample rates. **(b) Varying the selectivity.** **(c) Varying data sizes.**

Figure 10: PilotDB achieves reasonable speedups on various settings.

Table 6: Speedups of PilotDB over PilotDB (Uniform).

	PostgreSQL	SQL Server	DuckDB
Geometric mean	12.6×	9.37×	1.92×
Maximum	219×	71.4×	13.2×

Table 5 summarizes the slowdown factor of PilotDB compared to PilotDB (Oracle), computed as the geometric mean of the ratio of runtimes. Compared to the overall runtime of PilotDB, PilotDB (Oracle) is 4%∼61% faster. Notably, the runtime of PilotDB (Oracle) does not include the time to calculate the sample rates, which requires executing the exact query, whereas the runtime of PilotDB includes the time for sample rate estimation (i.e., the first phase sampling) To decouple the factors that affect the runtime, we compare the second phase of PilotDB with PilotDB (Oracle). In this case, oracle improves the performance of PilotDB by 4%∼19%, demonstrating that the performance of PilotDB is close to optimal.

PilotDB Outperforms Uniform Sampling. In Section 4, we have shown the advantage of block sampling over uniform sampling with a motivating example in Figure 3. Here, we demonstrate the benefit of block sampling in terms of overall performance. We compare the performance of PilotDB and PilotDB (Uniform) across all the workloads. The execution of each query follows the same setting as described in Section 7.3. We implement uniform sampling by rewriting queries with "`TABLESAMPLE BERNOULLI(p)`" in PostgreSQL and DuckDB, and with "`WHERE rand_value > p`" in SQL Server, where `rand_value` is a random number in [0,1] calculated for each row.

Table 6 summarizes the speedups of PilotDB compared to PilotDB (Uniform). We show the geometric mean and maximum speedups for each DBMS. Overall, PilotDB with block sampling achieves greater performance by 8.0× on geometric mean and up to 219×, demonstrating that block sampling is significantly more efficient than uniform samplinge. We observe that block sampling shows a greater advantage on PostgreSQL and SQL Server compared to DuckDB. We believe this is because DuckDB only scans necessary columns in tables which naturally has smaller costs of data scans than PostgreSQL and SQL Server.

7.5 Sensitivity Study

In this section, we analyze the performance of PilotDB under various settings.

Impact of First-Phase Sample Rate. We analyze how the sample rate of the first phase of PilotDB affects its performance. Specifically, we execute TPC-H query 6 on SQL Server with various sample rates for the first phase, targeting a 1% error. The execution follows the same setting as described in Section 7.3. Figure 10a shows the speedups of PilotDB with various sample rates, where the error bars show the maximum and minimum values in 10 executions. We find that the performance of PilotDB is not monotonic with respect to the sample rate. The performance is worse at low sample rates because the estimation for the sample rate can be loose, while the performance is worse at high sample rates because the first-phase sampling becomes time-consuming. Nevertheless, we observe that PilotDB achieves significant speedups (> 6×) consistently across a wide range of sample rates.

Impact of Selectivity. We analyze how the selectivity of the query affects the performance of PilotDB. Specifically, we modify the predicate of TPC-H query 6 to represent different selectivities. We execute the modified queries on PostgreSQL with the same setting as in Section 7.3, targeting a 1% error. Figure 10b shows the speedups of PilotDB with various selectivities. As expected, PilotDB achieves better performance with higher selectivity. This is because we require smaller sample rates for queries with higher selectivity to achieve the same requested error.

Impact of Data Size. We analyze how the data size affects the performance of PilotDB. Specifically, we execute TPC-H query 6 on PostgreSQL with various scale factors. The execution follows the same setting as described in Section 7.3. Figure 10c shows the speedups of PilotDB with 50GB to 1,500GB data. We observe that speedups of PilotDB increase monotonically regarding the data size, demonstrating that PilotDB is suitable for big data analytics.

8 CONCLUSION

In this work, we propose PilotDB as a step toward AQP that achieves (1) a priori error guarantees, (2) minimal modifications or maintenance, and (3) significant cost reductions. We develop two novel techniques to achieve the goal. First, we leverage block sampling to efficiently process queries with error guarantees. Second, we propose a two-phase sampling scheme to realize online AQP with a priori error guarantees. Our evaluation over three workloads and three DBMSs reveals that PilotDB achieves error guarantees with up to 126× speedups compared to executing exact queries.

REFERENCES

[1] Swarup Acharya, Phillip B Gibbons, and Viswanath Poosala. 1999. Aqua: A fast decision support systems using approximate query answers. In *PVLDB*.

[2] Swarup Acharya, Phillip B Gibbons, and Viswanath Poosala. 2000. Congressional samples for approximate answering of group-by queries. In *SIGMOD*.

[3] Swarup Acharya, Phillip B Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999. The aqua approximate query answering system. In *SIGMOD*.

[4] Swarup Acharya, Phillip B Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. 1999. Join synopses for approximate query answering. In *SIGMOD*.

[5] 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 *EuroSys*.

[6] Brian Babcock, Surajit Chaudhuri, and Gautam Das. 2003. Dynamic sample selection for approximate query processing. In *SIGMOD*.

[7] Lukas Berg, Tobias Ziegler, Carsten Binnig, and Uwe Röhm. 2019. ProgressiveDB: progressive data analytics as a middleware. *PVLDB* (2019).

[8] Surajit Chaudhuri, Gautam Das, Mayur Datar, Rajeev Motwani, and Vivek Narasayya. 2001. Overcoming limitations of sampling for aggregation queries. In *ICDE*.

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

[10] Jiecao Chen and Qin Zhang. 2017. Bias-Aware Sketches. *PVLDB* (2017).

[11] Xingguang Chen, Fangyuan Zhang, and Sibo Wang. 2022. Efficient Approximate Algorithms for Empirical Variance with Hashed Block Sampling. In *KDD*.

[12] ClickHouse. [n.d.]. *ClickBench: a Benchmark For Analytical Databases.* https://github.com/ClickHouse/ClickBench Accessed: 2024-06-04.

[13] Google Cloud. [n.d.]. *Table sampling | BigQuery | Google Cloud.* https://cloud.google.com/bigquery/docs/table-sampling Accessed: 2024-05-12.

[14] Transaction Processing Performance Council. [n.d.]. *TPC-H Homepage.* https://www.tpc.org/tpch/ Accessed: 2024-06-04.

[15] Sanjoy Dasgupta and Anupam Gupta. 2003. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms* (2003).

[16] Databricks. [n.d.]. *TABLESAMPLE clause | Databricks on AWS.* https://docs.databricks.com/en/sql/language-manual/sql-ref-syntax-qry-select-sampling.html Accessed: 2024-05-12.

[17] Azure Databricks. 2024. *TABLESAMPLE clause.* https://learn.microsoft.com/en-us/azure/databricks/sql/language-manual/sql-ref-syntax-qry-select-sampling Accessed: 2024-05-12.

[18] Bolin Ding, Silu Huang, Surajit Chaudhuri, Kaushik Chakrabarti, and Chi Wang. 2016. Sample+ seek: Approximating aggregates with distribution precision guarantee. In *SIGMOD*.

[19] Alin Dobra, Chris Jermaine, Florin Rusu, and Fei Xu. 2009. Turbo-charging estimate convergence in DBO. *PVLDB* (2009).

[20] DuckDB. [n.d.]. *Samples - DuckDB.* https://duckdb.org/docs/sql/samples.html Accessed: 2024-05-12.

[21] Alex Galakatos, Andrew Crotty, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2017. Revisiting reuse for approximate query processing. *PVLDB* (2017).

[22] Venkatesh Ganti, Mong-Li Lee, and Raghu Ramakrishnan. 2000. Icicles: Self-tuning samples for approximate query answering. In *PVLDB*.

[23] Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. 2002. Querying and Mining Data Streams: You Only Get One Look. (2002).

[24] Sudipto Guha and Boulos Harb. 2005. Wavelet synopsis for data streams: minimizing non-euclidean error. In *KDD*.

[25] Peter J Haas and Joseph M Hellerstein. 1999. Ripple joins for online aggregation. *ACM SIGMOD Record* (1999).

[26] Joseph M Hellerstein, Peter J Haas, and Helen J Wang. 1997. Online aggregation. In *SIGMOD*.

[27] Apache Hive. [n.d.]. *LanguageManual Sampling - Apache Hive - Apache Software Foundation.* https://cwiki.apache.org/confluence/display/hive/languagemanual+sampling Accessed: 2024-05-12.

[28] Robert V Hogg, Joseph W McKean, and Allen T Craig. 2019. *Introduction to mathematical statistics.* Pearson.

[29] Wen-Chi Hou, Gultekin Ozsoyoglu, and Baldeo K Taneja. 1988. Statistical estimators for relational algebra expressions. In *SIGMOD*.

[30] Apache Impala. [n.d.]. *TABLESAMPLE Clause.* https://impala.apache.org/docs/build/html/topics/impala_tablesample.html Accessed: 2024-05-12.

[31] Instacart. [n.d.]. *Instacart Market Basket Analysis.* https://www.kaggle.com/c/instacart-market-basket-analysis/data Accessed: 2024-05-12.

[32] Chris Jermaine, Subramanian Arumugam, Abhijit Pol, and Alin Dobra. 2008. Scalable approximate query processing with the DBO engine. *ACM Transactions on Database Systems* (2008).

[33] Clemens Jochum, Peter Jochum, and Bruce R Kowalski. 1981. Error propagation and optimal performance in multicomponent analysis. *Analytical Chemistry* 53, 1 (1981), 85–92.

[34] Ramesh Johari, Pete Koomen, Leonid Pekelis, and David Walsh. 2017. Peeking at a/b tests: Why it matters, and what to do about it. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1517–1525.

[35] Srikanth Kandula, Kukjin Lee, Surajit Chaudhuri, and Marc Friedman. 2019. Experiences with approximating queries in Microsoft's production big-data clusters. *Proceedings of the VLDB Endowment* 12, 12 (2019), 2131–2142.

[36] Srikanth Kandula, Anil Shanbhag, Aleksandar Vitorovic, Matthaios Olma, Robert Grandl, Surajit Chaudhuri, and Bolin Ding. 2016. Quickr: Lazily approximating complex adhoc queries in bigdata clusters. In *SIGMOD*.

[37] DuckDB Labs. 2024. *DuckDB 1.0.0.* https://duckdb.org/docs/installation

[38] Database Languages. 2003. SQL, ISO/IEC 9075*:2003.

[39] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. 2016. Wander join: Online aggregation via random walks. In *SIGMOD*.

[40] Kaiyu Li and Guoliang Li. 2018. Approximate query processing: What is new and where to go? a survey on approximate query processing. *Data Science and Engineering* 3, 4 (2018), 379–397.

[41] Kaiyu Li, Yong Zhang, Guoliang Li, Wenbo Tao, and Ying Yan. 2018. Bounded approximate query processing. *TKDE* (2018).

[42] Microsoft. 2024. *SQL Server 2022 CU13.* https://packages.microsoft.com/ubuntu/20.04/mssql-server-2022/pool/main/m/mssql-server/

[43] Barzan Mozafari. 2017. Approximate query engines: Commercial challenges and research opportunities. In *SIGMOD*.

[44] Barzan Mozafari, Radu Alexandru Burcuta, Alan Cabrera, Andrei Constantin, Derek Francis, David Grömling, Alekh Jindal, Maciej Konkolowicz, Valentin Marian Spac, Yongjoo Park, et al. 2023. Making Data Clouds Smarter at Keebo: Automated Warehouse Optimization using Data Learning. In *Companion of the 2023 International Conference on Management of Data*. 239–251.

[45] Ioannis Mytilinis, Dimitrios Tsoumakos, and Nectarios Koziris. 2016. Distributed wavelet thresholding for maximum error metrics. In *Proceedings of the 2016 International Conference on Management of Data*. 663–677.

[46] Supriya Nirkhiwale, Alin Dobra, and Christopher Jermaine. 2013. A Sampling Algebra for Aggregate Estimation. *PVLDB* (2013).

[47] Frank Olken and Doron Rotem. 1986. Simple Random Sampling from Relational Databases. In *PVLDB*.

[48] Matthaios Olma, Odysseas Papapetrou, Raja Appuswamy, and Anastasia Ailamaki. 2019. Taster: Self-tuning, elastic and online approximate query processing. In *ICDE*. IEEE.

[49] Patrick E O'Neil, Elizabeth J O'Neil, and Xuedong Chen. 2007. The star schema benchmark (SSB). *Pat* 200, 0 (2007), 50.

[50] M Palmer. 2003. Propagation of uncertainty through mathematical operations. *Massachusetts Institute of* (2003).

[51] Prashant Pandey, Michael A Bender, Rob Johnson, and Rob Patro. 2017. A general-purpose counting filter: Making every bit count. In *SIGMOD*.

[52] Yongjoo Park, Barzan Mozafari, Joseph Sorenson, and Junhao Wang. 2018. Verdictdb: Universalizing approximate query processing. In *SIGMOD*.

[53] Gregory Piatetsky-Shapiro and Charles Connell. 1984. Accurate estimation of the number of tuples satisfying a condition. *ACM Sigmod Record* (1984).

[54] Abhijit Pol and Christopher Jermaine. 2005. Relational confidence bounds are easy with the bootstrap. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 587–598.

[55] Viswanath Poosala, Peter J Haas, Yannis E Ioannidis, and Eugene J Shekita. 1996. Improved histograms for selectivity estimation of range predicates. *ACM SIGMOD Record* (1996).

[56] PostgreSQL. [n.d.]. *TABLESAMPLE Implementation - PostgreSQL wiki.* https://wiki.postgresql.org/wiki/TABLESAMPLE_Implementation Accessed: 2024-05-12.

[57] Kexin Rong, Yao Lu, Peter Bailis, Srikanth Kandula, and Philip Levis. 2020. Approximate partition selection for big-data workloads using summary statistics. *PVLDB* (2020).

[58] Michael Shekelyan, Anton Dignös, and Johann Gamper. 2017. Digithist: a histogram-based data summary with tight error bounds. *PVLDB* (2017).

[59] Nikhil Sheoran. 2022. Deepola: Online aggregation for deeply nested queries. In *SIGMOD*.

[60] Talia Tamarin-Brodsky and John Marshall. [n.d.]. Error Analysis. ([n. d.]).

[61] PostgreSQL Team. 2024. *PostgreSQL 16.3.* The PostgreSQL Global Development Group.

[62] Sai Wu, Beng Chin Ooi, and Kian-Lee Tan. 2010. Continuous sampling for online aggregation over multiple queries. In *SIGMOD*.

[63] Fei Xu, Christopher Jermaine, and Alin Dobra. 2008. Confidence bounds for sampling-based group by estimates. *ACM Transactions on Database Systems* (2008).

[64] Ying Yan, Liang Jeff Chen, and Zheng Zhang. 2014. Error-bounded sampling for analytics on big sparse data. *Proceedings of the VLDB Endowment* 7, 13 (2014), 1508–1519.

[65] Kai Zeng, Sameer Agarwal, Ankur Dave, Michael Armbrust, and Ion Stoica. 2015. G-ola: Generalized on-line aggregation for interactive analysis on big data. In *SIGMOD*.

[66] Kai Zeng, Shi Gao, Jiaqi Gu, Barzan Mozafari, and Carlo Zaniolo. 2014. ABS: a system for scalable approximate queries with accuracy guarantees. In *SIGMOD*.

[67] Kai Zeng, Shi Gao, Barzan Mozafari, and Carlo Zaniolo. 2014. The analytical bootstrap: a new method for fast error estimation in approximate query processing. In *SIGMOD*. 277–288.
[68] Chengsong Zhang. [n.d.]. *Improvement the speed of table sample systems #12631*. https://github.com/duckdb/duckdb/pull/12631 Accessed: 2024-06-21.
[69] Yuxuan Zhu and Daniel Kang. 2024. *PilotDB: A Step Toward Minimalist Approximate Query Processing with A Priori Error Guarantees (Extended Version)*. https://github.com/uiuc-kang-lab/PilotDB/blob/vldb25/extended_report.pdf

A TABLE FOR TABLESAMPLE CLAUSE AND PAGE ID

We show examples of TABLESAMPLE clauses and page id related columns for several databases in Table 7.

Database	TABLESAMPLE Clause	Physical Location
PostgreSQL	TABLESAMPLE SYSTEM (0.05)	table.ctid
SQL Server	TABLESAMPLE (0.05 PERCENT)	table.%%physloc%%
DuckDB	TABLESAMPLE SYSTEM (0.05%)	table.rowid

Table 7: TABLESAMPLE Clause Specifications and Page ID Related Columns in PostgreSQL, SQL Server, and DuckDB.

B EXAMPLE QUERY REWRITING

In this section, we show an example of our query rewriter for PostgreSQL using the following input query:

```
SELECT l_returnflag, AVG(l_quantity) as avg_qty
    FROM lineitem
    GROUP BY l_returnflag;
```

The rewritten pilot query will be:

```
SELECT l_returnflag,
       SUM(l_quantity),
       COUNT(l_quantity),
       'page_id_0:' || CAST(
       (lineitem.ctid::text::point)[0] AS INT
       ) AS page_id_0
    FROM lineitem TABLESAMPLE SYSTEM (0.05)
    GROUP BY l_returnflag, page_id_0;
```

And the final sample query will be:

```
SELECT l_returnflag, AVG(l_quantity) as avg_qty
    FROM lineitem TABLESAMPLE SYSTEM (rate)
    GROUP BY l_returnflag;
```

The pilot query substitutes AVG estimator with SUM and COUNT estimator. By executing this pilot query, we can get page statistics, which is used to estimate the required sample size for the final query. The 'rate' specified in the final query represents this estimated sample size. We can get the AQP results by executing the final query.