# The Analytical Bootstrap: a New Method for Fast Error Estimation in Approximate Query Processing

Kai Zeng[†] Shi Gao[†] Barzan Mozafari[‡] Carlo Zaniolo[†]
[†]University of California, Los Angeles [‡]University of Michigan
[†]{kzeng, gaoshi, zaniolo}@cs.ucla.edu [‡]mozafari@umich.edu

**Abstract**

Sampling is one of the most commonly used techniques in Approximate Query Processing (AQP)—an area of research that is now made more critical by the need for timely and cost-effective analytics over "Big Data". Assessing the quality (i.e., estimating the error) of approximate answers is essential for meaningful AQP, and the two main approaches used in the past to address this problem are based on either (i) analytic error quantification or (ii) the bootstrap method. The first approach is extremely efficient but lacks generality, whereas the second is quite general but suffers from its high computational overhead. In this paper, we introduce a probabilistic relational model for the bootstrap process, along with rigorous semantics and a unified error model, which bridges the gap between these two traditional approaches. Based on our probabilistic framework, we develop efficient algorithms to predict the error distribution of the approximation results. These enable the computation of any bootstrap-based quality measure for a large class of SQL queries via a single-round evaluation of a slightly modified query. Extensive experiments on both synthetic and real-world datasets show that our method has superior prediction accuracy for bootstrap-based quality measures, and is several orders of magnitude faster than bootstrap.

## 1 Introduction

Data-driven activities in business, science and engineering are rapidly growing in terms of both significance and the size of the data they use. This situation has brought even more attention to the already-active area of Approximate Query Processing (AQP), and in particular to sampling approaches as a critical and general technique for coping with the ever-growing size of big data. Sampling techniques are widely used in databases [2, 6, 11, 18, 20, 28], stream processors [7, 27], and even Map-Reduce systems [4, 26].

This most commonly used technique consists in evaluating the queries on a small random *sample* of the original database. Of course, the approximate query answers so obtained are of very limited utility unless they are accompanied by some *accuracy guarantees*. For instance, in estimating income from a small sample of the population, a statistician seeks assurance that the answer so derived falls within a given margin of the correct answer computed on the whole population, with high confidence, say within $\pm 1\%$ of the correct answer with probability $\geq 95\%$. This enables the user to decide whether the current approximation is "good enough" for his/her purpose. Thus, assessing the quality (i.e., error estimation) of approximate answers is a fundamental aspect of AQP.

The past two decades have seen much work on error estimation, which can be categorized into two main approaches. The first approach [4, 10, 11, 18, 19, 20, 21, 29, 40] uses asymptotic theory to analytically derive closed-form error estimates for common aggregate functions in a database, such as `SUM`, `AVG`, etc. Although computationally appealing, analytic error quantification is restricted to a very limited set of queries. Thus, for every new type of query, a new closed-form formula must be derived. This derivation is a manual process that is ad-hoc and often impractical for complex queries[1] [30].

To address this problem, a second approach, called *bootstrap*, has emerged as a more general method for routine estimation of errors [23, 26, 30]. Bootstrap [14] is essentially a Monte-Carlo procedure, which for a given initial sample, (i) repeatedly forms simulated datasets by resampling tuples i.i.d. (identically and independently) from the given sample, (ii) recomputes the query on each of the simulated datasets, and (iii) assesses the quality of answer on the basis of the empirical distribution of the query answers so produced. The wide applicability and automaticity of bootstrap is confirmed both in theory [8, 37] and practice [23, 26, 30]. Unfortunately, bootstrap tends to suffer from its high computational overhead, since hundreds, or even thousands of bootstrap trials are typically needed to obtain reliable estimates [23, 30].

In this paper, we introduce a new technique, called *Analytical Bootstrap Method* (ABM), which is both computationally efficient and automatically applicable to a large class of SQL queries, and thus combines the benefits of these two approaches. Thus, this paper's main contribution is *a probabilistic relational model for the bootstrap process that allows for automatic error quantification of a large class of SQL queries (defined in Section 2.2) under sampling, but without performing the actual Monte Carlo simulation.* We show that our error estimates are provably equivalent to those produced by the simulation-based bootstrap (Theorems 1 and 2).

The basic idea of ABM is to annotate each tuple of the database with an integer-valued random variable that represents the possible multiplicity with which this tuple would appear in the simulated datasets generated by bootstrap. The small annotated database so

[1]This is evidenced by the difficulties faced by previous analytic approaches in supporting approximation for queries that are more complex than simple group-by-aggregates.

obtained from the initial sample is called a *probabilistic multiset database* (PMDB for short). This PMDB succinctly models all possible simulated datasets that could be generated by bootstrap trials. Then, we extend relational operators to manipulate these random variables to ensure that the execution of the query on the PMDB will generate an annotated relation which encodes the distribution of all possible answers that would be produced *if* we actually performed bootstrap on the database. In particular, using a single-round query evaluation, ABM accurately estimates the empirical distribution of the query answers that would be produced by hundreds or thousands of bootstrap trials on the sampled database[2].

We have evaluated ABM through extensive experiments on the TPC-H benchmark and on the actual queries and datasets used by leading customers of a major vendor of enterprise databases. Our results show that ABM is an accurate prediction of the simulation-based bootstrap, while it is 3–4 orders of magnitude faster than the state-of-the-art parallel implementations of bootstrap [24].

Therefore, ABM promises to be a technique of considerable practical significance: The immediate implication of this new technique is that, the quality assessment module of any AQP system that currently relies on bootstrap (e.g., [23, 26, 30]) can now be replaced by a process that uses 3–4 orders of magnitude fewer resources (resp. lower latency) if they currently use a parallel (resp. sequential) implementation of bootstrap. We envision that by removing bootstrap's computational overhead, ABM would also significantly broaden the application of AQP to areas which require *interactive* and *complex* analytics that are expressible in SQL, such as root cause analysis and A/B testing [4], real-time data mining, and exploratory data analytics.

The paper is organized as follows. Section 2 provides a small example to explain bootstrap and the query evaluation problem considered in this paper. Section 3 provides the necessary theoretical background. In Section 4, we present our probabilistic multiset relational model. We explain our efficient exact and approximate query evaluation algorithms respectively in Sections 5 and 6. In Section 7, we generalize our techniques using several extensions. Finally, we report our experiments in Section 8, followed by the related work in Section 9. We conclude in Section 10.

# 2 Problem Statement

In this section, we formally state the problem addressed by this paper. We then provide a small example of bootstrap in Section 2.1, and a high-level overview of our approach in Section 2.2.

Given a database $\mathfrak{D}$ and a query $\mathfrak{q}$, let $\mathfrak{q}(\mathfrak{D})$ denote the exact answer of evaluating $\mathfrak{q}$ on $\mathfrak{D}$. An approximate answer can be obtained by (i) extracting from $\mathfrak{D}$ a random sample $D$, (ii) evaluating a potentially modified version of $\mathfrak{q}$ (say $q$) on $D$, and (iii) using $q(D)$ as an approximation of $\mathfrak{q}(\mathfrak{D})$. In some cases, such as `AVG`, $q$ is the same as $\mathfrak{q}$, but in other cases, $q$ can be a modified query that produces *better* results. For instance, when evaluating `SUM` on a sample of size $f \times |\mathfrak{D}|$, one will choose $q = \frac{1}{f}\mathfrak{q}$ to scale up the sample sum by a factor of $\frac{1}{f}$. The particular selection of $q$ for a given $\mathfrak{q}$ is outside the scope of this paper[3]. Instead, we focus on estimating the quality of $q(D)$ for a given $q$, as described next.

Let $D_1, \cdots, D_N$ be all possible sample instantiations of $\mathfrak{D}$; thus $q(D)$ could be any of the $\{q(D_i), i = 1, \cdots, N\}$ values. Therefore, to assess the quality (e.g. accuracy) of $q(D)$, one needs to (conceptually) consider all possible query answers $\{q(D_i)\}$, and then compute some user-specified measure of quality, denoted by $\xi(\mathfrak{q}(\mathfrak{D}), \{q(D_i)\})$. For instance, when approximating an `AVG` query, $\xi$ could be the variance, or the $99\%$ confidence interval for the $\{q(D_i) - \mathfrak{q}(\mathfrak{D})\}$ values. However, since direct computation of $\xi(\mathfrak{q}(\mathfrak{D}), \{q(D_i)\})$ is typically infeasible, a technique called *bootstrap* is often used to approximate this value.

**Bootstrap**. Bootstrap [14] is a powerful technique for approximating unknown distributions. Bootstrap consists of a simple Monte-Carlo procedure: it repeatedly carries out a sub-routine, called a *trial*. Each trial generates a simulated database, say $\hat{D}_j$, which is of the same size as $D$ (by sampling $|D|$ tuples i.i.d. from $D$ *with* replacement), and then computes query $q$ on $\hat{D}_j$. The collection $\{q(\hat{D}_j)\}$ from all the bootstrap trials forms an empirical distribution, based on which $\xi(q(D), \{q(\hat{D}_j)\})$ is computed and returned as an approximation of $\xi(\mathfrak{q}(\mathfrak{D}), \{q(D_i)\})$. Bootstrap is effective and robust across a wide range of practical situations [23, 26, 30].

**Problem Statement**. Our goal is to devise an efficient algorithm to compute the empirical distribution $\{q(\hat{D}_j)\}$ produced by bootstrap, but *without* executing the actual bootstrap trials. In particular, we are interested in the marginal distribution of each individual result tuple, i.e. the probability of each tuple appearing in any $q(\hat{D}_j)$ (as the set of all possible query answers $\{q(\hat{D}_j)\}$ can be very large, namely, of $O(2^{|D|})$ values, making it impractical to return them to the user). This marginal distribution enables us to compute the commonly used quality assessments $\xi$ (e.g., mean, variance, standard deviation, quantiles as used in [23, 26, 30]). Next, we demonstrate this by a small example.

## 2.1 A Small Example of Bootstrap

**Bootstrapping a Database**. To perform a bootstrap trial on a relation $R$ of $n$ tuples, we randomly choose exactly $n$ tuples with replacement from $R$. This produces a multiset relation since a tuple may be drawn more than once. Thus, different trials could result in different multiset relations. This set of multiset relations can be modeled as a single *probabilistic* multiset relation, where each tuple

[2]Note that we do not claim to provide low approximation error for arbitrary queries. For instance, random sampling is known to be futile for certain queries (e.g., joins, `MIN`, `MAX`). However, we guarantee the same empirical distribution that would be produced by thousands of bootstrap trials (which is the state-of-the-art error quantification technique for AQP of complex SQL analytics), whether or not sampling leads to low-error or unbiased answers.

[3]Similar to the original bootstrap [14, 24], we assume that $q$ is given. Deriving $q$ for a given $\mathfrak{q}$ has been discussed in [27].

has a random multiplicity. Let $R^r$ denote the probabilistic relation that results from bootstrapping $R$. Likewise, let $D^r$ denote the probabilistic database obtained by bootstrapping the relations of a database sample $D$. For instance, consider $D$ in Figure 1(a) which has a single relation $R$ (named *stock*), containing three tuples. Figure 1(b) shows a possible instance $\hat{R}$ of $R^r$, produced by a single bootstrap trial, where tuple $t_2$ and $t_3$ are drawn twice and once respectively, while $t_1$ is not selected. For brevity, we denote this resample by $\{(t_2, 2), (t_3, 1)\}$.

$R =$ *stock*

| | Part | Type | Qty |
|---|---|---|---|
| $t_1$ | p01 | a | 4 |
| $t_2$ | p02 | b | 5 |
| $t_3$ | p03 | a | 3 |

(a)

$\hat{R} =$ *stock*

| | Part | Type | Qty |
|---|---|---|---|
| $t_2$ | p02 | b | 5 |
| $t_2$ | p02 | b | 5 |
| $t_3$ | p03 | a | 3 |

(b)

Figure 1: (a) An example of a database sample $D$ with one relation $R$ (named *stock*), and (b) a resample instance of $R^r$

Bootstrapping this particular sampled database $D$ generates 10 possible multiset relations. We refer to these as *possible multiset worlds* of $D^r$, denoted by $pmw(D^r)$. Figure 2 shows all the ten possible instances with their probabilities of being generated.

**Queries on the Resampled Database**. Consider the simple "Important Stock Types" query $q$ in Example 1, which finds stock types having quantity $> 30\%$ of the total quantities. To answer $q$ on $D^r$, one needs to evaluate $q$ against each world in $pmw(D^r)$, then adding up the probabilities of all the worlds that return the same answer. For instance, since $q(D_1) = q(D_2) = \{t_a, t_b\}$ (as in Figure 3(a)), then $\Pr(\{t_a, t_b\}) = \frac{2}{9} + \frac{1}{9} = \frac{1}{3}$. The set of possible answers from evaluating $q$ on $D^r$, denoted by $q(D^r)$, are shown in Figure 3(b).

**Example 1** (Important Stock Types)**.**

```
SELECT distinct Type FROM stock
WHERE Qty > 0.3 * (SELECT SUM(Qty) FROM stock)
```

In general, when $R$ has $n$ tuples, $q(D^r)$ can have as many as $O(2^n)$ possible answers, which makes delivering the distribution on all possible answers impractical. Instead, for each possible tuple in the query result ($t_a$ and $t_b$ in this case) we compute its marginal probability of appearing in the query result—Figure 3(c). For example, $a$ may appear in the result $\{t_a, t_b\}$ or $\{t_a\}$. Thus, the marginal probability of $a$ appearing in any result is $\frac{1}{3} + \frac{8}{27} = \frac{17}{27}$. We denote this marginal distribution by $q^{mrg}(D^r)$.

In real life, instead of our three-tuple example, we have tables with millions, or billions of tuples, where enumerating all possible worlds is impossible. Thus, bootstrap uses a Monte Carlo simulation to approximate the above process, where hundreds or thousands of trials are used to achieve an accurate estimate for $q^{mrg}(D^r)$.

With $q^{mrg}(D^r)$, we can now measure the quality of the approximate answer to $q$ evaluated on sample $D$ shown in Figure 1(a). For instance, say that we are interested in the average false negative rate of tuples. In this case, since computing $q$ directly on $D$ yields $\{t_a, t_b\}$, we evaluate the average probability of missing $a$ or $b$: this can be easily computed as $\frac{1}{2}\{(1 - \Pr(t_a)) + (1 - \Pr(t_b))\} = \frac{1}{3}$.

## 2.2 Scope of Our Approach

| Techniques | Constraints | in TPCH | in Customer Log |
|---|---|---|---|
| Analytic Approach | Simple group-by-aggregate queries w/ no MIN/MAX [4, 10, 11, 18, 19, 20, 21, 29, 40] | 9/22 | 36.9% |
| ABM Semantics | Conjunctive queries with aggregates (no MIN/MAX):<br>(a) projected/group-by/aggregated columns cannot be generated by another aggregate | 18/22 | 99.1% |
| ABM Eligible | (a) all ABM Semantics constraints;<br>(b) with eligible query plan (see Definition 9) | 18/22 | 98.6% |
| ABM DBPTIME-eligible | (a) all ABM Eligible constraints;<br>(b) with DBPTIME-eligible projection (see Theorems 2 and 3) | 16/22 | 59.6% |
| Bootstrap | no MIN/MAX aggregates | 20/22 | 99.1% |

Table 1: Classes of SQL queries supported by different techniques, and their coverage of TPC-H and a major customer's query log.

In this paper, we propose a new approach, called Analytical Bootstrap Method (ABM), which avoids the computational overhead of bootstrap. ABM (1) rigorously models the semantics of bootstrap using a novel *probabilistic multiset database* model, whereby each tuple is annotated with a random variable to capture its random multiplicity (Section 4); (2) employs a new query evaluation technique by extending relational algebra to manage these annotations, in order to succinctly encode all possible query answers that could be generated by bootstrap trials (Section 5); and (3) provides methods to efficiently decode the distributions from the annotations (Section 6).

In this paper, we study the set of conjunctive queries with aggregates, i.e. queries expressed in the following relational algebra: $\sigma$ (selection), $\Pi$ (projection), $\bowtie$ (foreign key join), $\delta$ (deduplication), and $\gamma$ (aggregate). $\gamma_{A,\alpha(B)}$ denotes applying aggregate $\alpha$ on $B$ grouped by $A$. We require that the projected/group-by/aggregated columns are not generated by another aggregate. For ease of

| multiset world | prob. |
|---|---|
| $D_1 = \{(t_1, 1), (t_2, 1), (t_3, 1)\}$ | $2/9$ |
| $D_2 = \{(t_1, 2), (t_2, 1)\}$ | $1/9$ |
| $D_3 = \{(t_1, 2), (t_3, 1)\}$ | $1/9$ |
| $D_4 = \{(t_2, 2), (t_1, 1)\}$ | $1/9$ |
| $D_5 = \{(t_2, 2), (t_3, 1)\}$ | $1/9$ |
| $D_6 = \{(t_3, 2), (t_1, 1)\}$ | $1/9$ |
| $D_7 = \{(t_3, 2), (t_2, 1)\}$ | $1/9$ |
| $D_8 = \{(t_1, 3)\}$ | $1/27$ |
| $D_9 = \{(t_2, 3)\}$ | $1/27$ |
| $D_{10} = \{(t_3, 3)\}$ | $1/27$ |

Figure 2: The possible multiset worlds of $D^r$

| | Type |
|---|---|
| $t_a$ | a |
| $t_b$ | b |

(a)

| answer | prob. |
|---|---|
| $\{t_a, t_b\}$ | $1/3$ |
| $\{t_a\}$ | $8/27$ |
| $\{t_b\}$ | $10/27$ |

(b)

| answer | prob. |
|---|---|
| $t_a$ | $17/27$ |
| $t_b$ | $19/27$ |

(c)

Figure 3: (a) The result of $q(D_1)$ and $q(D_2)$, (b) all possible answers of $q$ and (c) their marginal probabilities

presentation, we first focus on aggregates `SUM`, `COUNT`, and `AVG`, and defer the extension of ABM to more general aggregates to Section 7. Note that we do not consider `MIN` and `MAX`, because these extreme statistics are not supported by bootstrap either [14].

For the set of queries studied, we identify a set of *eligible queries*, for which we provide an efficient extensional evaluation. Further, a subset of eligible queries, namely *DBPTIME*-eligible queries, can be evaluated by our extensional evaluation in *DBPTIME*.

To provide an intuitive sense of how general/restrictive these classes of queries are in practice, in Table 1 we summarize the syntactic constraints imposed by different error estimation techniques along with some statistics. Table 1 compares the generality of our formal semantics, eligible queries, and *DBPTIME*-eligible queries to previous analytical techniques (which are strictly subsumed by ABM) and bootstrap (which strictly subsumes ABM). We have analyzed the 22 TPC-H benchmark queries, as well as a real-life query log from a major customer of an analytical database (referred to as *Customer* log) which consisted of 6660 queries. For each technique, Table 1 reports the fraction of queries (from TPC-H and our Customer log) that are supported by that technique (i.e., queries that satisfy its constraints).

The set of queries supported by ABM strictly subsumes those of previous analytical approaches, and it also constitutes a majority subset of those supported by bootstrap. For instance, ABM supports 18/22 in TPC-H, and $98.6\%$ of the Customer queries, which cover most of the queries supported by bootstrap ($20/22$ in TPC-H and $99.1\%$ in Customer log). Previous analytical approaches only support $9/22$ of TPC-H and $36.9\%$ of the Customer queries. Also, note that almost $60\%$ of Customer queries are supported by ABM while enjoying a guaranteed *DBPTIME* complexity.

# 3 Background

In this section, we provide background on semirings and their connection with relational operators, followed by a brief overview of semiring random variables. These concepts will be used in our probabilistic relational model and query evaluation techniques.

## 3.1 Semirings and Relational Operators

Here, we provide an overview on semirings as well as how they can be used to compute queries on a database.

A **monoid** is a triple $(S, +, 0)$, where:

- $S$ is a set closed under $+$, i.e. $\forall s_1, s_2 \in S$, $s_1 + s_2 \in S$
- $+$ is an associative binary operator, i.e. $\forall s_1, s_2, s_3 \in S$, $(s_1 + s_2) + s_3 = s_1 + (s_2 + s_3)$
- $0$ is the identity element of $+$, i.e. $\forall s \in S$,$s + 0 = 0 + s = s$

For example, $(\mathbb{N}, +, 0)$ is a monoid, where $\mathbb{N}$ is the set of natural numbers, and $0$ and $+$ are the numerical zero and addition. A **semiring** is a quintuplet $(S, +, \cdot, 0, 1)$ which follows the four axioms below:

- $(S, +, 0)$ is a monoid where $+$ is also commutative, i.e. $\forall s_1, s_2 \in S$, $s_1 + s_2 = s_2 + s_1$ (a.k.a. **commutative monoid**)
- $(S, \cdot, 1)$ is a monoid
- $\cdot$ is left and right distributive over $+$, i.e. $\forall a, b, c \in S$, $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(b + c) \cdot a = b \cdot a + c \cdot a$
- $0$ annihilates $S$, i.e. $\forall s \in S$, $0 \cdot s = s \cdot 0 = 0$

A **commutative semiring** is a semiring in which $(S, \cdot, 1)$ is also a commutative monoid. In this paper, all semirings are commutative semirings. E.g., $S_{\mathbb{N}} = (\mathbb{N}, +, \cdot, 0, 1)$ is a commutative semiring.

Green et al. [22] showed that many different extensions of relational algebra can be formulated by annotating database tuples with semiring elements and propagating these annotations during query processing. Consider a multiset database as an example. Tuples in a multiset relation are annotated with natural numbers $\mathbb{N}$, representing their multiplicities (i.e. number of occurrences) in the database.

Formally, a multiset relation $R$ with the tuple domain $U$ is described by an annotation function $\pi_R : U \to \mathbb{N}$, where a tuple $t \in R \Leftrightarrow \pi_R(t) \neq 0$.

During query processing, the relational algebra is extended with the $+$ and $\cdot$ operators in the semiring $S_{\mathbb{N}}$, which manipulate the annotated multiplicities: for projection we add the multiplicities of all input tuples that are projected to the same result tuple; while for join we multiply the multiplicities of joined tuples. That is, inductively we have:

- **Select** $\sigma_c(R)$. $\pi_{\sigma_c(R)}(t) = \pi_R(t) \cdot \mathbb{1}(c(t))$, where $\mathbb{1}(c(t))$ returns 1 if $c(t)$ is true and 0 otherwise.
- **Projection** $\Pi_A(R)$. $\pi_{\Pi_A(R)}(t) = \sum_{t'[A]=t} \pi_R(t')$ where $t'[A]$ is the projection of $t'$ on $A$.
- **Join** $R_1 \bowtie R_2$. $\pi_{R_1 \bowtie R_2}(t) = \pi_{R_1}(t_1) \cdot \pi_{R_2}(t_2)$, where $t_i$ is $t$ on $U_i$.

Green et al. showed that by annotating tuples with elements from $S_{\mathbb{N}}$, and using the extended relational algebra operators defined above, we obtain the relational algebra with multiset semantics. However, This extension is not applicable to bootstrap, because the multiset relation generated by bootstrap is probabilistic (shown in Section 2.1). Thus, we introduce our *probabilistic multiset relational model* in Section 4 by using *semiring random variables* (described next).

## 3.2 Semiring Random Variables

A random variable which takes values from the elements of a semiring $S$ is called a *semiring random variable*, denoted by $S$-rv. Analogous to operators $+$ and $\cdot$ on semiring elements, there are corresponding operators that operate on semiring random variables, called *convolutions*. Below we define the $+$ convolution (denoted by $\oplus$), and the $\cdot$ convolution (denoted by $\odot$).

**Definition 1** (Convolution)**.** *Let $r_1$ and $r_2$ be two $S$-rvs. The convolution $\oplus$ (resp. $\odot$) is a binary operator defined on monoid $(S, +, 0)$ (resp. $(S, \cdot, 0)$), such that $r_1 \oplus r_2$ (resp. $r_1 \odot r_2$) is a $S$-rv, where $\forall s \in S$,*

$$\Pr(r_1 \oplus r_2 = s) = \sum\{\Pr(r_1 = x \wedge r_2 = y) | \forall x, y \in S, x + y = s\}$$
$$\Pr(r_1 \odot r_2 = s) = \sum\{\Pr(r_1 = x \wedge r_2 = y) | \forall x, y \in S, x \cdot y = s\}$$

Similar to conventional random variables, we define the *disjointness*, *independence* and *entailment* between $S$-rvs, but with some special treatment for $0 \in S$.

**Definition 2.** *Let $r_1$ and $r_2$ be two $S$-rvs.*

- *$r_1$ and $r_2$ are **disjoint**, denoted as $r_1 \sqcup r_2$, if $\Pr(r_1 \neq 0 \wedge r_2 \neq 0) = 0$.*
- *$r_1$ and $r_2$ are **independent**, denoted as $r_1 \perp r_2$, if $\forall s_1, s_2 \in S$,*

$$\Pr(r_1 = s_1 \wedge r_2 = s_2) = \Pr(r_1 = s_1) \cdot \Pr(r_2 = s_2)$$

- *$r_1$ **entails** $r_2$, denoted as $r_1 \models r_2$, if $\Pr(r_1 \neq 0 | r_2 \neq 0) = 1$.*

The marginal distribution of a $S$-rv $r$ can be represented by a vector $\mathbf{p}^r$ indexed by $S$, namely $\forall s \in S$, $\mathbf{p}^r[s] = \Pr(r = s)$.

In general, using Definition 1 to compute convolution of $S$-rvscan be quite inefficient, especially when the semiring $S$ is large. However, we make the observation that under certain conditions, the convolution of $S$-rvs can be quickly computed by simply manipulating their probability vectors, as stated next[4].

**Proposition 1** (Efficient Convolution)**.** *Let $r_1$ and $r_2$ be two $S$-rvs on semiring $(S, +, \cdot, 0, 1)$,*

- *If $r_1$ and $r_2$ are disjoint, then $\mathbf{p}^{r_1 \oplus r_2} = \mathbf{p}^{r_1} \uplus \mathbf{p}^{r_2}$, where*

$$(\mathbf{p}^{r_1} \uplus \mathbf{p}^{r_2})[s] \stackrel{\text{def}}{=\!=} \begin{cases} \mathbf{p}^{r_1}[s] + \mathbf{p}^{r_2}[s] & \text{if } s \neq 0 \\ \mathbf{p}^{r_1}[0] + \mathbf{p}^{r_2}[0] - 1 & \text{if } s = 0 \end{cases}$$

- *If $r_1$ entails $r_2$, and $r_2$ is $\{0, 1\}$-valued (can only take 0 or 1), then $\mathbf{p}^{r_1 \odot r_2} = \mathbf{p}^{r_1}$,*

# 4 Semantics & Query Evaluation

Next, we introduce our probabilistic multiset relational model. The basic idea is to annotate each tuple with a $S_{\mathbb{N}}$-rv to represent its nondeterministic multiplicity, and extend the relational algebra to propagate these annotations during query processing. The notations used in the following sections are listed in Table 2.

[4] All the proofs in this paper can be found in Appendix A.

| | |
|---|---|
| $R(D)$ | deterministic relation (database) |
| $R^r(D^r)$ | probabilistic relation (database) from bootstrap |
| $\hat{R}(\hat{D})$ | a resample instance from bootstrap |
| $m_{\hat{R}}(t)$ | multiplicity of $t$ in $\hat{R}$ |
| $U$ | tuple domain of a relation |
| $head(\cdot)$ | set of attributes in the output of a query |
| $rels(\cdot)$ | set of relations occurring in a query |
| $\pi$,$\varpi$ | annotation functions for PMRs |

Table 2: Summary of Notations

## 4.1 Formal Semantics

**Probabilistic Multiset Database Semantics**. A *probabilistic multiset relation* (PMR) is a multiset relation whose tuples have nondeterministic multiplicities. Formally, the functional representation of a PMR $R^r$ is an annotation function $\pi_R : U \to \{S_{\mathbb{N}}\text{-rv}\}$, where $\pi_R(t) = \pi_t$ when tuple $t$ occurs in $R^r$ with a random multiplicity $\pi_t$; otherwise $\pi_R(t) = \mathbf{0}$. Here, $\mathbf{0}$ denotes a special random variable which takes 0 with probability 1.

Specifically, the PMR $R^r$ which results from bootstrapping $R$ (of size $n$) is modeled as follows: for all tuples $\{t_i | i = 1, \cdots, n\}$ in $R$, $(\pi(t_1), \cdots, \pi(t_n))$ jointly follow a multinomial distribution $M(n, [\frac{1}{n}, \cdots, \frac{1}{n}])$. A deterministic relation $R$ could also be modeled as a special PMR, where $\forall t \in R$, $\pi_R(t) = \mathbf{1}$. Here, $\mathbf{1}$ is a special random variable which takes value 1 with probability 1.

**Definition 3** (PMR Semantics)**.** *The semantics of a PMR $R^r$ annotated by $\pi_R : U \to \{S_{\mathbb{N}}\text{-}rv\}$ is a set of possible multiset worlds $pmw(R^r)$, where the probability of each world (i.e. resample instance) $\hat{R}$ is*

$$\Pr(\hat{R}) = \Pr\left(\bigwedge\{\pi_R(t) = m_{\hat{R}}(t) \mid t \in R\}\right)$$

A *probabilistic multiset database* (PMDB) $D^r$ is a database with PMRs.

**Definition 4** (PMDB Semantics)**.** *The semantics of $D^r$ which consists of $k$ PMRs $(R_1^r, R_2^r, \cdots, R_k)$ is a set of possible multiset worlds $pmw(D^r)$ defined as:*

$$pmw(D^r) = \{(\hat{R}_1, \hat{R}_2, \cdots, \hat{R}_k) \mid \hat{R}_i \in pmw(R_i^r), \forall i\}$$

*The probability of each world $\hat{D}$ is $\Pr(\hat{D}) = \prod_{i=1}^{k} \Pr(\hat{R}_i)$.*

**Query Semantics**. Evaluating a query $q$ on a PMDB $D^r$ is equivalent to evaluating $q$ on every possible multiset world in $pmw(D^r)$.

**Definition 5** (Query Semantics)**.** *The semantics of evaluating a query $q$ on a PMDB $D^r$ is a set of possible answers $q^{pmw}(D^r)$, where the probability of each possible answer is:*

$$\forall \hat{R}_q \subseteq U_q : \ \Pr(\hat{R}_q) = \sum\{\Pr(\hat{D}) \mid \hat{D} \in pmw(D^r) \wedge q(\hat{D}) = \hat{R}_q\}$$

As discussed in Section 2.1, computing $q^{pmw}(D^r)$ is infeasible. Instead, we return a marginal summary $q^{mrg}(D^r)$. That is, for each tuple $t \in U_q$, we return a probability vector $\mathbf{p}$, where $\mathbf{p}[m]$ is the probability of $t$ appearing in any possible answer $m$ times, i.e.

$$\mathbf{p}[m] = \sum\{\Pr(\hat{R}_q) | \hat{R}_q \subseteq U_q, \pi_{\hat{R}_q}(t) = m\}$$

## 4.2 Intensional Query Evaluation

Direct application of the possible worlds semantics is impractical due to the prohibitive number of worlds. Thus, in this section we introduce our *intensional evaluation* and its semantics, which lays the theoretical foundation of our efficient evaluation technique which will be introduced in Section 5. The basic idea of intensional evaluation is to extend relational algebra to symbolically propagate the annotations (i.e., $S$-rvs) throughout the query execution.

**Extending Relational Algebra**. Analogous to the multiset semantics introduced in Section 3, we extend relational operators with $\oplus$ and $\odot$ operators from the semiring $(S_{\mathbb{N}}\text{-rv}, \oplus, \odot, \mathbf{0}, \mathbf{1})$, to manipulate the random multiplicities of each tuple, such that query evaluation using these operators will produce the correct results (with respect to the possible multiset worlds semantics). We define this extended relational algebra as follows. (Below we only discuss how aggregates manipulate tuple multiplicities, and will explain how aggregates compute values later.)

**Definition 6** (Intensional Evaluation)**.** *The intensional evaluation is defined inductively on a query Plan $P$:*

- *If $P = \sigma_c(P_1)$, then $\pi_P(t) = \pi_{P_1}(t) \odot \mathbb{1}(c(t))$.*

| tuple | $\varpi_{\texttt{SUM}}$ | $\pi$ |
|---|---|---|
| $t_1$ | $4\pi_1$ | $\pi_1$ |
| $t_2$ | $5\pi_2$ | $\pi_2$ |
| $t_3$ | $3\pi_3$ | $\pi_3$ |

(a)

| $\varpi_{\texttt{SUM}}$ | $\pi$ |
|---|---|
| $4\pi_1 \oplus 5\pi_2 \oplus 3\pi_3$ | $\mathbb{1}(\pi_1 \oplus \pi_2 \oplus \pi_3)$ |

(b) Step ① in Figure 5

| tuple | $\varpi_{\texttt{SUM}}$ | $\pi$ |
|---|---|---|
| $t_1$ | $4\pi_1 \oplus 5\pi_2 \oplus 3\pi_3$ | $\pi_1 \odot \mathbb{1}(\pi_1 \oplus \pi_2 \oplus \pi_3)$ |
| $t_2$ | $4\pi_1 \oplus 5\pi_2 \oplus 3\pi_3$ | $\pi_2 \odot \mathbb{1}(\pi_1 \oplus \pi_2 \oplus \pi_3)$ |
| $t_3$ | $4\pi_1 \oplus 5\pi_2 \oplus 3\pi_3$ | $\pi_3 \odot \mathbb{1}(\pi_1 \oplus \pi_2 \oplus \pi_3)$ |

(c) Step ② in (a)

| tuple | $\pi$ |
|---|---|
| $t_1$ | $\pi_1' = \mathbb{1}(4\pi_1 \oplus 5\pi_2 \oplus 3\pi_3 < 13.3) \odot \pi_1 \odot \mathbb{1}(\pi_1 \oplus \pi_2 \oplus \pi_3)$ |
| $t_2$ | $\pi_2' = \mathbb{1}(4\pi_1 \oplus 5\pi_2 \oplus 3\pi_3 < 16.7) \odot \pi_2 \odot \mathbb{1}(\pi_1 \oplus \pi_2 \oplus \pi_3)$ |
| $t_3$ | $\pi_3' = \mathbb{1}(4\pi_1 \oplus 5\pi_2 \oplus 3\pi_3 < 10) \odot \pi_3 \odot \mathbb{1}(\pi_1 \oplus \pi_2 \oplus \pi_3)$ |

(d) Step ③ in (a)

| | Type | $\pi$ |
|---|---|---|
| $t_a$ | a | $\pi_1' \oplus \pi_3'$ |
| $t_b$ | b | $\pi_2'$ |

(e) Step ④ in (a)

| $\pi_1$ | $\pi_2$ | $\pi_3$ | prob. |
|---|---|---|---|
| 1 | 1 | 1 | 2/9 |
| 2 | 1 | 0 | 1/9 |
| 2 | 0 | 1 | 1/9 |
| 1 | 0 | 2 | 1/9 |
| 3 | 0 | 0 | 1/27 |
| 0 | 0 | 3 | 1/27 |

(f)

Figure 4: (a) initial annotation of $R^r$, (b-e) intensional evaluation steps, and (f) truth table of $\pi_q(t_a) = 1$

- *If* $P = \Pi_A(P_1)$, *then* $\pi_P(t) = \bigoplus_{t'[A]=t} \pi_{P_1}(t')$.
- *If* $P = P_1 \bowtie P_2$, *then* $\pi_P(t) = \pi_{P_1}(t_1) \odot \pi_{P_2}(t_2)$, *where* $t_i = t[head(P_i)]$.
- *If* $P = \delta(P_1)$, *then* $\pi_P(t) = \mathbb{1}(\pi_{P_1}(t))$.
- *If* $P = \gamma_{A,\alpha(B)}(P_1)$, *then* $\pi_P(t) = \pi_{\delta(\Pi_A(P_1))}(t)$.

*where* $\mathbb{1}(r)$ *maps a random variable* $r$ *to another random variable, i.e.* $\mathbb{1}(r) = 1$ *if* $r \neq 0$, *and* $\mathbb{1}(r) = 0$ *otherwise. When* $r$ *is deterministic,* $\mathbb{1}(r)$ *degenerates to* **0** *or* **1**.

**Aggregates**. Besides manipulating the $\pi$ annotation, an aggregate operator produces values that are not in the original input relations; an extra semiring annotation (denoted by $\varpi$) is thus required to define this computation.

Next, we define the intensional evaluation of SUM. Since manipulating $\pi$ is defined in Definition 6, we focus on the manipulation of $\varpi$. (Intensional evaluation of COUNT can be defined similarly and AVG can be defined as an arithmetic function of SUM and COUNT.)

**Definition 7** (Intensional Evaluation of $\gamma_{A,\texttt{SUM}(B)}$)**.** *Intensional evaluation of* $\gamma_{A,SUM(B)}(P)$[5] *is defined as follows:*

1. *Annotate* $P$ *with* $\varpi : U \to \{S_{\mathbb{R}}\text{-}rv\}$ *where* $S_{\mathbb{R}} = (\mathbb{R}, +, \cdot, 0, 1)$. *That is,* $\varpi(t) = t[B] \odot \pi_P(t)$, *where* $t[B]$ *is treated as a degenerated random variable taking a constant value.*
2. *Compute* $\varpi_{\gamma_{A,SUM(B)}(P)}(t) = \bigoplus_{t'[A]=t} \varpi_P(t')$ *and return it as* $SUM(B)$.

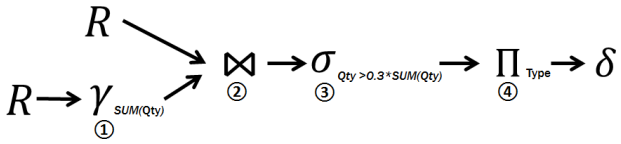


Figure 5: Query plan of Example 1

**Example 2.** *Consider the database* $D$ *in Figure 1(a), and the query* $q$ *in Example 1. Figure 4 shows the intensional evaluation of* $q$ *on* $D^r$ *using the query plan shown in Figure 5. Figure 4(f) shows the truth table for tuple* $t_a$, *from which one can easily calculate that the probability of* $t_a$ *appearing in the result is* $\frac{17}{27}$.

Intensional evaluation of $q$ on $D^r$ produces a new PMR, denoted by $q^i(D^r)$, whose semantics is defined as follows.

**Definition 8** (Intensional Semantics)**.** *The semantics of* $q^i(D^r)$ *is a set of possible multiset worlds,* $pmw(q^i(D^r))$, *where any assignment of the annotations of each tuple* $t$, *namely* $\pi_q(t)$ *and* $\varpi_q(t)$, *yields a possible world instance* $\hat{R}_q$, *such that:*

$$\Pr(\hat{R}_q) = \Pr\left(\bigwedge\{\pi_q(t) = m_{\hat{R}_q}(t) \wedge \varpi_q(t) \models t \mid t \in U_q\}\right)$$

*Here,* $\varpi_q(t) \models t$ *means that* $\varpi_q(t)$ *takes the corresponding aggregate value in* $t$.

Because of the commutative and distributive properties of semirings, $q^i(D^r)$ does not depend on the particular plan chosen for $q$ [22]. Furthermore, this semantics is equivalent to the possible multiset worlds semantics, $q^{pmw}(D^r)$, as stated next.

**Theorem 1.** *For every conjunctive query* $q$ *with aggregates, and every PMDB* $D^r$, *as long as the projected, group-by, and aggregated columns are not generated from another aggregate, we have* $pmw(q^i(D^r)) \equiv q^{pmw}(D^r)$.

Theorem 1 proves the correctness of intensional evaluation. However, intensional evaluation is quite inefficient, since each annotation can grow to the same size as the database, and to compute the distribution, we would need to enumerate all possible values of the annotations. Thus, in the next section, we develop an efficient query evaluation technique.

[5] W.l.o.g, in the following discussion, we assume $dom(B) = \mathbb{R}$.

# 5 Extensional Query Evaluation

In this section, we develop an *extensional evaluation*, which manipulates succinct multinomial representations of the annotations rather than the annotations themselves, and thus, is much more efficient than its intensional counterpart.

For ease of exposition, in this section we restrict our discussion to queries with a single (re)sampled relation and without any self-joins of the (re)sampled relation. (Extensions to general queries are discussed in Section 7.) Next, we review our multinomial representation, and subsequently, introduce our efficient evaluation techniques for queries without and with aggregates.

## 5.1 The Multinomial Representation

We propose a multinomial representation of the annotations based on the following observation.

**Observation 1.** *A bootstrap trial on a relation $R$ of $n$ tuples comprises $n$ i.i.d. experiments. The $i$-th experiment picks a single tuple at random, hence producing a probabilistic relation $R_i^r$. Formally, each $R_i^r$ is annotated by $\rho_i : U \to \{S_{\mathbb{N}}\text{-}rv\}$, such that $\rho_i(t) = 1$ when tuple $t$ is selected, and $\rho_i(t) = 0$ otherwise. The relation $R^r$ resulting from bootstrap is the union of $\{R_i^r\}$. One can easily verify that $\pi(t) = \bigoplus_{i=1}^{n} \rho_i(t)$.*

Based on Observation 1, we define *atoms* of an annotation $\pi(t)$ as the set of annotations from each experiment that compose $\pi(t)$, i.e. $atom(\pi(t)) = \{\rho_i(t) \mid i = 1, \cdots, n\}$. The atoms have the following properties:

- Within the same experiment $i$, $\rho_i(t)$ and $\rho_i(t')$ are disjoint for any two tuples $t, t'$;
- Across different experiment $i$ and $j$, $\rho_i(\cdot)$ and $\rho_j(\cdot)$ are independent;
- $\{\rho_i(t) \mid i = 1, \cdots, n\}$ are i.i.d. $S$-rvs with the same marginal probability vector $\mathbf{p}$, where $\mathbf{p}[0, 1] = [\frac{n-1}{n}, \frac{1}{n}]$.

These properties enable us to uniquely and succinctly represent $\pi(t)$ by a pair $[n, \mathbf{p}]$, namely a *multinomial representation*, which can be interpreted as *the convolution sum of $n$ i.i.d. semiring random variables with the probability vector* $\mathbf{p}$. The probability distribution of $\pi(t)$ can be easily reconstructed from its multinomial representation, using the probability mass function of the Multinomial distribution $M(n, \mathbf{p})$.

More interestingly, when an $atom(\pi(t))$ satisfy certain properties, the convolution operations $\oplus$ and $\odot$ can be directly computed on its multinomial representation. Specifically, using Proposition 1, we have the following rules.

**Proposition 2.** *Let $\pi_1 = [n, \mathbf{p}_1]$ and $\pi_2 = [n, \mathbf{p}_2]$.*

- *If $atom(\pi_1) \cap atom(\pi_2) = \emptyset$, then $\pi_1 \oplus \pi_2 = [n, \mathbf{p}_1 \uplus \mathbf{p}_2]$.*
- *If $atom(\pi_1) \subseteq atom(\pi_2)$, then $\pi_1 \odot \mathbb{1}(\pi_2) = [n, \mathbf{p}_1]$.*

Next, we show how our multinomial representation and its properties enable us to devise an extensional evaluation.

## 5.2 Queries without Aggregates

In this section, we first formally define the set of queries that are *eligible* for efficient evaluation. Then, we present our extensional evaluation algorithm.

**Preliminaries**. Let us denote the sampled relation by $R^f$. To ease the presentation, we base our discussion on *canonical query plans*, which can be obtained by repeating the procedure below on an arbitrary query plan (using relational algebra's rewriting rules [17]):

1. Rename different occurrences of $R^f$ to distinguish them.
2. Push $\sigma$ below $\Pi$ and $\delta$, e.g., $\sigma_c(\Pi_A(R)) \equiv \Pi_A(\sigma_c(R))$.
3. Eliminate duplicate $\delta$, e.g., $\delta(\Pi_A(\delta(R))) \equiv \delta(\Pi_A(R))$.
4. Pull $\Pi$ and $\sigma$ above $\bowtie$, i.e. $R_1 \bowtie \Pi_A(R_2) \equiv \Pi_{head(R_1),A}(R_1 \bowtie R_2)$ and $R_1 \bowtie \sigma_c(R_2) \equiv \sigma_c(R_1 \bowtie R_2)$.
5. If both subqueries of $\bowtie$ are deduplicated, pull one $\delta$ above $\bowtie$, i.e. $\delta(R_1) \bowtie \delta(R_2) \equiv \delta(R_1 \bowtie \delta(R_2))$.

Note that, in the canonical form, $\delta$ is always the last operator of a join subtree. Since we do not consider self-joins of $R^f$, w.l.o.g., we use $\bowtie \delta$ instead of $\bowtie$, e.g., $q_1 \bowtie \delta(q_2)$ means $q_1$ joined with the deduplicated $q_2$.

We also define the *induced functional dependencies* of query $q$, denoted by $\Gamma(q)$ similar to [13]:

- Any functional dependency of $rels(q)$ is in $\Gamma(q)$.
- $R^f.\pi \to head(R^f)$ and $head(R^f) \to R^f.\pi$ are in $\Gamma(q)$, i.e., each tuple in $R^f$ has a unique annotation.
- For every join predicate $R_i.A = R_j.B$, both $R_i.A \to R_j.B$ and $R_j.B \to R_i.A$ are in $\Gamma(q)$;
- For every selection $R_i.A = constant$, $\emptyset \to R_i.A$ is in $\Gamma(q)$.

**Eligible Plan**. A query can be efficiently computed by extensional evaluation, if it has an eligible query plan. For a given $\Gamma(q)$, we define an *eligible plan* as follows.

**Definition 9** (Eligible Plan)**.** *A canonical plan $P$ is eligible if all its operators are eligible:*

- *Operators $\sigma$, $\delta$ and $\bowtie$ are always eligible.*
- *$\Pi_A(P_1)$ is eligible, if for each occurrence of $R^f \in rels(P_1)$ where no $\delta$ appears between $R^f$ and $\Pi_A$, the FDs in $\Gamma(P_1)$ imply $A, R^f.\pi \to head(P_1)$.*

The eligibility of a query can be efficiently checked at compile time. Next, we introduce the extensional evaluation. For ease of presentation, we first restrict ourselves to a strict subset of eligible queries (i.e. those with containment joins), and then generalize to all eligible queries.

**Extensional Evaluation of Queries with Containment Joins**. A join $P_1 \bowtie \delta(P_2)$ is a *containment join*, if one of the following holds:

- $R^f \notin rels(P_2)$.
- $Q = \delta(\Pi_{head(P_1)}(P_1 \bowtie \delta(P_2)))$ contains $\delta(P_1)$, i.e. $\delta(P_1) \subseteq Q$ for any input database.

Whether a join is a containment join can be decided in polynomial time for a large class of conjunctive queries [25].

Given an eligible plan $P$ where all joins are containment joins and standalone deduplication is the last operator of $P$, one can evaluate $P$ via the following extensional evaluation procedure, which directly manipulates the multinomial representations of annotations.

**Definition 10** (Extensional Evaluation (part 1))**.** *The extensional evaluation is defined inductively on a query plan $P$. Let $\pi_{P_1}(t) = [n, \mathbf{p}_t]$, then:*

- *If $P = \sigma_c(P_1)$, then $\pi_P(t) = [n, \mathbf{p}_t]$ if $c(t)$ is true, and $[n, \mathbf{0}]$ otherwise.*
- *If $P = \Pi_A(P_1)$, then $\pi_P(t) = [n, \biguplus_{t'[A]=t} \mathbf{p}_{t'}]$.*
- *If $P = P_1 \bowtie \delta(P_2)$, then $\pi_P(t) = [n, \mathbf{p}_{t_1}]$ where $t_1 = t[head(P_1)]$.*
- *If $P = \delta(P_1)$, then $\pi_P(t) = [1, [\mathbf{p}_t[0]^n, 1 - \mathbf{p}_t[0]^n]]$.*

*To reconstruct $q^{mrg}(D^r)$, we need the probability distribution of $\pi_P(t) = [n, \mathbf{p}]$, which is given by $\mathbf{p}^{\pi_P(t)}[k] = \binom{n}{k}\mathbf{p}[0]^{n-k}\mathbf{p}[1]^k$.*

Next, we illustrates this procedure via an example.

**Example 3.** *Consider the query in Example 1, and its evaluation steps in Figures 4(b) and 4(c). Let $\pi_i = [3, \mathbf{p}_i]$ for $i = 1, 2, 3$, where $\mathbf{p}_i[0, 1] = [2/3, 1/3]$.*

- *$\pi_1 \oplus \pi_2 \oplus \pi_3 = [3, \biguplus_{i=1}^3 \mathbf{p}_i] = [3, \mathbf{p}]$ where $\mathbf{p}[0, 1] = [0, 1]$.*
- *$\pi_i \odot \mathbb{1}(\pi_1 \oplus \pi_2 \oplus \pi_3) = [3, \mathbf{p}_i]$, for $i = 1, 2, 3$.*

Intuitively, the correctness of extensional evaluation is guaranteed by the disjointness and entailment properties implied by the eligibility rules. In other words, an eligible projection guarantees that tuples' annotations are still disjoint after projection; a containment join guarantees joined tuples' annotations satisfy entailment. We formalize this intuition through tracking the "lineage" of tuples. Following the convention in provenance literature [12], we define the *lineage* of annotation $\pi(t)$ during query processing, denoted by $lin(\pi(t))$, as the set of tuples contributing to it. Let $\pi_t$ be the annotation of tuple $t \in R^f$. The following lemma shows that the disjointness and entailment properties hold during extensional evaluation of queries with containment joins.

**Lemma 1.** *For any subplans $P_1$ and $P_2$ of an eligible query with containment joins, where $R^f \in rels(P_1), rels(P_2)$, we have:*

1. *$\forall t \in P_1$, $\pi_{P_1}(t) = \bigoplus_{t \in lin(\pi_{P_1}(t))} \pi_t$.*
2. *$\forall t_1, t_2 \in P_1$, $lin(\pi_{P_1}(t_1)) \cap lin(\pi_{P_1}(t_2)) = \emptyset$.*
3. *$\forall t \in P_1 \bowtie \delta(P_2)$, $lin(\pi_{P_1}(t_1)) \subseteq lin(\pi_{P_2}(t_2))$ where $t_i = t[head(P_i)]$.*

Lemma 1 allows us to use Proposition 2 for manipulating the annotations. It is easy to see that the extensional evaluation will deliver the correct $q^{mrg}(D^r)$ on any $D^r$.

**Extensional Evaluation of General Queries**. For eligible queries with general joins, Lemma 1(3) may not hold. For example, $P_1 \bowtie \delta(P_2)$ may produce different annotations, i.e., if $\pi_{P_2}(t_2) \neq 0$ then $\pi_{P_1 \bowtie \delta(P_2)}(t) = \pi_{P_1}(t_1)$, and otherwise $\pi_{P_1 \bowtie \delta(P_2)}(t) = \mathbf{0}$. The extensional evaluation in Definition 10 cannot be directly applied in this case. To solve this problem, we enumerate all possible values of $\pi_{P_2}$; since for each value, $P_1 \bowtie \delta(P_2)$ is deterministic, Definition 10 becomes applicable.

To facilitate this enumeration, for each tuple $t$, we represent $\pi(t)$ by a set of pairs $\{(\mathfrak{c}_i, \pi_i)\}$, where $\mathfrak{c}_i$ is a set of conjunctive conditions $\{c_j(\pi_{i,j})\}$. A pair $(\mathfrak{c}_i, \pi_i)$ is interpreted as follows: *if all $c_j(\pi_{i,j}) \in \mathfrak{c}_i$ are true, the annotation takes value $\pi_i$*. The set $\{\mathfrak{c}_i, \forall i\}$ has the following properties: (1) any $\mathfrak{c}_i$ and $\mathfrak{c}_j$ are disjoint, and (2) $\{\mathfrak{c}_i, \forall i\}$ enumerates all possible cases. Thus, the distribution of $\pi(t)$ can be reconstructed by a weighted mixture of the distributions under all conditions, i.e.

$$f(\pi(t)) = \sum_{\forall i} \Pr(\mathfrak{c}_i) f(\pi_i \mid \mathfrak{c}_i) \tag{1}$$

However, to compute Equation 1, we need to know the correlations between $\{\pi_{i,j} \in \mathfrak{c}_i\}$ and $\pi_i$, namely the relationship between $lin(\pi_i)$ and the set of $\{lin(\pi_{i,j}), \forall j\}$. To capture these correlations, we extend the multinomial representation as follows. Let $P$ and its subquery $P_j$ be the queries that produce $\pi_i$ and $\pi_{i,j}$, respectively. We can construct "lineage tracing queries" using Cui et al.'s technique [12], denoted by $P^{-1}$ (resp. $P_j^{-1}$), which computes $\bigcup_{\forall t \in P} lin(\pi_i(t))$ (resp. $\bigcup_{\forall t \in P_j} lin(\pi_{i,j}(t))$). Clearly, $P^{-1}$ and $\{P_j^{-1}, \forall j\}$ partition $R^f$ into $2^k$ parts, where $k$ is at most the number of occurrences of $R^f$ in the query. These partitions can be computed as the outer join of $P^{-1}$ and $\{P_j^{-1}, \forall j\}$. Thus, instead of using $[n, \mathbf{p}]$, we use $[n, \mathbf{p}_1, \cdots, \mathbf{p}_{2^k}]$ to represent the annotations, where $\mathbf{p}_l$ corresponds to the tuples in the $l$-th partition. This extended representation is equivalent to $[n, \uplus_{1 \leq l \leq 2^k} \mathbf{p}_l]$. During the extensional evaluation, we manage each $\mathbf{p}_l$ using Proposition 2. Given the correlations, one can easily compute Equation 1. We leave the mathematical analysis to Appendix B.

During query processing, we manipulate both the conditions and the extended multinomial representations. In other words, whenever we combine two conditions, we also modify their corresponding multinomial representations accordingly. Below, we show the query

evaluation procedure. For ease of presentation, we only show the manipulation of the conditions, but omit that of the corresponding multinomial representation, as it will be the same as in Definition 10. We denote the set $\{\mathfrak{c}_i, \forall i\}$ of $\pi_q(t)$ by $\mathbb{c}_q(t)$. Here, $\times$ denotes the set Cartesian product.

**Definition 11** (Extensional Evaluation (part 2))**.** *The extensional evaluation is defined inductively on a query plan $P$:*

- *If $P = \sigma_c(P_1)$, then $\mathbb{c}_P(t) = \mathbb{c}_{P_1}(t)$.*
- *If $P = \Pi_A(P_1)$, then $\mathbb{c}_P(t) = \times_{t'[A]=t} \mathbb{c}_{P_1}(t')$.*
- *If $P = P_1 \bowtie \delta(P_2)$, then $\mathbb{c}_P(t) = \mathbb{c}_{P_1}(t_1) \times \mathbb{c}_{P_2}(t_2) \times \{\{\pi_{P_2}(t_2) \neq 0\}, \{\pi_{P_2}(t_2) = 0\}\}$ where $t_i = t[head(P_i)]$.*
- *If $P = \delta(P_1)$, then $\mathbb{c}_P(t) = \mathbb{c}_{P_1}(t) \times \{\{\pi_{P_1}(t) \neq 0\}, \{\pi_{P_1}(t) = 0\}\}$.*

The extensional evaluation above works for any eligible query $q$. However, for certain queries, in theory one can construct a worst-case database that could make $|\mathbb{c}_q(t)|$ grow exponentially in the size of the database. Next, we identify situations where $|\mathbb{c}_q(t)| = O(1)$, i.e., when *DBPTIME* evaluation is guaranteed[6].

**Definition 12** (*DBPTIME*-Eligible Projection)**.** $\Pi_A(P_1)$ *is* DBPTIME-*eligible, if for every $\delta(P_3)$ and non-containment join $P_2 \bowtie \delta(P_3)$ in $P_1$, $\Gamma(P_1)$ implies $A \rightarrow head(P_3)$.*

If $\Pi_A(P_1)$ is *DBPTIME*-eligible, then $\mathbb{c}_{\pi_A(P_1)}(t) = \mathbb{c}_{P_1}(t')$ for any $t'[A] = t$. Thus, we have the following lemma.

**Lemma 2.** *If every $\Pi_A$ in an eligible plan $P$ is* DBPTIME-*eligible, then $|\mathbb{c}_P(t)| = O(1)$.*

## 5.3 Queries with Aggregates

In this section, we extend our extensional evaluation to queries with aggregates. We first consider aggregates in the return clause of a query, and then aggregates in predicates.

**Handling Aggregates in Return Clauses**. Similar to Section 5.1, the annotation $\varpi$ of aggregate $\gamma_{A,\alpha(B)}$ can be represented in a multinomial representation, such that Proposition 2 still applies to $\varpi$. Taking $\gamma_{A,\texttt{SUM}(B)}(P)$ as an example, let $\pi_P(t) = [n, \mathbf{p}]$. The annotation $\varpi(t)$ can be represented as $[n, \mathbf{p}']$ where $\mathbf{p}'[0] = \mathbf{p}[0]$ and $\mathbf{p}'[t[B]] = \mathbf{p}[1]$.

To check if plan $P$ of a query with aggregates is eligible, one can simply substitute every $\gamma_{A,\alpha(B)}$ in $P$ with $\Pi_A$, and check the eligibility of this modified plan $P^*$. $P$ is eligible if and only if $P^*$ is eligible. Extensional evaluation of $\gamma$ is very similar to $\Pi$ (as in Definition 10 and 11), and is omitted. (When maintaining the multinomial representation $[n, \mathbf{p}]$ of $\varpi$, $\mathbf{p}$ could grow to the database size. Space-efficient approximation is introduced in Section 6.)

**Handling Aggregates in Predicates**. In operator $\sigma_c$, if the predicate $c$ contains aggregate $\gamma(q)$ and $R^f \in rel(q)$ (e.g., see step ③ in Figure 5), then $c(t)$ is uncertain since $\gamma$ is random. Thus, we need to enumerate all possible value ranges of $\gamma$. We modify Definition 11 as follows.

**Definition 13** (Extensional Evaluation (part 3))**.** *We extend Definition 11 by modifying the rule of $\sigma$ as follows:*

- *If $P = \sigma_c(P_1)$, then $\mathbb{c}_P(t) = \mathbb{c}_{P_1}(t) \times enum_c(t)$*

*where $enum_c(t)$ enumerates all possible valuations of predicate $c$, that is, if $c$ is deterministic, $enum_c(t) = \{\mathbf{T}\}$ where $\mathbf{T}$ is the event true; otherwise, $enum_c(t) = \{\{c(t)\}, \{\neg c(t)\}\}$.*

Similar to Section 5.2, the size of $\mathbb{c}_q(t)$ may grow exponentially with the size of the database due to projection and aggregation. Next, we propose an optimization technique to prune conditions with 0 probability, which greatly reduces the size of $\mathbb{c}_q(t)$. Then, we again identify cases where *DB-PTIME* evaluation is guaranteed.

**Observation 2.** *Consider two sets of conditions $\{\gamma < 4, \gamma \geq 4\}$ and $\{\gamma < 3, \gamma \geq 3\}$. A Cartesian product of these two sets produces four conditions. However, clearly $\gamma \geq 4 \wedge \gamma < 3$ cannot be true. In fact, 3 and 4 partition the domain of $\gamma$ into three parts: $(-\infty, 3)$, $[3, 4)$ and $[4, \infty)$. Now a Cartesian product of these two sets produces exactly three valid conditions corresponding to the three partitions, i.e. $\gamma \in (-\infty, 3)$, $\gamma \in [3, 4)$ and $\gamma \in [4, \infty)$.*

We can generalize this observation into the following pruning rule, which reduces the cardinality of the Cartesian product of $m$ condition-sets from $O(2^m)$ to $O(m)$.

**Proposition 3.** *Consider $m$ condition-sets $\mathbb{c}_i = \{\{\gamma < c_i\}, \{\gamma \geq c_i\}\}$, where $c_1 < \cdots < c_m$. Let $c_0 = -\infty$, $c_{m+1} = \infty$.*

$$\times_{i=1}^{m} \mathbb{c}_i = \{\{\gamma \in (c_j, c_{j+1}]\} \mid j = 0, \cdots, m\}$$

Next, we identify situations with efficient condition enumeration.

**Definition 14** (*DBPTIME*-Eligible Aggregate)**.** $\gamma_{A,\alpha(B)}$ *is* DBPTIME-*eligible, if (1) after substituting $\gamma_{A,\alpha(B)}$ with $\delta(\Pi_A)$, $\Pi_A$ is* DBPTIME-*eligible, and (2) for every $\Pi_{A'}(P_1)$ where $\gamma_{A,\alpha(B)}$ appears in $P_1$, $\Gamma(P_1)$ implies $A' \rightarrow A$.*

**Lemma 3.** *If every $\Pi_A$ and $\gamma_{A,\alpha(B)}$ in an eligible plan $P$ are* DBPTIME-*eligible, then $|\mathbb{c}_P(t)| = O(n^{|P|})$, where $n$ is the size of the database.*

[6]However, some queries may still evaluate efficiently in practice, even if they are not *DBPTIME*-eligible, e.g. V4 query in Section 8.

### 5.4 Correctness and Complexity

Let $q^e(D^r)$ be the result of extensional evaluation of query $q$ on $D^r$. The next theorem guarantees that extensional evaluation delivers the correct $q^{mrg}(D^r)$ for eligible queries on any $D^r$, and guarantees an efficient evaluation time for *DBPTIME*-eligible queries.

**Theorem 2.** *For any eligible query q and any PMDB $D^r$, $q^e(D^r) \equiv q^{mrg}(D^r)$. For any* DBPTIME*-eligible query q, $q^e(D^r)$ can be computed in* DBPTIME*.*

The extensional evaluation can efficiently compute the multinomial representations. However, in the presence of aggregates, reconstructing the distribution using Equation 1 can be computationally expensive. For example, the multinomial representation of SUM could be of size $O(n)$ where $n$ is the size of the database. This requires enumerating $O(2^n)$ possible cases to compute the distribution of SUM. However, in many real-world applications, users might be willing to trade accuracy for more efficiency. Thus, in Section 6, we introduce a fast approximation technique for reconstructing the distribution from the output of our extensional evaluation.

## 6 Efficient Approximation

We now describe our solution to the problem left open in Section 5.3. By approximating multinomial representations as asymptotically Gaussian distributions, we efficiently reconstruct the required distributions from extensional evaluation outputs. Further, we approximate the multinomial representation by maintaining a few moments (i.e. mean, variance and covariance) instead of the probability vector itself, which reduces the space overhead of each annotation to $O(1)$.

First, we review an important asymptotic result in Theorem 3. Here, we abuse the notations a bit, and refer to a random variable by its probability vector. W.l.o.g., in the following discussion we refer to annotations in general by $\varpi$, as $\pi$ can be treated as special cases of $\varpi$.

**Theorem 3.** *Let $\varpi_i, i = 1, \cdots, k$ be $k$ semiring random variables, whose multinomial representations are $[n, \mathbf{p}_i]$ respectively. When $n \to \infty$, $\vec{\varpi} = [\varpi_1, \cdots, \varpi_k]$ jointly follows asymptotically a Gaussian distribution:*

$$\vec{\varpi} \sim \mathcal{N}(n\vec{\mu}, n\Sigma)$$

*where $\vec{\mu}_i = \mu_{\mathbf{p}_i}$ the mean of $\mathbf{p}_i$, $\Sigma_{i,i} = \sigma^2_{\mathbf{p}_i}$ the variance of $\mathbf{p}_i$, and $\Sigma_{i,j} = \sigma_{\mathbf{p}_i,\mathbf{p}_j}$ the covariance of $\mathbf{p}_i, \mathbf{p}_j$. Furthermore, applying any differentiable function $\psi$ on $\vec{\varpi}$ still yields an asymptotic Gaussian distribution:*

$$\psi(\vec{\varpi}) \sim \mathcal{N}\left(\psi(n\vec{\mu}), n\nabla\psi(n\vec{\mu})^T\Sigma\nabla\psi(n\vec{\mu})\right)$$

*where $\nabla\psi$ is the gradient of $\psi$.*

This theorem provides accurate approximation whenever the size $n$ of the fact relation is sufficiently large; this is typically the case in large-scale real-life applications. Also, various rules of thumb [9] can be used to check the quality of approximation. With Theorem 3, we can easily compute the probability distribution of SUM and COUNT. Next, we explain how to compute the distribution of AVG, and the conditional probability/distributions (as shown in Equation 1) involving aggregates.

**Computing AVG**. AVG can be expressed as SUM/COUNT. Thus, to compute $\gamma_{A,\texttt{AVG}(B)}$, we rewrite it as two aggregates $\gamma_{A,\texttt{SUM}(B)}$ and $\gamma_{A,\texttt{COUNT}(B)}$, and evaluate them instead of the original AVG. Let $\varpi$ be the AVG, and $\varpi_1, \varpi_2$ be the corresponding SUM and COUNT. We have $\varpi = \psi(\varpi_1, \varpi_2)$ where $\psi(x, y) = x/y$, for which we can directly apply Theorem 3 to compute the distribution.

**Computing Conditions**. As discussed in Section 5.3, there are two types of conditions: (1) comparing an aggregate with a constant, and (2) comparing two aggregates. In this paper, we focus on the comparison operators $\{<, >, \leq, \geq\}$, as applying $\{=, \neq\}$ on SUM is proven to be $\#\mathcal{P}$-complete even for data with binary uncertainty (i.e. either exist or not), which is a degenerated case of our model.

W.l.o.g. consider extensional evaluation output $(\mathfrak{c}, \varpi^*)$ where conditions $\mathfrak{c} = \{\varpi_i < 0, i = 1, \cdots, m\} \cup \{\varpi'_i < \varpi''_i, i = 1, \cdots, n\}$. Let $\vec{\varpi}$, $\vec{\varpi}'$ and $\vec{\varpi}''$ denote the vector $[\varpi_i]$, $[\varpi'_i]$ and $[\varpi''_i]$ respectively. We have the following corollary.

**Corollary 1.** *Let $[\varpi^*, \vec{\varpi}, \vec{\varpi}', \vec{\varpi}''] \sim \mathcal{N}(n\mu, n\Sigma)$. Then*

$$[\varpi^*, \vec{\varpi}, \vec{\varpi}' - \vec{\varpi}''] \sim \mathcal{N}(nA\mu, nA^T\Sigma A) = f$$

*where $A = [1, \mathbf{0}_{1(m+2n)}; \mathbf{0}_{m1}, I_m, \mathbf{0}_{m(2n)}; \mathbf{0}_{n(m+1)}, I_n, -I_n]$. Let $\vec{l} = -\vec{\infty}$ and $\vec{u} = [\infty; \vec{0}]$. Thus, (1) $f(\pi \mid \mathfrak{c})$ is the marginal distribution of $\varpi$ when $f$ is truncated by the range $[\vec{l}, \vec{u}]$; and (2) $\Pr(\mathfrak{c})$ is the cumulative probability in the range $[\vec{l}, \vec{u}]$.*

Computing the truncated probability/distribution in Corollary 1 is well-studied in Statistics. Efficient solutions can be found in [39].

Moreover, Corollary 1 gives us a pruning method to further reduce the number of conditions we need to enumerate, as in Corollary 2, which we can use to quickly prune conditions with extremely low probability.

**Corollary 2.** *Let $\mathfrak{c}$ be a set of conditions. $\Pr(\mathfrak{c})$ is computed by the cumulative probability of $\mathcal{N}(n\vec{\mu}, n\Sigma)$ in the range $[\vec{l}, \vec{u}]$,*

$$\Pr(\mathfrak{c}) \leq \Phi(\frac{\vec{u}_i - n\vec{\mu_i}}{\sqrt{n\Sigma_{i,i}}}) - \Phi(\frac{\vec{l}_i - n\vec{\mu_i}}{\sqrt{n\Sigma_{i,i}}}), \forall i$$

**Sketching Multinomial Representations**. As discussed above, our approximation techniques only requires mean, variance and covariance to reconstruct the distributions. We can use these moments to sketch the multinomial representations without maintaining the actual probability vector; this greatly reduces the storage overhead of our algorithms, achieving $O(1)$ space for each annotation.

As discussed in Section 5, only $\uplus$ modifies a probability vector. Thus, when computing $\mathbf{p}_1 \uplus \mathbf{p}_2$, we directly maintain the mean and variance using the following equations, which follow from the disjointness property.

**Proposition 4.** *Let $\mathbf{p}_1$, $\mathbf{p}_2$ be two semiring random variables where $\mathbf{p}_1 \sqcup \mathbf{p}_2$. The mean and variance of $\mathbf{p}_1 \uplus \mathbf{p}_2$ can be computed by the following equations:*

$$\mu_{\mathbf{p}_1 \uplus \mathbf{p}_2} = \mu_{\mathbf{p}_1} + \mu_{\mathbf{p}_2}$$
$$\sigma^2_{\mathbf{p}_1 \uplus \mathbf{p}_2} = \sigma^2_{\mathbf{p}_1} + \sigma^2_{\mathbf{p}_2} - 2\mu_{\mathbf{p}_1}\mu_{\mathbf{p}_2}$$

Next, we discuss how to compute the covariance of two semiring random variables $\mathbf{p}_1$ and $\mathbf{p}_2$, denoted by $\sigma_{\mathbf{p}_1,\mathbf{p}_2}$, where $[n, \mathbf{p}_i]$ is the multinomial representation of $\varpi_i$ which is produced by a subplan $P_i$. W.l.o.g., we consider the case where both $P_i$'s are aggregates $\gamma_{A_i,\texttt{SUM}(B_i)}(P_i')$, since (1) queries without aggregates and (2) `COUNT` aggregate queries can be treated as special cases of `SUM` aggregate queries. Covariance can be rewritten as a function of means. Therefore, $\sigma_{\mathbf{p}_1,\mathbf{p}_2}$ can be computed by the following procedure.

1. Add a new column $C$ to the sampled relation, where $C$ is defined as:

$$t[C] = \begin{cases} t[B_1] * t[B_2] & \text{if } t \in P_1^{-1} \cap P_2^{-1} \\ 0 & \text{otherwise} \end{cases}$$

2. During query processing, we compute an extra query $\gamma_{A_1,\texttt{SUM}(C)}(P_1')$ (one can also compute query $\gamma_{A_2,\texttt{SUM}(C)}(P_2')$ instead).
3. Let $\varpi_3 = [n, \mathbf{p}_3]$ denote the annotations of $\gamma_{A_1,\texttt{SUM}(C)}(P_1')$. Then $\sigma_{\mathbf{p}_1,\mathbf{p}_2}$ can be computed by Proposition 5 below.

**Proposition 5.** *Let $\mathbf{p}_1$, $\mathbf{p}_2$ be two semiring random variables, and $\mathbf{p}_3$ as described above, the covariance $\sigma_{\mathbf{p}_1,\mathbf{p}_2}$ can be computed by the following equation:*

$$\sigma_{\mathbf{p}_1,\mathbf{p}_2} = \mu_{\mathbf{p}_3} - \mu_{\mathbf{p}_1}\mu_{\mathbf{p}_2}$$

We refer interested readers to Appendix A for detailed proofs.

Note that, in the special case of simple group-by aggregates, our extensional evaluation becomes equivalent to previous analytical approaches. This is because using the approximate multinomial representation, extensional evaluation in these cases simply consists of summing up the annotations of the corresponding tuples, thereby computing the basic statistics in an identical fashion to previous analytical approaches. However, as we show in Table 1, our technique can handle a much larger class of queries.

# 7 Extensions of ABM

In this section, we discuss several extensions of ABM. **General Aggregates**. Many common aggregates can be written as functions of simple aggregates. E.g., `VAR` and `STDEV` can be written as functions of the mean of 1st and 2nd order moments. Other aggregates such as `MEDIAN`, `QUANTILE` can also be computed using our annotation, but with a different CLT approximation as described in [34].

**Multiple Sampled Relations**. It is known that uniform sampling on multiple relations participating in a join yields undesirably sparse results [3]. Join synopses [3] are usually used to solve this problem, where the basic idea is to pre-compute the join (including self-joins), and uniformly sample from the join results. ABM can easily support join synopses, by simply treating the join synopsis as the input sampled relation.

**Using Stratified Samples**. Although uniform samples are widely used in practice, in some cases stratified samples enable better approximations (e.g. for skewed data) where each stratum is itself a uniform sample, but with a different sampling rate than other strata [4, 11]. Bootstrap can also work on stratified samples, by bootstrapping each stratum and combining the resamples from all strata as the simulated dataset [14].

ABM can be easily extended to support stratified samples. Instead of using a single pair $[n, \mathbf{p}]$ to represent each annotation, we extend the multinomial representation to a set of pairs $\{[n_i, \mathbf{p}_i] \mid, i = 1, \cdots, h\}$, one per each stratum. We can simply manipulate this generalized multinomial representation following a similar procedure as described in Section 5. One can even handle the case where some strata are sampled while other small strata are not. In Section 8.4, we study ABM's accuracy on stratified samples.

# 8 Experiments

To evaluate the effectiveness and efficiency of ABM, we conduct experiments on both synthetic and real-world workloads, and compare against both sequential and parallel/distributed implementations of bootstrap.

We use the same commercial DBMS for implementing both bootstrap and ABM. We do not modify the internals of the relational engine, but rather implement a middle layer in Java to re-write the SQL queries to support our extensional evaluation. These modified queries are then executed by the relational engine. The returned results are fed into a post-processing module (implemented in R[7]) to compute the probabilities/distributions. All pre- and post-processing times are included in ABM's execution times.

## 8.1 Experiment Setup

Parallel/distributed experiments are performed on a cluster of 15 machines, each with two 2.20 GHz Intel Xeon E5-2430 CPU cores and 96GB RAM. Sequential experiments use only one of these machines. We report experiments on three workloads: (1) TPC-H benchmark [1], (2) skewed TPC-H benchmark [31] and (3) a real-world dataset and query traces from the largest customer of a major analytical database vendor (referred to as Vendor for anonymity). For details of the datasets and queries see Appendix C.

**TPC-H**. We use a 100 GB benchmark (scale factor of 100). We use 17 queries out of the 22 TPC-H queries, namely: Q1, Q3, Q5-Q12, Q14, Q16-Q20, and Q22[8]. The other queries contain aggregates `MIN`/`MAX`, or otherwise do not satisfy our eligible query conditions. We (re)sample the largest relation *lineitem*. For queries without *lineitem*, we (re)sample the second largest relations, i.e. *customer* and *partsupp*.

**Skewed TPC-H**. We generate a 1 GB micro-benchmark (scale factor 1) using the SSB benchmark [31] (a star schema variation of TPC-H). All the numeric columns follow Zeta distribution with parameter $s \in [2.1, 2.3]$. We use 13 out of the above 17 TPC-H queries after modifying them according to the SSB schema; we leave out Q11, Q16, Q20 and Q22, which are inconsistent with the SSB schema. Again, we (re)sampe the largest relation *lineorder*.

**Vendor**. The Vendor benchmark is of 52 GB and consists of 310 relations. We pick the 6 most complex queries (denote as V1-V6) from the query logs, which have similar query structures as TPC-H queries Q1, Q11, Q18, Q22. Again, for each query, we (re)sample the largest relation. All (re)sampled relations have 9.2 million tuples each, and are 37.8 GB in total.

## 8.2 Error Quantification Accuracy

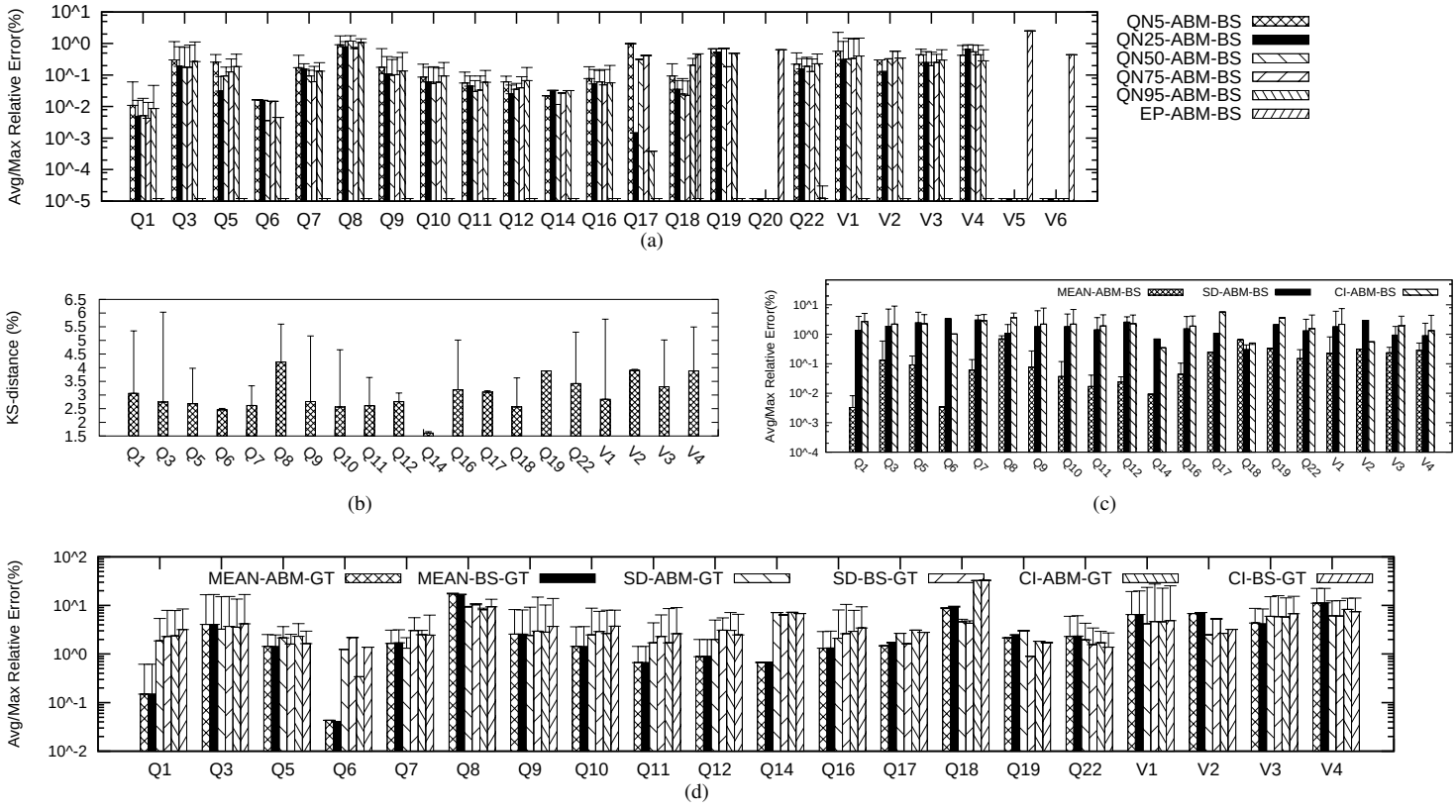


Figure 6: Comparing the distributions given by ABM and bootstrap on (a) Quantiles & existence probabilities, (b) KS distance and (c) User-defined quality measures; (d) Comparing user-defined quality measures given by ABMand bootstrap to ground truth

[7] http://www.r-project.org/

[8] As some of queries produce undesirably sparse results under sampling, we keep the query structures but modify the very selective `WHERE` predicates and/or `GROUP BY` clauses.

In this section, we evaluate the accuracy of our ABM. For each workload and each query $q$, we conduct three sets of experiments:
**1. Ground Truth (GT)**. Similar to [23], we take an $x\%$ random sample from a single relation, leave the other relations intact, and compute $q$ on them. We repeat this procedure $n$ times to collect the empirical distribution of all the $n$ results.
**2. Bootstrap (BS)**. We take an $x\%$ random sample from a single relation. Then, we bootstrap this sample, i.e. we resample the sample, and compute $q$ on each resample and the other intact relations. We repeat the resampling process $n$ times to collect the empirical distribution of all the $n$ results.
**3. ABM**. We take an $x\%$ random sample from a single relation, and apply extensional evaluation to compute $q$ on the sample and the other intact relations. For comparison reason, we use the same random sample as bootstrap.

To compare the predicted distribution of approximate results given by ABM with the empirical distributions given by the ground truth and bootstrap, we measure various distribution statistics, including mean (`MEAN`), standard deviation (`SD`), quantiles (`QN`), $5\%$-$95\%$ confidence interval (`CI`) and existence probability (`EP`) (probability of each tuple appearing in the query's output). For ground truth and bootstrap, we compute these statistics based on the empirical distribution of the collected results, whereas for ABM, we compute these statistics directly on the estimated distribution of the query results. We report the relative error of these statistics given by different methods. In the figures, we use the notation `S-A-B` to denote the relative error of the statistic `S` given by method `A` compared to the same statistic given by method `B`. For example, the relative error of our mean prediction $\mu_{\text{ABM}}$ to the empirical mean produced by Ground Truth $\mu_{\text{GT}}$ is defined as follows `MEAN-ABM-GT` $= |\frac{\mu_{\text{ABM}}-\mu_{\text{GT}}}{\mu_{\text{GT}}}|$. Note that some test queries (e.g. Q20 in TPC-H, V5 and V6 in Vendor) do not return aggregate values, for which we only report the existence probability.

When a query returns more than one column, we compute both the average and maximum relative error of all the columns in the query. Both errors are shown in our figures, where the histograms represent average error, and the T-shaped error bars represent the maximum error.
**1. Predicting the Empirical Distribution of Bootstrap**. In the first set of experiments, we study whether the approximate distribution produced by ABM is an accurate prediction of the empirical distribution given by bootstrap. For this purpose, we compare the two distributions in terms of: (1) `EP-ABM-BS` and `QN`$k$`-ABM-BS` for ($k = 5\%, 10\%, \cdots, 95\%$) as shown in Figure 6(a), and (2) the Kolmogorov-Smirnov (KS) distance between the distributions [9] as shown in Figure 6(b).

We perform the experiments on both TPC-H and Vendor benchmarks with different sample rates ($x = 1\%, 2\%, 5\%, 10\%$) and $n = 1000$ bootstrap trials. Due to space limitations, we only report the results on the smallest sample size $x$=$1\%$. The results on larger sample sizes have better accuracy, and thus are omitted. ABM provides highly accurate predictions of bootstrap across all different metrics and queries: On all quantiles, ABM's largest average relative error is less than $2\%$, while most of the average relative errors are even below $0.1\%$. Also, the maximum relative error is always below $5\%$ across all quantiles. For most queries, both distributions predict the same existence probabilities, i.e. `EP-ABM-BS` is 0. The average KS distance is about $5\%$, which is relatively small for a $1\%$ sampling rate. In summary, these results show that ABM produces highly accurate predictions of the empirical distribution of bootstrap.
**2. Predicting User-defined Quality Measures**. In this set of experiments, we study the accuracy of various user-defined quality measures predicted by ABM. We take $x = 1\%$ samples and conduct $n = 1000$ sampling/bootstrap trials. The comparison results between ABM and bootstrap on TPC-H and Vendor workloads are reported in Figures 6(c). Again, ABM provides highly accurate predictions of bootstrap: On all statistics, ABM's maximum relative error is less than $10\%$, while most of the relative errors are even below $2\%$. We also report the results of comparing both ABM and bootstrap against the ground truth in Figure 6(d) on TPC-H and Vendor workloads. More interestingly, ABM can also serve as an accurate predication of the ground truth. As shown, comparing to the ground truth, most of ABM's average relative errors are below $5\%$. Noticeably, ABM performs very consistent with bootstrap, that is, when bootstrap approximates the ground truth well, ABM predicts very accurately; when bootstrap's approximation has a relatively large error, so does ABM's prediction. This evidences that ABM is equivalent to bootstrap.
**3. Evaluating on skewed TPC-H benchmark**. To study how ABM performs when encounters skewed data, we conduct a micro-benchmark study using the skewed TPC-H workload. We directly run 1000 bootstrap trials on the whole dataset ($x = 100\%, n = 1000$), and compare the user-defined quality measures predicted by ABM and bootstrap. The results are shown in Figure 7(a). Over all queries and all the compared statistics, ABM predicts bootstrap very accurately: ABM's maximum relative error is less than $10\%$, while most of the relative errors are even below $2\%$. which shows that the data skewness does not impact the accuracy of ABM much.
**4. Varying Number of Bootstrap Trials & Sample Size**. We also study the effect of the sample size and the number of bootstrap trials on the prediction accuracy of ABM. We compare ABM with bootstrap on both the distance measures used in Experiment 1 and the user-defined measures used in Experiment 2. Due to space limitations, we only report some representative measures on TPC-H and take the average and maximum of their relative errors across all queries. The other measures and the results on Vendor workloads are similar, and are thus omitted.

To study the effect of the number of bootstrap trials, we fix the sampling rate to $x = 1\%$, but vary the number of bootstrap trails from $n = 100$, $n = 200$, $n = 500$ to $n = 1000$. For each $n$, we compute the relative error of the statistics given by ABM against those given by bootstrap. As shown in Figure 7(b), the relative error between ABM and bootstrap decreases as the number of trials increases. which clearly shows that bootstrap suffers from accuracy loss with a finite number of trials, whereas ABM's analytical modeling of all the possible worlds overcomes this limitation of bootstrap. Moreover, ABM saves the user from cumbersome and error-prone parameter tuning required by bootstrap.

To study the effect of the sample size, we conduct the experiments using a fixed number of bootstrap trials ($n = 1000$), but varying

[9] http://en.wikipedia.org/wiki/Kolmogorov-Smirnov_test

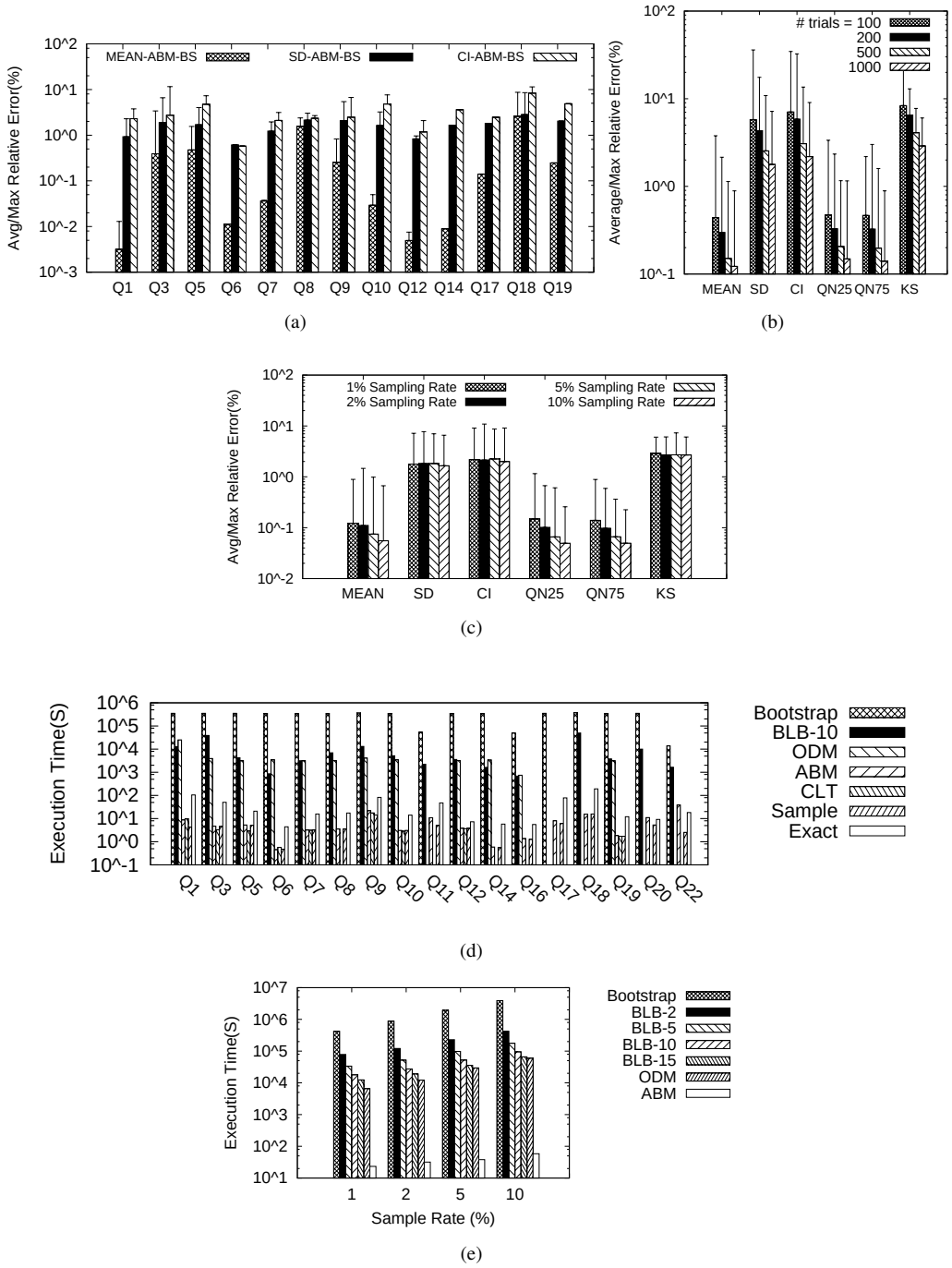


Figure 7: (a) ABM vs. bootstrap on user-defined quality measures for Skewed TPC-H; effect of varying (b) number of bootstrap trails, and (c) sampling rate; comparing time performance of ABM & various techniques (d) under 10% sampling rate, (e) under different sampling rates

the sampling rates ($x = 1\%, 2\%, 5\%, 10\%$). As shown in Figure 7(c), ABM performs stably for `CI`, `SD`, and `KS`, while for more linear statistics (i.e. mean and quantiles) its error further decreases with higher sampling rates. Nonetheless, the average relative error consistently stays below $3\%$ even for $1\%$ samples.

## 8.3 Error Quantification Performance

In this section, we demonstrate the superior speed of our ABM, by comparing it against both sequential and parallel/distributed state-of-the-art bootstrap implementations, as long as CLT-based analytical approach. We show that ABM is $5$ orders of magnitude faster than the naïve bootstrap, and 2-4 orders of magnitude faster than highly optimized variants of bootstrap.

**5. Comparing time performance**. In the first experiment, we compare ABM against two sequential and one parallel/distributed bootstrap

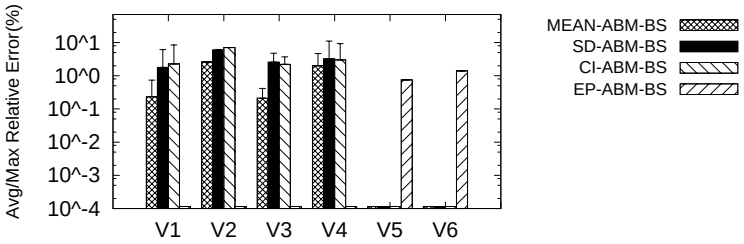


Figure 8: ABMvs. bootstrap under stratified sampling

algorithms: (1) naïve bootstrap, (2) *On-Demand Materialization*(ODM) [30], and (3) Bag of Little Bootstrap (BLB) [24]. To the best of our knowledge, ODM is the best sequantial bootstrap algorithm reported. However, it is limited to simple group-by-aggregate queries. Bag of Little Bootstrap (BLB) [24] is a bootstrap variant optimized for distributed and parallel platforms. We deploy the sequential algorithms (naïve bootstrap and ODM) and ABM on a single machine, while the parallel/distributed algorithm (BLB) on 10 machines. We compare ABM with all the three counterparts on TPC-H ($x = 10\%$). All the counterpart bootstrap algorithms use $n = 1000$ trials. Figure 7(d) reports the running time. ABM is 5 orders of magnitude faster than the naïve bootstrap, and more than 2–4 orders of magnitude faster than ODM. Comparing to BLB, ABM is 2-4 orders of magnitude faster although BLB is using 10 times more computation resources.

We also compare ABM with (1) CLT-based analytical approach (which is only applicable to simple group-by aggregates), (2) executing the approximate query on the sample (Sample), and (3) executing the exact query on the original DB (Exact). This comparison clearly demonstrates the efficiency of ABM: although Sample incurs no overhead but computes the query, and CLT collects minimal statistics, ABM achieves almost identical running time as CLT and Sample; since ABM is computed on $10\%$ sample, ABM achieves almost 10X speed up on every query compared with Exact.

**6. Varying the Sample Size**. Furthermore, as shown in Figure 7(e), the time of naïve bootstrap and ODM increases with the sample size, as the Monte-Carlo simulation is quite computation intensive. On the contrary, our ABM does not vary much in terms of time as the sample size increases, since a large portion of ABM's time is spent on query evaluation, which can be highly optimized by modern database engines.

### 8.4 Using Stratified Samples

We study the accuracy of ABM when applied to stratified samples using the Vendor dataset. We apply different stratification on the 4 relations used in the experiments, consisting of 74 and up to 360 strata. The dataset is skewed such that the smallest stratum contains $1/100000$ of the corresponding relation, while the largest stratum contains $63\%$. We apply the same stratification configuration described in [4]. In particular, we take a stratified sample of size $1\%$ of the original relation equally from each stratum, while the overly small strata are not sampled. Again, we report the relative error of the predictions given by ABM and bootstrap in Figure 8. This result shows that ABM can be extended to support stratified sampling with very good accuracy. This result shows that ABM can be extended to support stratified sampling with very good accuracy.

## 9 Related Work

There has been a large body of research on using sampling to provide quick answers to database queries, on database systems [18, 10, 11, 20, 21, 19, 40, 29, 4] and data stream systems [7, 27]. Approximate aggregate processing has been the focus of many of these works which randomized joins [20], optimal sample construction [11, 4], sample reusing [40], and sampling plan in a stream setting [7, 27]. Most of them use statistical inequalities and the central limit theorem to model the confidence interval or variance of the approximate aggregate answers [18, 11, 20, 19, 40, 4]. Recently, Pansare et al. [29] developed a very sophisticated Bayesian framework to infer the confidence bounds of approximate aggregate answers. However, this approach is limited to simple group-by aggregate queries and does not provide a systematic way of quantifying approximation quality.

Many other works have focused on specific types of queries. For example, Charikar et al. [10] study distinct value estimation from a sample; Joshi and Jermaine [21] propose an EM algorithm to quantify aggregate queries with subset testing.

The bootstrap has become increasingly popular in statistics during the last two decades. Various theoretical [14, 8, 37] and experimental works [30, 26, 23] have proven its effectiveness as a powerful quality assessment tool. And recent works [30, 26] have used bootstrap in a database setting, in order to quantify the quality of approximate query answers. Nevertheless, all these works focus on improving the Monte-Carlo process of the bootstrap. Thus, Pol et al. [30] focus on efficiently generating bootstrap samples in a relational database setting, while Laptev et al. [26] target MapReduce platforms and study how to overlap computation across different bootstrap trials or bootstrap samples.

Another line of related work is approximate query processing in probabilistic databases. Much existing work in this area [13, 38, 5, 32, 33] uses possible world semantics to model uncertain data and its query evaluation. Tuples in a probabilistic database have binary uncertainty, i.e., either exist or not with a certain probability. Specifically, [13, 32] use semirings for modeling and querying probabilistic databases, focusing on conjunctive queries with `HAVING` clauses. On the contrary, we focus on the bootstrap process and model resampled data, using a possible multiset world semantics where database tuples have uncertain multiplicities. Furthermore, bootstrap is fundamentally different from probabilistic databases, since tuples in a resampled relation are always correlated, whereas many probabilistic database assume that tuples are independent [13, 5, 32, 33], or propose new query evaluation methods to handle particular correlations. For instance, [36, 35] propose Gaussian models to process continuous uncertainty data. Our work is instead based on the bootstrap, which is naturally characterized by discrete distributions, rather than the continuous distributions required by previous techniques.

# 10 Conclusion

In this paper, we developed a probabilistic model for the statistical bootstrap process, and showed how it can be used for automatically deriving error estimates for complex database queries. First, we provided a rigorous semantics and a unified analytical model for bootstrap-based error quantification; then we developed efficient query evaluation technique for a general class of analytical SQL queries. Evaluation using the new method is 2–4 orders of magnitude faster than the state-of-the-art bootstrap implementations. Extensive experiments on a variety of synthetic and real-world datasets and queries confirm the effectiveness and superior performance of our approach.

# References


[1] TPC-H Benchmark. http://www.tpc.org/tpch/.

[2] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. The aqua approximate query answering system. In *SIGMOD*, 1999.

[3] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. In *SIGMOD*, 1999.

[4] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *EuroSys*, 2013.

[5] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *ICDE*, 2008.

[6] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *SIGMOD*, 2003.

[7] B. Babcock, M. Datar, and R. Motwani. Load shedding for aggregation queries over data streams. In *ICDE*, 2004.

[8] P. J. Bickel and D. A. Freedman. Some asymptotic theory for the bootstrap. *The Annals of Statistics*, 9(6):1196–1217, 1981.

[9] G. Box, J. S. Hunter, and W. Hunter. *Statistics for experimenters: design, innovation, discovery*. Wiley-Interscience, 2005.

[10] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya. Towards estimation error guarantees for distinct values. In *PODS*, 2000.

[11] S. Chaudhuri, G. Das, and V. R. Narasayya. Optimized stratified sampling for approximate query processing. *TODS*, 32(2), 2007.

[12] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *TODS*, 25(2):179–227, 2000.

[13] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDBJ*, 16, 2007.

[14] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1993.

[15] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, January 1968.

[16] T. S. Ferguson. *A Course in Large Sample Theory*. Chapman and Hall, 1996.

[17] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database systems - the complete book (2. ed.)*. Pearson Education, 2009.

[18] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *SIGMOD*, 1997.

[19] Y. Hu, S. Sundara, and J. Srinivasan. Estimating aggregates in time-constrained approximate queries in oracle. In *EDBT*, 2009.

[20] C. Jermaine, S. Arumugam, A. Pol, and A. Dobra. Scalable approximate query processing with dbo engine. In *SIGMOD*, 2007.

[21] S. Joshi and C. Jermaine. Sampling-based estimators for subset-based queries. *VLDB J.*, 18(1), 2009.

[22] G. Karvounarakis and T. J. Green. Semiring-annotated data: queries and provenance? *SIGMOD Record*, 41(3):5–14, 2012.

[23] A. Kleiner, A. Talwalkar, S. Agarwal, I. Stoica, and M. Jordan. A general bootstrap performance diagnostic. In *KDD*, 2013.

[24] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan. The big data bootstrap. In *ICML*, 2012.

[25] P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *PODS*, 1998.

[26] N. Laptev, K. Zeng, and C. Zaniolo. Early accurate results for advanced analytics on mapreduce. *PVLDB*, 5(10), 2012.

[27] B. Mozafari and C. Zaniolo. Optimal load shedding with aggregates and mining queries. In *ICDE*, 2010.

[28] C. Olston, E. Bortnikov, K. Elmeleegy, F. Junqueira, and B. Reed. Interactive analysis of web-scale data. In *CIDR*, 2009.

[29] N. Pansare, V. R. Borkar, C. Jermaine, and T. Condie. Online aggregation for large mapreduce jobs. *PVLDB*, 4(11), 2011.

[30] A. Pol and C. Jermaine. Relational confidence bounds are easy with the bootstrap. In *SIGMOD*, 2005.

[31] T. Rabl, M. Poess, H.-A. Jacobsen, P. O'Neil, and E. O'Neil. Variations of the star schema benchmark to test the effects of data skew on query performance. In *SPEC*, 2013.

[32] C. Ré and D. Suciu. The trichotomy of having queries on a probabilistic database. *VLDBJ*, 18(5), 2009.

[33] P. Sen, A. Deshpande, and L. Getoor. Read-once functions and query evaluation in probabilistic databases. *PVLDB*, 3(1), 2010.

[34] M. M. Siddiqui and C. Butler. Asymptotic joint distribution of linear systematic statistics from multivariate distributions. *Journal of the American Statistical Association*, 64(325), 1969.

[35] T. T. L. Tran, Y. Diao, C. Sutton, and A. Liu. Supporting user-defined functions on uncertain data. *PVLDB*, 6(6), 2013.

[36] T. T. L. Tran, L. Peng, Y. Diao, A. McGregor, and A. Liu. Claro: modeling and processing uncertain data streams. *VLDBJ*, (5), 2012.

[37] A. van der Vaart and J. Wellner. *Weak Convergence and Empirical Processes*. Springer, corrected edition, Nov. 2000.

[38] D. Z. Wang, E. Michelakis, M. N. Garofalakis, and J. M. Hellerstein. Bayesstore: managing large, uncertain data repositories with probabilistic graphical models. *PVLDB*, 1(1), 2008.

[39] S. Wilhelm. tmvtnorm: A package for the truncated multivariate normal distribution. *sigma*, 2:2.

[40] S. Wu, B. C. Ooi, and K.-L. Tan. Continuous sampling for online aggregation over multiple queries. In *SIGMOD*, 2010.

# A Correctness of Intensional & Extensional Evaluation

## A.1 Background

**Proposition 1** Let $r_1$ and $r_2$ be two $S$-rvs on semiring $(S, +, \cdot, 0, 1)$,

- If $r_1$ and $r_2$ are disjoint, then $\mathbf{p}^{r_1 \oplus r_2} = \mathbf{p}^{r_1} \uplus \mathbf{p}^{r_2}$, where

$$(\mathbf{p}^{r_1} \uplus \mathbf{p}^{r_2})[s] \stackrel{\text{def}}{=\!=} \begin{cases} \mathbf{p}^{r_1}[s] + \mathbf{p}^{r_2}[s] & \text{if } s \neq 0 \\ \mathbf{p}^{r_1}[0] + \mathbf{p}^{r_2}[0] - 1 & \text{if } s = 0 \end{cases}$$

- If $r_1$ entails $r_2$, and $r_2$ is $\{0, 1\}$-valued (can only take 0 or 1), then $\mathbf{p}^{r_1 \odot r_2} = \mathbf{p}^{r_1}$,

*Proof.* *Case* $1^\circ$. Consider two $S$-rv $r_1$ and $r_2$, where $r_1 \sqcup r_2$.

- For $s \in S, s \neq 0$, as $\Pr(r_1 \neq 0 \wedge r_2 \neq 0) = 0$, then

$$\begin{aligned} \Pr(r_1 \oplus r_2 = s) &= \sum_{\substack{\forall x, y \in S, \\ x+y=s}} \Pr(r_1 = x \wedge r_2 = y) \\ &= \Pr(r_1 = 0 \wedge r_2 = s) + \Pr(r_1 = s \wedge r_2 = 0) \\ &= \Pr(r_2 = s) + \Pr(r_1 = s) \end{aligned}$$

- For $s \in S, s = 0$, then

$$\begin{aligned} \Pr(r_1 \oplus r_2 = 0) &= 1 - \sum_{s \neq 0} \Pr(r_1 \oplus r_2 = s) \\ &= 1 - \sum_{s \neq 0} \Pr(r_1 = s) - \sum_{s \neq 0} \Pr(r_2 = s) \\ &= (1 - \sum_{s \neq 0} \Pr(r_1 = s)) + (1 - \sum_{s \neq 0} \Pr(r_2 = s)) - 1 \\ &= \Pr(r_1 = 0) + \Pr(r_2 = 0) - 1 \end{aligned}$$

*Case* $2^\circ$. Consider two $S$-rv $r_1$ and $r_2$, where $r_1 \models r_2$ and $r_2$ is 0, 1-valued.

- For $s \in S, s \neq 0$, then

$$\begin{aligned} \Pr(r_1 \odot r_2 = s) &= \sum_{\substack{\forall x, y \in S, \\ x \cdot y = s}} \Pr(r_1 = x \wedge r_2 = y) \\ &= \Pr(r_1 = s \wedge r_2 = 1) \\ &= \Pr(r_1 = s) \end{aligned}$$

- For $s \in S, s = 0$, then

$$\begin{aligned} \Pr(r_1 \odot r_2 = 0) &= 1 - \sum_{s \neq 0} \Pr(r_1 \odot r_2 = s) \\ &= 1 - \sum_{s \neq 0} \Pr(r_1 = s) \\ &= \Pr(r_1 = 0) \end{aligned}$$

□

## A.2 Semantics & Query Evaluation

**Theorem 1** For every conjunctive query $q$ with aggregates, and every PMDB $D^r$, as long as the projected, group-by, and aggregated columns are not generated from another aggregate, we have $pmw(q^i(D^r)) \equiv q^{pmw}(D^r)$.

*Proof.* Given a world $W$ (i.e. a deterministic multiset database W). We use $pmw(q^i(D^r))[W]$ to represent the possibility given by the intensional semantics. We use $q^{pmw}(D^r)[W]$ to represent the possible multiset worlds semantics. We prove inductively on the size of $q$. If $q$ is a single relation, this holds trivially. Otherwise, $q$ is one of the following five cases. For case (1)-(4), we only show the annotation $\pi$, while the annotation $\varpi$ for any aggregate follows similar proofs and thus are omitted.
(1) $q = \sigma_c(q_1)$.

$$
\begin{aligned}
&q^{pmw}(D^r)[W] = \sum\{Pr(W')|W' \in q_1^{pmw}(D^r), \sigma_c(W') = W\} \\
&= \sum\{Pr(W')|W' \in pmw(q_1^i(D^r)), \sigma_c(W') = W\} \\
&= \sum\{Pr(\bigwedge_{t\in W'}\{m_{q_1}(t) = occ(t,W')\} \wedge \bigwedge_{t\notin W'}\{m_{q_1}(t) = 0\}) \\
&\qquad |W' \in pmw(q_1^i(D^r)), \sigma_c(W') = W\} \\
&= \sum\{Pr(\bigwedge_{t\in W}\{m_{q_1}(t) = occ(t,W) \wedge c(t) = true\} \wedge \bigwedge_{t\notin W, \in W'}\{c(t) = false\} \wedge \bigwedge_{t\notin W'}\{m_{q_1}(t) = 0\}) \\
&\qquad |W' \in pmw(q_1^i(D^r)), \sigma_c(W') = W\} \\
&= Pr(\bigwedge_{t\in W}\{m_{q_1}(t) = occ(t,W) \wedge c(t) = true\} \wedge \bigwedge_{t\notin W}\{m_{q_1}(t) = 0 \vee c(t) = false\}) \\
&= Pr(\bigwedge_{t\in W}\{m_{q_1}(t) \otimes \mathbf{1}(c(t)) = occ(t,W)\} \wedge \bigwedge_{t\notin W}\{m_{q_1}(t) \otimes \mathbf{1}(c(t)) = 0\}) \\
&= pmw(q^i(D^r))[W]
\end{aligned}
$$

(2) $q = \Pi_A(q_1)$.

$$
\begin{aligned}
&q^{pmw}(D^r)[W] = \sum\{Pr(W')|W' \in q_1^{pmw}(D^r), \Pi_A(W') = W\} \\
&= \sum\{Pr(W')|W' \in pmw(q_1^i(D^r)), \Pi_A(W') = W\} \\
&= \sum\{Pr(\bigwedge_{t\in W'}\{m_{q_1}(t) = occ(t,W')\} \wedge \bigwedge_{t\notin W'}\{m_{q_1}(t) = 0\})|W' \in pmw(q_1^i(D^r)), \Pi_A(W') = W\} \\
&= \sum\{Pr(\bigwedge_{t\in W}\bigwedge_{\substack{t'\in W'\\ t'[A]=t}}\{m_{q_1}(t) = occ(t',W')\} \wedge \bigwedge_{t\notin W}\bigwedge_{\substack{t'\in W'\\ t'[A]=t}}\{m_{q_1}(t') = 0\} \\
&\qquad \wedge \bigwedge_{t\notin W'}\{m_{q_1}(t) = 0\})|W' \in pmw(q_1^i(D^r)), \Pi_A(W') = W\} \\
&= \sum\{Pr(\bigwedge_{t\in W}\{\bigwedge_{\substack{t'\in W'\\ t'[A]=t}}\{m_{q_1}(t) = occ(t',W')\} \wedge \sum_{\substack{t'\in W'\\ t'[A]=t}} occ(t',W') = occ(t,W)\} \wedge \bigwedge_{t\notin W}\bigwedge_{\substack{t'\in W'\\ t'[A]=t}}\{m_{q_1}(t') = 0\} \\
&\qquad \wedge \bigwedge_{t\notin W'}\{m_{q_1}(t) = 0\})|W' \in pmw(q_1^i(D^r))\} \\
&= \sum\{Pr(\bigwedge_{t\in W}\{\bigoplus_{\substack{t'\in W'\\ t'[A]=t}} m_{q_1}(t) = occ(t,W)\} \wedge \bigwedge_{t\notin W}\{\bigoplus_{\substack{t'\in W'\\ t'[A]=t}} m_{q_1}(t') = 0\} \wedge \bigwedge_{t\notin W'}\{m_{q_1}(t) = 0\})|W' \in pmw(q_1^i(D^r))\} \\
&= pmw(q^i(D^r))[W]
\end{aligned}
$$

(3) $q = q_1 \bowtie q_2$.

$$
\begin{aligned}
&q^{pmw}(D^r)[W] = \sum\{Pr(W_1 \wedge W_2) | W_1 \in q_1^{pmw}(D^r), W_2 \in q_2^{pmw}(D^r), W_1 \times W_2 = W\} \\
&= \sum\{Pr(W_1 \wedge W_2) | W_1 \in q_1^{pmw}(D^r), W_2 \in q_2^{pmw}(D^r), W_1 \times W_2 = W\} \\
&= \sum\{Pr(\bigwedge_{t_1 \in W_1}\{m_{q_1}(t_1) = occ(t, W_1)\} \wedge \bigwedge_{t_1 \notin W_1}\{m_{q_1}(t_1) = 0\} \wedge \bigwedge_{t_2 \in W_2}\{m_{q_2}(t_2) = occ(t, W_2)\} \\
&\qquad \wedge \bigwedge_{t_2 \notin W_2}\{m_{q_2}(t_2) = 0\}) | W_1 \in q_1^{pmw}(D^r), W_2 \in q_2^{pmw}(D^r), W_1 \times W_2 = W\} \\
&= \sum\{Pr(\bigwedge_{t \in W}\{m_{q_1}(t[q_1]) = occ(t[q_1], W_1) \wedge m_{q_2}(t[q_2]) = occ(t[q_2], W_2)\} \wedge \bigwedge_{t \notin W}\{m_{q_1}(t[q_1]) = 0 \vee m_{q_2}(t[q_2]) = 0\}) \\
&\qquad | W_1 \in q_1^{pmw}(D^r), W_2 \in q_2^{pmw}(D^r), W_1 \times W_2 = W\} \\
&= Pr(\bigwedge_{t \in W}\{m_{q_1}(t[q_1]) \otimes m_{q_1}(t[q_2]) = occ(t, W)\} \wedge \bigwedge_{t \notin W}\{m_{q_1}(t[q_1]) \otimes m_{q_1}(t[q_2]) = 0\}) \\
&= pmw(q^i(D^r))[W]
\end{aligned}
$$

(4) $q = \delta(q_1)$.

$$
\begin{aligned}
&q^{pmw}(D^r)[W] = \sum\{Pr(W') | W' \in q_1^{pmw}(D^r), \delta(W') = W\} \\
&= \sum\{Pr(W') | W' \in pmw(q_1^i(D^r)), \delta(W') = W\} \\
&= \sum\{Pr(\bigwedge_{t \in W'}\{m_{q_1}(t) = occ(t, W')\} \wedge \bigwedge_{t \notin W'}\{m_{q_1}(t) = 0\}) | W' \in pmw(q_1^i(D^r)), \delta(W') = W\} \\
&= \sum\{Pr(\bigwedge_{t \in W}\{m_{q_1}(t) = occ(t, W') \wedge occ(t, W') \neq 0\} \wedge \bigwedge_{t \notin W}\{m_{q_1}(t) = 0\}) \\
&\qquad | W' \in pmw(q_1^i(D^r))\} \\
&= Pr(\bigwedge_{t \in W}\{\mathbf{1}(m_{q_1}(t)) = 1\} \wedge \bigwedge_{t \notin W}\{\mathbf{1}(m_{q_1}(t)) = 0\}) \\
&= pmw(q^i(D^r))[W]
\end{aligned}
$$

(5) $q = \gamma_{A,\alpha(B)}(q_1)$. We use $\alpha = \texttt{SUM}$ as an example, $\varpi$ is the annotation mapping of $\alpha$.

$$
\begin{aligned}
&q^{pmw}(D^r)[W] = \sum\{Pr(W') | W' \in q_1^{pmw}(D^r), \gamma_{A,\alpha(B)}(W') = W\} \\
&= \sum\{Pr(W') | W' \in pmw(q_1^i(D^r)), \gamma_{A,\alpha(B)}(W') = W\} \\
&= \sum\{Pr(\bigwedge_{t \in W'}\{m_{q_1}(t) = occ(t, W') \wedge \varpi_{q_1}(t) = t[B]\} \wedge \bigwedge_{t \notin W'}\{m_{q_1}(t) = 0\}) \\
&\qquad | W' \in pmw(q_1^i(D^r)), \gamma_{A,\alpha(B)}(W') = W\} \\
&= \sum\{Pr(\bigwedge_{t \in W}\bigwedge_{\substack{t' \in W' \\ t'[A]=t}}\{m_{q_1}(t) = occ(t', W') \wedge \varpi_{q_1}(t') = t'[B]\} \wedge \bigwedge_{t \notin W}\bigwedge_{\substack{t' \in W' \\ t'[A]=t}}\{m_{q_1}(t') = 0\} \wedge \bigwedge_{t \notin W'}\{m_{q_1}(t) = 0\}) \\
&\qquad | W' \in pmw(q_1^i(D^r)), \Pi_A(W') = W\} \\
&= \sum\{Pr(\bigwedge_{t \in W}\{\bigwedge_{\substack{t' \in W' \\ t'[A]=t}}\{m_{q_1}(t) = occ(t', W')\} \wedge \sum_{\substack{t' \in W' \\ t'[A]=t}} occ(t', W') = occ(t, W) \wedge \varpi_{q_1}(t') = t'[B] occ(t', W') \\
&\qquad \wedge \sum_{\substack{t' \in W' \\ t'[A]=t}} t'[B] occ(t', W') = t[\alpha]\} \wedge \bigwedge_{t \notin W}\bigwedge_{\substack{t' \in W' \\ t'[A]=t}}\{m_{q_1}(t') = 0\} \wedge \bigwedge_{t \notin W'}\{m_{q_1}(t) = 0\}) | W' \in pmw(q_1^i(D^r))\} \\
&= \sum\{Pr(\bigwedge_{t \in W}\{\bigoplus_{\substack{t' \in W' \\ t'[A]=t}} m_{q_1}(t) = occ(t, W) \wedge \bigoplus_{\substack{t' \in W' \\ t'[A]=t}} \varpi_{q_1}(t') = t[\alpha]\} \wedge \bigwedge_{t \notin W}\{\bigoplus_{\substack{t' \in W' \\ t'[A]=t}} m_{q_1}(t') = 0\} \wedge \bigwedge_{t \notin W'}\{m_{q_1}(t) = 0\}) \\
&\qquad | W' \in pmw(q_1^i(D^r))\} \\
&= pmw(q^i(D^r))[W]
\end{aligned}
$$

Thus, by induction, $pmw(q^i(D^r)) = q^{pmw}(D^r)$. □

## A.3 Extensional Query Evaluation

**Proposition 2** $\pi_1 = [n, \mathbf{p}_1]$ and $\pi_2 = [n, \mathbf{p}_2]$.

- If $atom(\pi_1) \cap atom(\pi_2) = \emptyset$, then $\pi_1 \oplus \pi_2 = [n, \mathbf{p}_1 \uplus \mathbf{p}_2]$.
- If $atom(\pi_1) \subseteq atom(\pi_2)$, then $\pi_1 \odot \mathbb{1}(\pi_2) = [n, \mathbf{p}_1]$.

*Proof.* Consider $\pi_i = \bigoplus_{j=1}^{n} \rho_j^{(i)}$ for $i = 1, 2$.

*Case* $1^\circ$.

$$\pi_1 \oplus \pi_2 = \bigoplus_{j=1}^{n} \rho_j^{(1)} \oplus \bigoplus_{j=1}^{n} \rho_j^{(2)} = \bigoplus_{j=1}^{n} \rho_j^{(1)} \oplus \rho_j^{(2)}$$

As $atom(\pi_1) \cup atom(\pi_2) = \emptyset$ implies that $\rho_j^{(1)} \sqcup \rho_j^{(2)}$ for $\forall j = 1, \cdots, n$, then we have $\mathbf{p}^{\rho_j^{(1)} \oplus \rho_j^{(2)}} = \mathbf{p}^{\rho_j^{(1)}} \uplus \mathbf{p}^{\rho_j^{(2)}}$ by following Proposition 1. Also note that $\rho_j^{(1)} \oplus \rho_j^{(2)}$ are i.i.d. for different $j$ Thus, we have

$$\mathbf{p}^{\pi_1 \oplus \pi_2} = [n, \mathbf{p}_1 \uplus \mathbf{p}_2]$$

*Case* $2^\circ$.

$$\pi_1 \odot \mathbb{1}(\pi_2) = \bigoplus_{j=1}^{n} \rho_j^{(1)} \odot \mathbb{1}(\bigoplus_{j=1}^{n} \rho_j^{(2)}) = \bigoplus_{j=1}^{n} \rho_j^{(1)} \oplus \mathbb{1}(\rho_j^{(2)} \oplus \bigoplus_{k \neq j} \rho_k^{(2)})$$

As $atom(\pi_1) \subset atom(\pi_2)$ implies that $\pi_1 \models \pi_2$, then we have $\pi_1 \models \mathbb{1}(\pi_2)$. By following Proposition 1, we have

$$\mathbf{p}^{\pi_1 \odot \mathbb{1}(\pi_2)} = [n, \mathbf{p}_1]$$

□

**Lemma 1** For any subplans $P_1$ and $P_2$ of an eligible query with containment joins, where $R^f \in rels(P_1), rels(P_2)$, we have:

1. $\forall t \in P_1, \pi_{P_1}(t) = \bigoplus_{t \in lin(\pi_{P_1}(t))} \pi_t$.
2. $\forall t_1, t_2 \in P_1, lin(\pi_{P_1}(t_1)) \cap lin(\pi_{P_1}(t_2)) = \emptyset$.
3. $\forall t \in P_1 \bowtie \delta(P_2), lin(\pi_{P_1}(t_1)) \subseteq lin(\pi_{P_2}(t_2))$ where $t_i = t[head(P_i)]$.

*Proof.* We prove Lemma 1 inductively on the size of the query plan $P$. If $P$ is a single relation, Lemma 1 holds trivially. Otherwise, $P$ is one of the following three cases (Note that $\delta$ can only appear as the last operator in $P$, or with $\bowtie$, thus we omit the discussion of it).

We assume that for subqueries $P_1$ and $P_2$ of $P$, Lemma 1 holds.

(1) $P = \sigma_c(P_1)$. If $c(t) = false$, $\pi_P(t) = \mathbf{0}$, where Lemma 1 holds trivially. Otherwise,

$$\pi_P(t) = \pi_{P_1}(t) = \bigoplus_{t \in lin(\pi_{P_1}(t))} \pi_t$$

Thus, $lin(\pi_P(t)) = lin(\pi_{P_1}(t))$. And Lemma 1 holds for $P$.

(2) $P = \Pi_A(P_1)$.

$$\pi_P(t) = \bigoplus_{t'[A]=t} \pi_{P_1}(t') = \bigoplus_{t'' \in lin(\pi_{P_1}(t')) | \forall t', t'[A]=t} \pi_{t''}$$

Thus, $lin(\pi_P(t)) = \bigcup_{\forall t', t'[A]=t} lin(\pi_{P_1}(t'))$. Obviously, if $\forall t_1, t_2 \in P_1$, $lin(\pi_{P_1}(t_1)) \cap lin(\pi_{P_1}(t_2)) = \emptyset$, then $\forall t, t' \in P$, $lin(\pi_P(t)) \cap lin(\pi_P(t')) = \emptyset$. Therefore, Lemma 1 holds for $P$.

(3) $P = P_1 \bowtie \delta(P_2)$. We prove by contradiction. Assume $lin(\pi_{P_1}(t_1)) \not\subseteq lin(\pi_{P_2}(t_2))$, i.e. there exists such a tuple $t \in R^f$, such that $t \in lin(\pi_{P_1}(t_1))$, but $t \notin lin(\pi_{P_2}(t_2))$. Consider an $R^f$ only containing this tuple $t$. Clearly, $\delta(P_2)$ is empty but $\delta(P_1)$ is not. That is, $\delta(P_1)$ is not contained by $\delta(\Pi_{head(P_1)}(P_1 \bowtie \delta(P_2)))$. We reach a contradiction. Furthermore, Note that

$$lin(\pi_{P_1}(t_1)) \subseteq lin(\pi_{P_2}(t_2)) \Rightarrow \bigoplus_{t \in lin(\pi_{P_1}(t_1))} \pi_t \models \bigoplus_{t \in lin(\pi_{P_2}(t_2))} \pi_t \Rightarrow \pi_P(t) = \pi_{P_1}(t) = \bigoplus_{t \in lin(\pi_{P_1}(t))} \pi_t$$

Thus, $lin(\pi_P(t)) = lin(\pi_{P_1}(t))$. And we prove that Lemma 1 holds for $P$. □

**Lemma 2** If every $\Pi_A$ in an eligible plan $P$ is *DBPTIME*-eligible, then $|\mathbb{c}_P(t)| = O(1)$.

*Proof.* We prove by induction on the size of plan $P$. Assume for subqueries $P_1$ ($P_2$) of $P$, $\mathbb{c}_{P_1}(t)$ ($\mathbb{c}_{P_2}(t)$) has at most $O(1)$) conditions.

1. $P = \sigma_c(P_1)$. Since $\mathbb{c}_P(t) = \mathbb{c}_{P_1}(t)$, then $|\mathbb{c}_P(t)| = |\mathbb{c}_{P_1}(t)| = O(1)$. Lemma 2 holds for $P$.
2. $P = \Pi_A(P_1)$. Since $\Pi_A$ is *DBPTIME*-eligible, thus $\mathbb{c}_{\pi_A(P_1)}(t) = \mathbb{c}_{P_1}(t')$. $|\mathbb{c}_P(t)| = |\mathbb{c}_{P_1}(t')| = O(1)$. Lemma 2 holds for $P$.
3. $P = P_1 \bowtie \delta(P_2)$. Note that $\mathbb{c}_P(t) = \mathbb{c}_{P_1}(t_1) \times \mathbb{c}_{P_2}(t_2) \times \{\{\pi_{P_2}(t_2) \neq 0\}, \{\pi_{P_2}(t_2) = 0\}\}$. But only $\{\pi_{P_2}(t_2) \neq 0\}$ may produce a valid result. Thus $|\mathbb{c}_P(t)| = |\mathbb{c}_{P_1}(t_1)||\mathbb{c}_{P_2}(t_2)| = O(1)$. Lemma 2 holds for $P$.
4. $P = \delta(P_1)$. Note that $\mathbb{c}_P(t) = \mathbb{c}_{P_1}(t) \times \{\{\pi_{P_1}(t) \neq 0\}, \{\pi_{P_1}(t) = 0\}\}$. But only $\{\pi_{P_1}(t) \neq 0\}$ may produce a valid result. Thus $|\mathbb{c}_P(t)| = |\mathbb{c}_{P_1}(t)| = O(1)$. Lemma 2 holds for $P$.

□

**Proposition 3** Consider $m$ condition-sets $\mathbb{c}_i = \{\{\gamma < c_i\}, \{\gamma \geq c_i\}\}$, where $c_1 < \cdots < c_m$. Let $c_0 = -\infty$, $c_{m+1} = \infty$.

$$\times_{i=1}^{m} \mathbb{c}_i = \{\{\gamma \in (c_j, c_{j+1}]\} \mid j = 0, \cdots, m\}$$

*Proof.* Consider any pair $c_i < c_j$. The combination of $\gamma < c_i$ and $\gamma > c_j$ cannot hold. Proposition 3 directly follows this observation. □

**Lemma 3** If every $\Pi_A$ and $\gamma_{A,\alpha(B)}$ in an eligible plan $P$ are *DBPTIME*-eligible, then $|\mathbb{c}_P(t)| = O(n^{|P|})$, where $n$ is the size of the database.

*Proof.* We prove by induction on the size of plan $P$. Assume for subqueries $P_1$ ($P_2$) of $P$, $\mathbb{c}_{P_1}(t)$ ($\mathbb{c}_{P_2}(t)$) has at most $O(n^{|P_1|})$ ($O(n^{|P_2|})$) conditions.

1. $P = \sigma_c(P_1)$. Since $\mathbb{c}_P(t) = \mathbb{c}_{P_1}(t) \times enum_c(t)$ and $enum_c(t) = \{c(t), \neg c(t)\}$, then $|\mathbb{c}_P(t)| = 2|\mathbb{c}_{P_1}(t)| = 2O(n^{|P_1|}) = O(n^{|P|})$. Lemma 3 holds for $P$.
2. $P = \Pi_A(P_1)$. Since $\mathbb{c}_P(t) = \times_{t'[A]=t}\mathbb{c}_{P_1}(t')$. Since $A \to A'$ for any $\gamma_{A',\alpha'(B')}$ appearing the predicates in $q'$, thus for any tuple $t \in P$, $\mathbb{c}_P(t)$ has at most $|P|$ different aggregate instance (Here we call $\gamma_A \mid_{A=a}$ as a single aggregate instance). There are three different cases of predicates involving aggregates: (a) comparing an aggregate with another aggregate, (b) comparing an aggregate with a constant, and (c) comparing an aggregate with an attribute. For case (a) and (b), it is trivial to see that each aggregate instance can introduce at most 2 different conditions. For case (c), assume we compare aggregate $\gamma$ with attribute $C$ in $P_1$. According to Proposition 3, each aggregate instance can introduce at most $O(|P_1|)$ conditions. As implied by the disjointness property in Lemma 1, $|P_1| \leq n$. Thus, we prove that $|\mathbb{c}_q(t)| \leq O(n^{|q|})$. Lemma 3 holds for $P$.
3. $P = \gamma_{A,\alpha(B)}(P_1)$. The proof follows similarly to Case 2, and thus is omitted.
4. $P = P_1 \bowtie \delta(P_2)$. Since $\mathbb{c}_P(t) = \mathbb{c}_{P_1}(t_1) \times \mathbb{c}_{P_2}(t_2) \times \{\{\pi_{P_2}(t_2) \neq 0\}, \{\pi_{P_2}(t_2) = 0\}\}$, then $|\mathbb{c}_P(t)| = 2|\mathbb{c}_{P_1}(t_1)||\mathbb{c}_{P_2}(t_2)| = 2O(n^{|P_1|}) \cdot O(n^{|P_2|}) = 2O(n^{|P_1|+|P_2|}) = O(n^{|P|})$. Lemma 3 holds for $P$.
5. $P = \delta(P_1)$. Since $\mathbb{c}_P(t) = \mathbb{c}_{P_1}(t) \times \{\{\pi_{P_1}(t) \neq 0\}, \{\pi_{P_1}(t) = 0\}\}$, then $|\mathbb{c}_P(t)| = 2|\mathbb{c}_{P_1}(t)| = 2O(n^{|P_1|}) = O(n^{|P|})$. Lemma 3 holds for $P$.

□

**Theorem 2** For any eligible query $q$ and any PMDB $D^r$, $q^e(D^r) \equiv q^{mrg}(D^r)$. For any *DBPTIME*-eligible query $q$, $q^e(D^r)$ can be computed in *DBPTIME*.

*Proof.* The conclusion about correctness directly follows Theorem 1 and the discussion in Section 5. The conclusion about complexity directly follows Lemma 2 and 3. Thus, the proof is omitted. □

## A.4 Efficient Approximation

**Theorem 3** Let $\varpi_i, i = 1, \cdots, k$ be $k$ semiring random variables, whose multinomial representations are $[n, \mathbf{p}_i]$ respectively. When $n \to \infty$, $\vec{\varpi} = [\varpi_1, \cdots, \varpi_k]$ jointly follows asymptotically a Gaussian distribution:

$$\vec{\varpi} \sim \mathcal{N}(n\vec{\mu}, n\Sigma)$$

where $\vec{\mu}_i = \mu_{\mathbf{p}_i}$ the mean of $\mathbf{p}_i$, $\Sigma_{i,i} = \sigma^2_{\mathbf{p}_i}$ the variance of $\mathbf{p}_i$, and $\Sigma_{i,j} = \sigma_{\mathbf{p}_i,\mathbf{p}_j}$ the covariance of $\mathbf{p}_i, \mathbf{p}_j$. Furthermore, applying any differentiable function $\psi$ on $\vec{\varpi}$ still yields an asymptotic Gaussian distribution:

$$\psi(\vec{\varpi}) \sim \mathcal{N}\left(\psi(n\vec{\mu}), n\nabla\psi(n\vec{\mu})^T\Sigma\nabla\psi(n\vec{\mu})\right)$$

where $\nabla\psi$ is the gradient of $\psi$.

*Proof.* Theorem 3 follows straightforwardly the Central Limit Theorem and Cramér's Theorem [16]. Thus we omit the proofs. □

**Proposition 4** Let $\mathbf{p}_1$, $\mathbf{p}_2$ be two semiring random variables where $\mathbf{p}_1 \sqcup \mathbf{p}_2$. The mean and variance of $\mathbf{p}_1 \uplus \mathbf{p}_2$ can be computed by the following equations:

$$\mu_{\mathbf{p}_1\uplus\mathbf{p}_2} = \mu_{\mathbf{p}_1} + \mu_{\mathbf{p}_2}$$
$$\sigma^2_{\mathbf{p}_1\uplus\mathbf{p}_2} = \sigma^2_{\mathbf{p}_1} + \sigma^2_{\mathbf{p}_2} - 2\mu_{\mathbf{p}_1}\mu_{\mathbf{p}_2}$$

*Proof.* Since $\mathbf{p}_1 \sqcup \mathbf{p}_2$,

$$\mu_{\mathbf{p}_1\uplus\mathbf{p}_2} = \sum_{\substack{s\in S\\ s\neq 0}} s\cdot(\mathbf{p}_1[s] + \mathbf{p}_2[s]) + 0\cdot(\mathbf{p}_1[0] + \mathbf{p}_2[0] - 1) = \sum_{\substack{s\in S\\ s\neq 0}} s\cdot\mathbf{p}_1[s] + \sum_{\substack{s\in S\\ s\neq 0}} s\cdot\mathbf{p}_2[s] = \mu_{\mathbf{p}_1} + \mu_{\mathbf{p}_2}$$

According to the Law of Total Variance[10], we divide the space into three conditions: (a) $\mathbf{p}_1 \neq 0 \wedge \mathbf{p}_2 = 0$, (b) $\mathbf{p}_1 = 0 \wedge \mathbf{p}_2 \neq 0$ and (c) $\mathbf{p}_1 = 0 \wedge \mathbf{p}_2 = 0$, thus for case (a) we have

$$\mu_{\mathbf{p}_1\uplus\mathbf{p}_2} = \frac{\mu_{\mathbf{p}_1}}{1 - \mathbf{p}_1[0]}$$

and according to the Law of Total Variance,

$$\sigma^2_{\mathbf{p}_1\uplus\mathbf{p}_2}\cdot(1-\mathbf{p}_1[0]) + (\frac{\mu_{\mathbf{p}_1}}{1-\mathbf{p}_1[0]} - \mu_{\mathbf{p}_1})^2(1-\mathbf{p}_1[0]) + (0-\mu_{\mathbf{p}_1})^2\mathbf{p}_1[0] = \sigma^2_{\mathbf{p}_1} \Rightarrow \sigma^2_{\mathbf{p}_1\uplus\mathbf{p}_2}\cdot(1-\mathbf{p}_1[0]) = \sigma^2_{\mathbf{p}_1} - \mu^2_{\mathbf{p}_1}\frac{\mathbf{p}_1[0]}{1-\mathbf{p}_1[0]}$$

[10] http://en.wikipedia.org/wiki/Law_of_total_variance

We have for case (b) similarly:

$$\mu_{\mathbf{p}_1 \uplus \mathbf{p}_2} = \frac{\mu_{\mathbf{p}_2}}{1 - \mathbf{p}_2[0]}$$

$$\sigma^2_{\mathbf{p}_1 \uplus \mathbf{p}_2} \cdot (1 - \mathbf{p}_2[0]) = \sigma^2_{\mathbf{p}_2} - \mu^2_{\mathbf{p}_2} \frac{\mathbf{p}_2[0]}{1 - \mathbf{p}_2[0]}$$

Therefore,

$$\begin{aligned}
\sigma^2_{\mathbf{p}_1 \uplus \mathbf{p}_2} &= \sigma^2_{\mathbf{p}_1} - \mu^2_{\mathbf{p}_1} \frac{\mathbf{p}_1[0]}{1 - \mathbf{p}_1[0]} + \sigma^2_{\mathbf{p}_2} - \mu^2_{\mathbf{p}_2} \frac{\mathbf{p}_2[0]}{1 - \mathbf{p}_2[0]} \\
&+ (\frac{\mu_{\mathbf{p}_1}}{1 - \mathbf{p}_1[0]} - \mu_{\mathbf{p}_1 \uplus \mathbf{p}_2})^2 \cdot (1 - \mathbf{p}_1[0]) + (\frac{\mu_{\mathbf{p}_2}}{1 - \mathbf{p}_2[0]} - \mu_{\mathbf{p}_1 \uplus \mathbf{p}_2})^2 \cdot (1 - \mathbf{p}_2[0]) + (0 - \mu_{\mathbf{p}_1 \uplus \mathbf{p}_2})^2 \cdot (\mathbf{p}_1[0] + \mathbf{p}_2[0] - 1) \\
&= \sigma^2_{\mathbf{p}_1} + \sigma^2_{\mathbf{p}_2} - 2\mu_{\mathbf{p}_1}\mu_{\mathbf{p}_2}
\end{aligned}$$

□

**Proposition 5** Let $\mathbf{p}_1$, $\mathbf{p}_2$ be two semiring random variables, and $\mathbf{p}_3$ as described above, the covariance $\sigma_{\mathbf{p}_1,\mathbf{p}_2}$ can be computed by the following equation:

$$\sigma_{\mathbf{p}_1,\mathbf{p}_2} = \mu_{\mathbf{p}_3} - \mu_{\mathbf{p}_1}\mu_{\mathbf{p}_2}$$

*Proof.* This theorem directly follows from the equation below to compute covariance. For random variable $X$ and $Y$,

$$Cov(X, Y) = E(XY) - E(X)E(Y)$$

□

# B Constructing Distributions for General Queries without Aggregates

Given a pair of $(\mathfrak{c}, \pi)$, we study the problem of computing $\Pr(\mathfrak{c}) f(\pi \mid \mathfrak{c})$.

In general, $\mathfrak{c}$ can be divided into two sets $\mathfrak{c}_0$ and $\mathfrak{c}_1$, where $\mathfrak{c}_0$ is the set of conditions which are in the form of $\pi_{0,i} = 0$, and $\mathfrak{c}_1$ is the set of conditions which are in the form of $\pi_{1,i} \neq 0$. We use $\bar{\mathfrak{c}_1}$ to denote the condition that *at least one condition from* $\mathfrak{c}_1$ *is false*. Then $\Pr(\mathfrak{c})$ and $f(\pi \mid \mathfrak{c})$ can be computed by the following equations:

$$\Pr(\mathfrak{c}) = 1 - \Pr(\mathfrak{c}_0 \wedge \bar{\mathfrak{c}_1})$$
$$f(\pi \mid \mathfrak{c}) = f(\pi \mid \mathfrak{c}_0) - f(\pi \mid \mathfrak{c}_0 \wedge \bar{\mathfrak{c}_1})$$

The probability (distribution) involving $\bar{\mathfrak{c}_1}$ can be computed by using the Inclusion-Exclusion Principle[15][11].

# C Experiment Queries

## C.1 TPC-H Queries

In TPC-H query set, we select 17 queries with more complex structures out of the 22 TPC-H queries, namely Q1, Q3, Q5-Q12, Q14, Q16-Q20, and Q22. As some of queries produce undesirably sparse results under sampling, we keep the query structures but modify the very selective `WHERE` predicates and/or `GROUP BY` clauses. The queries we used in our experiments are as follows:

**Q1:**

```
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice*(1-l_discount)) as sum_disc_price,
    sum(l_extendedprice*(1-l_discount)*(1+l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= '1998-09-01'
group by
```

[11] http://en.wikipedia.org/wiki/Inclusion%E2%80%93exclusion_principle

```
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus
```

**Q3:**

```
select
    o_orderdate,
    o_shippriority,
    sum(l_extendedprice*(1-l_discount)) as revenue
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'BUILDING'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < '1995-07-01'
    and o_orderdate > '1994-01-01'
    and l_shipdate > '1994-01-01'
group by
    o_orderdate,
    o_shippriority
order by
    o_orderdate
```

**Q5:**

```
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'AMERICA'
    and o_orderdate >= '1995-01-01'
    and o_orderdate < '1996-01-01'
group by
    n_name
order by
    n_name desc
```

**Q6:**

```
select
    sum(l_extendedprice * l_discount) as revenues
from
    lineitem
where
    l_shipdate >= '1996-01-01'
    and l_shipdate < '1997-01-01'
    and l_discount between 0.06 and 0.08
    and l_quantity < 24
```

**Q7:**

```
select
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from (
    select
        n1.n_name as supp_nation,
        n2.n_name as cust_nation,
    substring(l_shipdate, 1, 4) as l_year,
        l_extendedprice * (1 - l_discount) as volume
    from
        supplier,
        lineitem,
        orders,
        customer,
        nation n1,
        nation n2
    where
        s_suppkey = l_suppkey
        and o_orderkey = l_orderkey
        and c_custkey = o_custkey
        and s_nationkey = n1.n_nationkey
        and c_nationkey = n2.n_nationkey
        and (
                (n1.n_name = 'VIETNAM' and n2.n_name = 'KENYA')
                or (n1.n_name = 'KENYA' and n2.n_name = 'VIETNAM')
        )
        and l_shipdate between '1995-01-01' and '1996-12-31'
    ) as shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year
```

**Q8:**

```
select
    o_year,
    sum(case
        when nation = 'JAPAN' then volume
        else 0
    end) / sum(volume) as mkt_share
from (
    select
        substring(o_orderdate, 1, 4) as o_year,
        l_extendedprice * (1 - l_discount) as volume,
        n2.n_name as nation
    from
        part,
        supplier,
        lineitem,
        orders,
        customer,
        nation n1,
        nation n2,
        region
    where
        p_partkey = l_partkey
```

```
        and s_suppkey = l_suppkey
        and l_orderkey = o_orderkey
        and o_custkey = c_custkey
        and c_nationkey = n1.n_nationkey
        and n1.n_regionkey = r_regionkey
        and r_name = 'ASIA'
        and s_nationkey = n2.n_nationkey
        and o_orderdate between '1995-01-01' and '1996-12-31'
        and p_type = 'LARGE POLISHED BRASS'
    ) as all_nations
group by
    o_year
order by
    o_year
```

**Q9:**

```
select
    nation,
    o_year,
    sum(amount) as sum_profit
from (
    select
        n_name as nation,
        substring(o_orderdate,1,4) as o_year,
        l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
    from
        part,
        supplier,
        lineitem,
        partsupp,
        orders,
        nation
    where
        s_suppkey = l_suppkey
        and ps_suppkey = l_suppkey
        and ps_partkey = l_partkey
        and p_partkey = l_partkey
        and o_orderkey = l_orderkey
        and s_nationkey = n_nationkey
        and p_name like '%ghost%'
    ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc
```

**Q10:**

```
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
from
    customer,
    orders,
    lineitem,
    nation
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate >= '1994-10-01'
    and o_orderdate < '1995-01-01'
    and l_returnflag = 'R'
    and c_nationkey = n_nationkey
```

```
group by
    n_name,
order by
    n_name
```

**Q11:**

```
select
    n_nationkey,
    sum(ps_supplycost * ps_availqty) as value
from
    partsupp,
    supplier,
    nation
where
    ps_suppkey = s_suppkey
    and s_nationkey = n_nationkey
group by
    n_nationkey
having
    sum(ps_supplycost * ps_availqty) > (
        select
            sum(ps_supplycost * ps_availqty ) * 0.00002
        from
            partsupp,
            supplier,
            nation
        where
            ps_suppkey = s_suppkey
            and s_nationkey = n_nationkey
        )
order by
    n_nationkey
```

**Q12**

```
select
    l_shipmode,
    sum(case
            when o_orderpriority = '1-URGENT'
                or o_orderpriority = '2-HIGH'
            then 1
            else 0
        end ) as high_line_count,
    sum(case
            when o_orderpriority <> '1-URGENT'
                and o_orderpriority <> '2-HIGH'
            then 1
            else 0
        end ) as low_line_count
from
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('RAIL', 'MAIL')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= '1993-01-01'
    and l_receiptdate < '1994-01-01'
group by
    l_shipmode
order by
    l_shipmode;
```

**Q14:**

```
select
    100.00 * sum(case
            when p_type like 'PROMO%'
            then l_extendedprice * (1 - l_discount)
            else 0
    end ) / sum(l_extendedprice * (1 - l_discount))  as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
    and l_shipdate >= '1996-12-01'
    and l_shipdate < '1997-01-01'
```

**Q16:**

```
select
    p_brand,
    p_type,
    p_size,
    count(ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#43'
    and p_type not like 'STANDARD BURNISHED%'
    and p_size in (22, 7, 8, 35, 33, 11, 31, 39)
    and ps_suppkey not in (
        select
            s_suppkey
        from
            supplier
        where
            s_comment like '%Customer%Complaints%'
        )
group by
    p_brand,
    p_type,
    p_size
order by
    p_brand,
    p_type,
    p_size
```

**Q17:**

```
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part as opart
where
    p_partkey = l_partkey
    and p_brand = 'Brand#42'
    and p_container = 'JUMBO BAG'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem,
            part as ipart
        where
            l_partkey = ipart.p_partkey
            and ipart.p_mfgr = opart.p_mfgr
```

```
    )
```

**Q18:**

```
select
    c_nationkey,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderpriority in (
        select
            o_orderpriority
        from
            orders,
            lineitem
        where
            o_orderkey = l_orderkey
        group by
            o_orderpriority
        having
            sum(l_quantity) > 3060250000
    )
    and c_custkey = o_custkey
    and o_orderkey = l_orderkey
group by
    c_nationkey
order by
    c_nationkey
```

**Q19:**

```
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    ((
        p_partkey = l_partkey
        and p_brand = 'Brand#45'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 7 and l_quantity <= 7 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#51'
        and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 20 and l_quantity <= 20 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#51'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 28 and l_quantity <= 28 + 10
        and p_size between 1 and 15
```

```
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    ))
```

**Q20:**

```
with lqty(nationkey, qty) as (
    select
        s_nationkey, sum(l_quantity) * 0.0001
    from
        lineitem,
        supplier
    where
        l_suppkey = s_suppkey
        and l_shipdate >= '1996-01-01'
        and l_shipdate < '1997-01-01'
    group by
        s_nationkey
)
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like 'cornsilk%'
                )
            and ps_availqty > (
                select
                    qty
                from
                    lqty,
                    supplier
                where
                    s_suppkey = ps_suppkey
                    and s_nationkey = nationkey)
        )
    and s_nationkey = n_nationkey
    and n_name = 'ETHIOPIA'
order by
    s_name
```

**Q22:**

```
select
    cntrycode,
    count(*) as numcust,
    sum(c_acctbal) as totacctbal
from (
    select
        substring(c_phone, 1, 2) as cntrycode,
        c_acctbal
    from
```

```
        customer
    where
        substring(c_phone, 1, 2) in
            ('26', '29', '31', '28', '30', '25', '40')
        and c_acctbal > (
            select
                avg(c_acctbal)
            from
                customer
            where
                c_acctbal > 0.00
                and substring(c_phone, 1, 2) in
                    ('26', '29', '31', '28', '30', '25', '40')
            )
        and not exists (
            select
                *
            from
                orders
            where
                o_custkey = c_custkey
            )
        ) as custsale
group by
    cntrycode
order by
    cntrycode;
```

## C.2 Skewed TPC-H Queries

In SSB, we use 12 out of the 17 TPC-H queries (except Q11, Q16, Q17, Q20 and Q22, as they do not conform to the SSB schema), but modify the queries according to the SSB schema as follows:

**Q1:**

```
select
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice*(1-l_discount)) as sum_disc_price,
    sum(l_extendedprice*(1-l_discount)*(1+l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineorder
```

**Q3:**

```
select
    l_orderdate,
    l_shippriority,
    sum(l_extendedprice*(1-l_discount*1.0/100)) as revenue
from
    customer,
    lineorder
where
    c_mktsegment = 'BUILDING'
    and c_custkey = l_custkey
    and l_orderdate < '1995-07-01'
    and l_orderdate > '1994-01-01'
order by
    l_orderdate,
    l_shippriority
```

**Q5:**

```
select
    c_nation,
    sum(l_extendedprice * (1 - l_discount*1.0/100)) as revenue
from
    customer,
    lineorder,
    supplier
where
    c_custkey = l_custkey
    and l_suppkey = s_suppkey
    and c_nation = s_nation
    and c_region = 'AMERICA'
    and l_orderdate >= '1995-01-01'
    and l_orderdate < '1996-01-01'
order by
    c_nation desc
```

**Q6:**

```
select
    sum(l_extendedprice * l_discount*1.0/100) as revenue
from
    lineorder
where
    l_discount between 6 and 8
    and l_quantity < 24
```

**Q7:**

```
select
    s_nation,
    c_nation,
    sum(volume) as revenue
from (
    select
        s_nation,
        c_nation,
        l_extendedprice * (1 - l_discount * 1.0/100) as volume
    from
        supplier,
        lineorder,
        customer
    where
        s_suppkey = l_suppkey
        and c_custkey = l_custkey
        and (
            (c_nation = 'ALGERIA' and s_nation = 'ARGENTINA')
            or (c_nation = 'ARGENTINA' and s_nation = 'ALGERIA')
        )
    )
group by
    s_nation,
    c_nation
order by
    s_nation,
    c_nation
```

**Q8:**

```
select
    o_year,
    sum(case
        when s_nation = 'JAPAN' then volume
        else 0
    end)/sum(volume) as mkt_share
```

```
from (
    select
        substring(l_orderdate, 1, 4) as o_year,
        l_extendedprice * (1 - l_discount * 1.0/100) as volume,
        s_nation
    from
        part,
        supplier,
        lineorder,
        customer
    where
        p_partkey = l_partkey
        and s_suppkey = l_suppkey
        and l_custkey = c_custkey
        and c_region = 'AMERICA'
        and l_orderdate between '1995-01-01' and '1996-12-31'
        and p_type = 'LARGE POLISHED BRASS')
group by
    o_year
order by
    o_year
```

**Q9:**

```
select
    nation,
    l_year,
    sum(amount) as sum_profit
from (
    select
        s_nation as nation,
        substring(l_orderdate,1,4) as l_year,
        (l_extendedprice * (1 - l_discount * 1.0/100) - l_supplycost * l_quantity)  as amount
    from
        part,
        supplier,
        lineorder
    where
        s_suppkey = l_suppkey
        and p_partkey = l_partkey
        and p_name like '%ghost%'
        and s_nation in ('BRAZIL','ALGERIA','ARGENTINA')
    )
group by
    nation,
    l_year
order by
    nation,
    l_year desc
```

**Q10:**

```
select
    c_nation,
sum(l_extendedprice * (1 - l_discount * 1.0/100)) as revenue
from
    customer,
    lineorder
where
    c_custkey = l_custkey
    and l_orderdate >= '1994-10-01'
    and l_orderdate < '1995-01-01'
    and c_nation in ('ARGENTINA','ALGERIA','BRAZIL')
group by
    c_nation
order by
```

```
    c_nation
```

**Q12:**

```
select
    l_shipmode,
    sum(case
            when l_orderpriority = '1-URGENT'
                or l_orderpriority = '2-HIGH'
            then 1
            else 0
        end) as high_line_count,
    sum(case
            when l_orderpriority <> '1-URGENT'
                and l_orderpriority <> '2-HIGH'
            then 1
            else 0
        end) as low_line_count
from
    lineorder
where
    l_shipmode in ('RAIL', 'MAIL')
group by
    l_shipmode
order by
    l_shipmode
```

**Q14:**

```
select
    sum(100.00 * case
            when p_type like 'PROMO%'
            then l_extendedprice * (1 - l_discount*1.0/100)
            else 0
        end)/sum(l_extendedprice * (1 - l_discount*1.0/100))
from
    lineorder,
    part
where
    l_partkey = p_partkey
```

**Q18:**

```
select
    c_nation,
    sum(l_quantity)
from
    customer,
    lineorder
where
    l_orderpriority in (
        select
            l_orderpriority
        from
            lineorder
group by
            l_orderpriority
having
            sum(l_orderpriority) > 6180000
    )
    c_custkey = l_custkey
    and c_nation <> 'UNITED KINGDOM'
group by
    c_nation
order by
    c_nation
```

**Q19:**

```
select
    sum(l_extendedprice* (1 - l_discount*1.0/100))
from
    lineorder,
    part
where
    ((
        p_partkey = l_partkey
        and p_brand = 'MFGR#115'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 1 and l_quantity <= 1 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'MFGR#116'
        and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 2 and l_quantity <= 2 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'MFGR#112'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 3 and l_quantity <= 3 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
    ))
```