Algorithms for Provisioning Queries and Analytics

Sepehr Assadi* Sanjeev Khanna* Yang Li* Val Tannen†

Abstract

Provisioning is a technique for avoiding repeated expensive computations in *what-if analysis*. Given a query, an analyst formulates k *hypotheticals*, each retaining some of the tuples of a database instance, *possibly overlapping*, and she wishes to answer the query under *scenarios*, where a scenario is defined by a subset of the hypotheticals that are "turned on". We say that a query admits *compact* provisioning if given any database instance and any k hypotheticals, one can create a poly-size (in k) *sketch* that can then be used to answer the query under any of the 2^k possible scenarios without accessing the original instance.

In this paper, we focus on provisioning complex queries that combine relational algebra (the logical component), grouping, and statistics/analytics (the numerical component). We first show that queries that compute quantiles or linear regression (as well as simpler queries that compute count and sum/average of positive values) can be compactly provisioned to provide (multiplicative) *approximate* answers to an arbitrary precision. In contrast, *exact* provisioning for each of these statistics requires the sketch size to be exponential in k. We then establish that for any complex query whose logical component is a *positive* relational algebra query, as long as the numerical component can be compactly provisioned, the complex query itself can be compactly provisioned. On the other hand, introducing negation *or* recursion in the logical component again requires the sketch size to be exponential in k. While our positive results use algorithms that do not access the original instance after a scenario is known, we prove our lower bounds even for the case when, knowing the scenario, limited access to the instance is allowed.

*Department of Computer and Information Science, University of Pennsylvania. Supported in part by National Science Foundation grants CCF-1116961, CCF-1552909, and IIS-1447470 and an Adobe research award.
Email: {sassadi,sanjeev,yangli2}@cis.upenn.edu.

†Department of Computer and Information Science, University of Pennsylvania. Supported in part by National Science Foundation grants IIS-1217798 and IIS-1302212. Email: val@cis.upenn.edu.

Contents

1 Introduction

"What if analysis" is a common technique for investigating the impact of decisions on outcomes in science or business. It almost always involves a data analytics computation. Nowadays such a computation typically processes very large amounts of data and thus may be expensive to perform, especially repeatedly. An analyst is interested in exploring the computational impact of multiple *scenarios* that assume modifications of the input to the analysis problem. Our general aim is to avoid repeating expensive computations for each scenario. For a given problem, and starting from a given set of potential scenarios, we wish to perform just *one* possibly expensive computation producing a small *sketch* (i.e., a compressed representation of the input) such that the answer for any of the given scenarios can be derived rapidly from the sketch, without accessing the original (typically very large) input. We say that the sketch is "provisioned" to deal with the problem under any of the scenarios and following [17], we call the whole approach *provisioning*. Again, the goal of provisioning is to allow an analyst to efficiently explore a multitude of scenarios, using only the sketch and thus avoiding expensive recomputations for each scenario.

In this paper, we apply the provisioning approach to queries that perform *in-database analytics* [28][1]. These are queries that combine *logical* components (relational algebra and Datalog), grouping, and *numerical* components (e.g., aggregates, quantiles and linear regression). Other analytics are discussed under further work.

Abstracting away any data integration/federation, we will assume that the inputs are relational instances and that the scenarios are defined by a set of *hypotheticals*. We further assume that each hypothetical indicates the fact that certain tuples of an input instance are *retained* (other semantics for hypotheticals are discussed under further work).

A scenario consists of turning on/off each of the hypotheticals. Applying a scenario to an input instance therefore means keeping only the tuples retained by at least one of the hypotheticals that are turned on. Thus, a trivial sketch can be obtained by applying each scenario to the input, solving the problem for each such modified input and collecting the answers into the sketch. However, with k hypotheticals, there are *exponentially* (in k) many scenarios. Hence, even with a moderate number of hypotheticals, the size of the sketch could be enormous. Therefore, as part of the statement of our problem we will aim to provision a query by an algorithm that maps each (large) input instance to a *compact* (essentially size $\text{poly}(k)$) sketch.

Example. Suppose a large retailer has many and diverse sales venues (e.g., its own stores, its own web site, through multiple other stores, and through multiple other web retailers). An analyst working for the retailer is interested in learning, for each product in, say, "Electronics", a regression model for the way in which the *revenue* from the product depends on both a sales venue's *reputation* (assume a numerical score) and a sales venue *commission* (in %; 0% if own store). Moreover, the analyst wants to ignore products with small sales volume unless they have a large MSRP (manufacturer's suggested retail price). Usually there is a large (possibly distributed/federated) database that captures enough information to allow the computation of such an analytic query. For simplicity we assume in this example that the revenue for each product ID and each sales venue is in one table and thus we have the following query with a self-explanatory schema:

```
SELECT x.ProdID, LIN_REG(x.Revenue, z.Reputation, z.Commission) AS (B, A1, A2)
FROM  RevenueByProductAndVenue x
INNER JOIN Products y ON x.ProdID=y.ProdID
```

[1] In practice, the MADlib project [2] has been one of the pioneers for in-database analytics, primarily in collaboration with Greenplum DB [1]. By now, major RDBMS products such as IBM DB2, MS SQL Server, and Oracle DB already offer the ability to combine extensive analytics with SQL queries.

```
INNER JOIN SalesVenues z ON x.VenueID=z.VenueID
WHERE y.ProdCategory="Electronics" AND (x.Volume>100 OR y.MSRP>1000)
GROUP BY x.ProdID
```

The syntax for treating linear regression as a multiple-column-aggregate is simplified for illustration purposes in this example. Here the values under the attributes `B,A1,A2` denote, for each `ProdID`, the coefficients of the linear regression model that is learned, i.e., `Revenue = B + A1*Reputation + A2*Commission`.

A desirable what-if analysis for this query may involve hypotheticals such as retaining certain venue types, retaining certain venues with specific sales tax properties, retaining certain product types (within the specified category, e.g., tablets), and many others. Each of these hypotheticals can in fact be implemented as selections on one or more of the tables in the query (assuming that the schema includes the appropriate information). However, combining hypotheticals into scenarios is problematic. The hypotheticals overlap and thus cannot be separated. With 10 (say) hypotheticals there will be $2^{10} = 1024$ (in practice at least hundreds) of regression models of interest for each product. Performing a lengthy computation for each one of these models is in total very onerous. Instead, we can *provision* the what-if analysis of this query since the query in this example falls within the class covered by our positive results.

Our results. Our goal is to characterize the feasibility of provisioning with sketches of *compact* size (see Section 2 for a formal definition) for a practical class of *complex queries* that consist of a *logical* component (relational algebra or Datalog), followed by a *grouping* component, and then by a *numerical* component (aggregate/analytic) that is applied to each group (a more detailed definition is given in Section 5).

The main challenge that we address, and the part where our main contribution lies, is the design of compact provisioning schemes for numerical queries, specifically linear (ℓ_2) regression and quantiles. Together with the usual count, sum and average, these are defined in Section 4 as queries that take a set of numbers or of tuples as input and return a number or a tuple of constant width as output. It turns out that if we expect exact answers, then none of these queries can be compactly provisioned. However, we show that compact provisioning schemes indeed exist for all of them if we relax the objective to computing near-exact answers (see Section 2 for a formal definition). The following theorem summarizes our results for numerical queries (see Section 4):

Theorem 1.1 (Informal)**.** *The* quantiles, linear (ℓ_2) regression, count, and sum/average (of positive numbers) queries *can be compactly provisioned to provide (multiplicative) approximate answers to an arbitrary precision, while their exact provisioning requires the sketch size to be exponential in the number of hypotheticals.*

Our results on provisioning numerical queries can then be used for complex queries, as the following theorem summarizes (see Section 5):

Theorem 1.2 (Informal)**.** *Any complex query whose logical component is a* positive relational algebra *query can be compactly provisioned to provide an approximate answer to an arbitrary precision as long as its numerical component can be compactly provisioned for the same precision, and as long as the number of groups is not too large. On the other hand, introducing* negation *or* recursion *in the logical component requires the sketch size to be exponential in the number of hypotheticals.*

Our techniques. At a high-level, our approach for compact provisioning can be described as follows. We start by building a sub-sketch for each hypothetical by focusing solely on the retained tuples of each hypothetical individually. We then examine these sub-sketches against each other and collect additional information from the original input to summarize

the effect of appearance of other hypotheticals to each already computed sub-sketch. The first step usually involves using well-known (and properly adjusted) sampling or sketching techniques, while the second step, which is where we concentrate the bulk of our efforts, is responsible for gathering the information required for combining the sketches and specifically dealing with overlapping hypotheticals. Given a scenario, we answer the query by fetching the corresponding sub-sketches and merging them together; the result is a new sketch that act as sketch for the input consist of the *union* of the hypotheticals.

We prove our lower bounds by first identifying a central problem, i.e., the Coverage problem (see Problem 1), with provably large space requirement for any provisioning scheme, and then we use reductions from this problem to establish lower bounds for other queries of interest. The space requirement of the Coverage problem itself is proven using simple tools from information theory (see Theorem 3.1).

Comparison with existing work. Our techniques for compact provisioning share some similarities with those used in data streaming and in the distributed computation models of [14, 15, 39] (see Section 6 for further discussion and formal separations), and in particular with *linear sketching*, which corresponds to applying a linear transformation to the input data to obtain the sketch. However, due to overlap in the input, our sketches are required to to be composable with the *union* operation (instead of the *addition* operation obtained by linear sketches) and hence linear sketching techniques are not directly applicable.

Dealing with duplicates in the input (similar to the overlapping hypotheticals) has also been considered in streaming and distributed computation models (see, e.g., [10, 13]), which consider sketches that are "duplicate-resilient". Indeed, for simple queries like count, a direct application of these sketches is sufficient for compact provisioning (see Section 4.1). We also remark that the Count-Min sketch [12] can be applied to approximate quantiles even in the presence of duplication (see [10]), i.e., is duplicate-resilient. However, the approximation guarantee achieved by the Count-Min sketch for quantiles is only *additive* (i.e., $\pm\epsilon n$), in contrast to the stronger notion of *multiplicative* approximation (i.e., $(1 \pm \epsilon)$) that we achieve in this paper. To the best of our knowledge, there is no similar result concerning duplicate-resilient sketches for multiplicative approximation of quantiles or the linear regression problem, and existing techniques do not seem to be applicable for our purpose. Indeed one of the primary technical contributions of this paper is designing provisioning schemes that can effectively deal with overlapping hypotheticals for quantiles and linear regression.

Further related work. *Provisioning*, in the sense used in this paper, originated in [17] together with a proposal for how to perform it taking advantage of provenance tracking. Answering queries under hypothetical updates is studied in [5, 21] but the focus there is on using a specialized warehouse to avoid transactional costs. (See also [17] for more related work.)

Estimating the number of distinct elements (corresponding to the count query) has been studied extensively in data streams [4,7,20,31] and in certain distributed computation models [14, 15, 39]. For estimating quantiles, [12, 22, 25, 26, 33, 41] achieve an additive error of ϵn for an input of length n, and [11, 27] achieve a (stronger guarantee of) $(1 \pm \epsilon)$-approximation. Sampling and sketching techniques for ℓ_2-regression problem have also been studied in [8, 18, 19, 36] for either speeding up the computation or in data streams (see [32, 38] for excellent surveys on this topic).

2 Problem Statement

Hypotheticals. Fix a relational schema Σ. Our goal is to provision queries on Σ-instances. A *hypothetical* w.r.t. Σ is a computable function h that maps every Σ-instance I to a sub-instance

$h(I) \subseteq I$. Formalisms for specifying hypotheticals are of course of interest (e.g., apply a selection predicate to each table in I) but we do not discuss them here because the results in this paper do not depend on them.

Scenarios. We will consider analyses (scenario explorations) that start from a finite set H of hypotheticals. A *scenario* is a non-empty set of hypotheticals $S \subseteq H$. The result of applying a scenario $S = \{h_1, \ldots, h_s\}$ to an instance I is defined as a sub-instance $I|_S = h_1(I) \cup \cdots \cup h_s(I)$. In other words, under S, if any $h \in S$ is said to be turned on (similarly, any $h \in H \setminus S$ is turned off), each turned on hypothetical h will retain the tuples $h(I)$ from I.

Definition 1 (Provisioning)**.** *Given a query Q, to* provision *Q means to design a pair of algorithms: (i) a **compression** algorithm that takes as input an instance I and a set H of hypotheticals, and outputs a data structure Γ called a **sketch**, and (ii) an **extraction** algorithm that for any scenario $S \subseteq H$, outputs $Q(I|_S)$ using only Γ (without access to I).*

To be more specific, we assume the compression algorithm takes as input an instance I and k hypotheticals $h_1, \ldots, h_k$ along with the sub-instances $h_1(I), \ldots, h_k(I)$ that they define. A hypothetical will be referred to by an index from $\{1, \ldots, k\}$, and the extraction algorithm will be given scenarios in the form of sets of such indices. Hence, we will refer to a scenario $S \subseteq H$ where $S = \{h_{i_1}, \ldots, h_{i_s}\}$ by abusing the notation as $S = \{i_1, \ldots, i_s\}$. Throughout the paper, we denote by k the number of hypotheticals (i.e. $k := |H|$), and by n the size of the input instance (i.e., $n := |I|$).

We call such a pair of compression and extraction algorithms a *provisioning scheme*. The compression algorithm runs only once; the extraction algorithm runs repeatedly for all the scenarios that an analyst wishes to explore. We refer to the time that the compression algorithm requires as the *compression time*, and the time that extraction algorithm requires for each scenario as the *extraction time*.

The definition above is not useful by itself for positive results because it allows for trivial space-inefficient solutions. For example, the definition is satisfied when the sketch Γ is defined to be a copy of I itself or, as mentioned earlier, a scenario-indexed collection of all the answers. Obtaining the answer for each scenario is immediate for either case, but such a sketch can be prohibitively large as the number of tuples in I could be enormous, and the number of scenarios is exponential in $k = |H|$.

This discussion leads us to consider complexity bounds on the size of the sketches.

Definition 2 (Compact provisioning)**.** *A query Q can be* compactly *provisioned if there exists a provisioning scheme for Q that given any input instance I and any set of hypotheticals H, constructs a sketch of size* $\text{poly}(k, \log n)$ *bits, where $k := |H|$ and $n := |I|$.*

We make the following important remark about the restrictions made in Definitions 1 and 2.

Remark 2.1. *At first glance, the requirement that the input instance I cannot be examined* at all *during extraction may seem artificial, and the same might be said about the size of the sketch depending polynomially on* $\log n$ *rather than a more relaxed requirement. However, we show in Theorem 3.1 that our central lower bound result holds* even if *a portion of size $o(n)$ of the input instance can be examined during extraction* after *the scenario is revealed, and* even if *the space dependence of the sketch is only restricted to be $o(n)$ (instead of depending only polynomially on* $\log n$*). These additional properties transfer to* all *our lower bound results (i.e., Theorems 4.1, 4.7, 4.10, 5.6) although we choose not to restate them in each of them. In spite of these additional properties, the positive results we obtain (i.e., Theorems 4.3, 4.4, 4.11, 5.1) all use sketches whose space requirements depend polynomially only on* $\log n$ *and do* not *require examining the original database* at all *during the extraction. These calibration results further justify our design choices for compact provisioning.*

Even though the definition of compact provisioning does not impose any restriction on either the compression time or the extraction time, all our positive results in this paper are supported by (efficient) polynomial time algorithms. Note that this is *data-scenario complexity*: we assume the size of the query (and the schema) to be a constant but we consider dependence on the size of the instance and the number of hypotheticals. Our negative results (lower bounds on the sketch size), on the other hand, hold even when the compression and the extraction algorithms are computationally unbounded.

Exact vs. approximate provisioning. Definition 2 focused on exact answers for the queries. While this is appropriate for, e.g., relational algebra queries, as we shall see, for queries that compute numerical answers such as aggregates and analytics, having the flexibility of answering queries approximately is essential for any interesting positive result.

Definition 3 (ϵ-provisioning)**.** *For any $0 < \epsilon < 1$, a query Q can be* ϵ-provisioned *if there exists a provisioning scheme for Q, whereby for each scenario S, the extraction algorithm outputs a $(1 \pm \epsilon)$ approximation of $Q(I|_S)$, where I is the input instance.*

We say a query Q can be compactly ϵ*-provisioned if Q can be ϵ-provisioned by a provisioning scheme that, given any input instance I and any set of hypotheticals H, creates a sketch of size* $\text{poly}(k, \log n, 1/\epsilon)$ *bits.*

We emphasize that throughout this paper, we only consider the approximation guarantees which are *relative* (multiplicative) as opposed to the weaker notion of additive approximations. The precise definition of relative approximation guarantee will be provided for each query individually. Moreover, as expected, randomization will be put to good use in ϵ-provisioning. We therefore extend the definition to cover the provisioning schemes that use both randomization and approximation.

Definition 4. *For any $\epsilon, \delta > 0$, an* (ϵ, δ)-provisioning scheme *for a query Q is a provisioning scheme where both the compression and extraction algorithms are allowed to be randomized and the output* for every *scenario S is a $(1 \pm \epsilon)$-approximation of $Q(I|_S)$ with probability $1 - \delta$.*

An (ϵ, δ)-provisioning scheme is called compact *if it constructs sketches Γ of size only* $\text{poly}(k, \log n, 1/\epsilon, \log(1/\delta))$ *bits, has compression time that is* $\text{poly}(k, n, 1/\epsilon, \log(1/\delta))$*, and has extraction time that is* $\text{poly}(|\Gamma|)$*.*

In many applications, the size of the database is a very large number, and hence the exponent in the $\text{poly}(n)$-dependence of the compression time might become an issue. If the dependence of the compression time on the input size is essentially linear, i.e., $O(n) \cdot \text{poly}(k, \log n, 1/\epsilon, \log(1/\delta))$ we say that the scheme is *linear*. We emphasize that in all our positive results for queries with numerical answers we give compact (ϵ, δ)-linear provisioning schemes, thus ensuring efficiency in both running time and sketch size.

Complex queries. Our main target consists of practical queries that combine logical, grouping, and numerical components. In Section 5, we focus on *complex queries* defined by a logical (relational algebra or Datalog) query that returns a set of tuples, followed by a group-by operation (on set of grouping attributes) and further followed by numerical query that is applied to each sets of tuples resulting from the grouping. This class of queries already covers many practical examples. We observe that the output of such a complex query is a set of p tuples where p is *the number of distinct values taken by the grouping attributes*. Therefore, the size of any provisioning sketches must also depend on p. We show (in Theorem 5.1) that a sketch for a query that involves grouping can be obtained as a collection of p sketches. Hence, if each of the p sketches is of compact size (as in Definitions 2 and 4) and the value p itself is bounded by $\text{poly}(k, \log n)$, then the overall sketch for the complex query is also of compact size. Note that p is typically small for the kind of grouping used in practical analysis queries

(e.g., number of products, number of departments, number of locations, etc.). Intuitively, an analyst would have trouble making sense of an output with a large number of tuples.

Notation. For any integer $m > 0$, $[m]$ denotes the set $\{1, 2, \ldots, m\}$. The $\widetilde{O}(\cdot)$ notation suppresses $\log\log(n)$, $\log\log(1/\delta)$, $\log(1/\epsilon)$, and $\log(k)$ factors. All logarithms are in base two unless stated otherwise.

3 A "Hard" Problem for Provisioning

To establish our lower bounds in this paper, we introduce a "hard" problem called Coverage. Though not defined in the exact formalism of provisioning, the Coverage problem can be solved by many provisioning schemes using proper "reductions" and hence a lower bound for the Coverage problem can be used to establish similar lower bounds for provisioning various queries.

We start by describing the Coverage problem in details and then present our lower bound. In Appendix A, we survey some simple tools from information theory that we need in our lower bound proof.

3.1 The Coverage Problem

Informally speaking, in the Coverage problem, we are given a collection of k subsets of a universe $[n]$ and the goal is to "compress" this collection in order to answer to the questions in which indices of some subsets in the collection are provided and we need to figure out whether these subsets cover the universe $[n]$ or not. We are interested in compressing schemes for this problem that when answering each question, in addition to the already computed summary of the collection, also have a limited access to the original instance (see Remark 2.1 for motivation of this modification). The Coverage problem is defined formally as follows.

Problem 1 (Coverage)**.** *Suppose we are given a collection $\mathcal{S} = \{S_1, S_2, \ldots S_k\}$ of the subsets of $[n]$. The goal in the* Coverage *problem is to find a* compressing scheme *for $\mathcal{S}$, defined formally as a triple of algorithms:*

- *A **compression algorithm** which given the collection $\mathcal{S}$ creates a* data structure *D.*
- *An **examination algorithm** which given a subset of $[k]$, a* question*, $Q = \{i_1, \ldots, i_s\}$ and the data structure D, computes a set $J \subseteq [n]$ of indices and lookup for each $j \in J$ and each S_i ($i \in [k]$), whether $j \in S_i$ or not. The output of the examination algorithm is a tuple $S^J := (S_1^J, \ldots, S_k^J)$, where $S_i^J = S_i \cap J$.*
- *An **extraction algorithm** which given a question $\{i_1, \ldots, i_s\}$, the data structure D, and the tuple S^J, outputs "Yes", if $S_{i_1} \cup \ldots \cup S_{i_s} = [n]$ and "No" otherwise.*

We refer to the size of D, denoted by $|D|$, as the storage requirement of the compression scheme and to the size of J, denoted by $|J|$, as the examination requirement of the scheme. The above algorithms can all be randomized; in that case, we require that for *each* question Q, the final answer (of the extraction algorithm) to be correct with a probability at least 0.99. Note that this choice of constant is arbitrary and is used only to simplify the analysis; indeed, one can always amplify the probability of success by repeating the scheme constant number of times and return the majority answer.

While Coverage is not stated in the exact formalism of provisioning, the analogy between this problem and provisioning schemes should be clear. In particular, for our lower bound proofs for provisioning schemes, we can alter the Definition 1 to add an examination algorithm and allow a similar access to the original database to the provisioning scheme.

3.2 The Lower Bound

We prove the following lower bound on storage and examination requirement of any compressing scheme for the Coverage problem.

Theorem 3.1. *Any compressing scheme for the* Coverage *problem that answers each question correctly with probability at least* 0.99*, either has storage requirement or examination requirement of* $\min(2^{\Omega(k)}, \Omega(n))$ *bits.*

Allowing access to the original input in Theorem 3.1 makes the lower bound very robust. However, due to this property, the lower bound does not seem to follow from standard communication complexity lower bounds and hence we use an information-theoretic approach to prove this theorem directly, which may be of independent interest. We note that since our other lower bounds are typically proven using a reduction from the Coverage problem, the properties in Theorem 3.1 (i.e., allowing randomization and $o(n)$ access to the database *after* being given the scenario) also hold for them and we do not mention this explicitly.

In order to prove this lower bound, by Yao's minimax principle [40] (see also [34]), it suffices to define a distribution of instances of the problem and show that any *deterministic* algorithm that is correct on the input chosen according to this distribution with probability 0.99, has to have either large storage requirement or large examination requirement. We define the following distribution $\mathcal{D}$ (for simplicity assume k is even).

Hard distribution $\mathcal{D}$ for the Coverage **problem.**

1. Let $n = \binom{k}{k/2}$; pick a string $x \in \{0,1\}^n$ uniformly at random.
2. Let $\mathcal{F}$ be the family of all subsets of k with size $k/2$. Pick uniformly at random a *bijection* $\sigma : \mathcal{F} \mapsto [n]$.
3. **Embedding.** Starting from $S_i = [n]$ for all $i \in [k]$, for any $j \in [n]$, if $x_j = 0$, remove j from all sets $S_i \in \sigma^{-1}(j)$.
4. **Question.** Pick the question Q to be a uniformly at random member of $\mathcal{F}$.

The following claim is immediate.

Claim 3.2. *For any question* $Q = \{S_{i_1}, \ldots, S_{i_{k/2}}\} \in \mathcal{F}$, $S_{i_1} \cup \ldots \cup S_{i_{k/2}} = [n]$ *iff* $x_{\sigma(Q)} = 1$.

Fix any deterministic compressing scheme for the distribution $\mathcal{D}$. We define $\mathbf{D}$, $\mathbf{Q}$, and $\mathbf{J}$ as the random variables corresponding to, respectively, the data structure D, the question Q, and the examination indices J in this compressing scheme. Moreover $\mathbf{X}$ is a random variable for the input string x, $\mathbf{\Sigma}$ is for the bijection σ, and $\mathbf{S}_i$ ($i \in [k]$) is for the subset S_i. We use $\mathbf{S}^J := (\mathbf{S}^J_1, \ldots, \mathbf{S}^J_k)$ for the output of the examination algorithm.

Overview of the proof. The intuition behind the proof is as follows. By Claim 3.2, in order to give a correct answer to the question Q, the scheme has to determine the value of $x_{\sigma(Q)}$ correctly. Suppose first that the scheme only consists of compression and extraction algorithms, i.e., without the examination algorithm. In this case, even if we give the bijection σ to the extraction algorithm, the extraction algorithm has to effectively recover the value of $x_{\sigma(Q)}$ from D, where $\sigma(Q)$ is chosen uniformly at random from $[n]$. In this case, simple information-theoretic facts imply that D has to be of size $\Omega(n)$.

Now consider the other case where we remove the compression algorithm. In this case, even if we give the string x to the examination algorithm and assume that upon examining an entry j in the input, it can determine whether $\sigma(Q) = j$ or not, for the extraction algorithm

to be able to find the value of $x_{\sigma(Q)}$ correctly, it better be the case that $\sigma(Q) \in J$. In other words, the set of indices computed by the examination algorithm should contain the target index $\sigma(Q)$. However, for any fixed J of size $o(n)$, the probability that $\sigma(Q) \in J$ is $o(1)$, and hence the extraction algorithm cannot recover the correct answer with high probability.

To prove Theorem 3.1, we have to consider schemes that consist of both compression and examination algorithms and a priori it is not clear that how much the interleaved information obtained from both these algorithms can help the extraction algorithm to compute the final answer, especially considering the fact that having a compression algorithm also helps examination algorithm in finding the "correct" index. However, using a careful analysis we separate the information given from these two algorithms to the extraction algorithm and argue that at least one of the examination or compression requirements of any scheme has to be of size $\Omega(n)$.

Proof of Theorem 3.1. Suppose $\boldsymbol{\theta}$ is the random variable which is 1 if the correct answer is Yes, and is zero otherwise and $\mathbf{I}$ is the random variable that denotes the index of the string x which determines the correct answer, i.e., $\mathbf{I} = \boldsymbol{\Sigma}(\mathbf{Q})$ (by Claim 3.2). Let δ (≤ 0.01) be the probability of failure; we have,

$$\begin{aligned} H_2(\delta) &\geq H(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{Q}, \mathbf{S}^J) && \text{(Fano's inequality, Claim A.1-(5))} \\ &\geq H(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{Q}, \mathbf{S}^J, \mathbf{I}) && \text{(Conditioning reduces entropy, Claim A.1-(2))} \\ &= H(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{Q}, \mathbf{I}) - I(\boldsymbol{\theta}; \mathbf{S}^J \mid \mathbf{D}, \mathbf{Q}, \mathbf{I}) \end{aligned}$$

We bound each term in the above equation in the following two claims separately.

Claim 3.3. *Suppose $|\mathbf{D}| = o(n)$; then $H(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{Q}, \mathbf{I}) = 1 - o(1)$.*

Proof.

$$\begin{aligned} H(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{Q}, \mathbf{I}) &\geq H(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{Q}, \mathbf{I}, \boldsymbol{\Sigma}) && \text{(Conditioning reduces entropy, Claim A.1-(2))} \\ &= H(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{I}, \boldsymbol{\Sigma}) && (\mathbf{Q} \text{ is uniquely determined by } \boldsymbol{\Sigma} \text{ and } \mathbf{I}) \\ &= \sum_{i=1}^{n} \Pr(\mathbf{I} = i) \cdot H(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{I} = i, \boldsymbol{\Sigma}) \\ &= \sum_{i=1}^{n} \frac{1}{n} \cdot H(\mathbf{X}_i \mid \mathbf{D}, \mathbf{I} = i, \boldsymbol{\Sigma}) && (\mathbf{I} \text{ is uniform on } [n] \text{ and } \boldsymbol{\theta} = \mathbf{X}_i) \end{aligned}$$

Now, notice that $\boldsymbol{\Sigma}$ is independent of the event $\mathbf{I} = i$, and moreover conditioned on $\boldsymbol{\Sigma}$, $\mathbf{D}$ is a function of $\mathbf{X}$ alone and hence is independent of $\mathbf{I} = i$. Additionally, $\mathbf{X}_i$ is chosen independent of the value of $\mathbf{I} = i$; hence $\mathbf{X}_i$ is also independent of $\mathbf{I} = i$. Consequently, we can drop conditioning on $\mathbf{I} = i$ and have,

$$\begin{aligned} H(\boldsymbol{\theta} \mid \mathbf{D}, \mathbf{Q}, \mathbf{I}) &\geq \sum_{i=1}^{n} \frac{1}{n} \cdot H(\mathbf{X}_i \mid \mathbf{D}, \boldsymbol{\Sigma}) \\ &\geq \frac{H(\mathbf{X} \mid \mathbf{D}, \boldsymbol{\Sigma})}{n} && \text{(By subadditivity of entropy, Claim A.1-(4))} \\ &= \frac{H(\mathbf{X} \mid \boldsymbol{\Sigma}) - I(\mathbf{X}, \mathbf{D} \mid \boldsymbol{\Sigma})}{n} \\ &\geq \frac{H(\mathbf{X}) - H(\mathbf{D})}{n} && (\mathbf{X} \perp \boldsymbol{\Sigma} \text{ and } I(\mathbf{X}, \mathbf{D} \mid \boldsymbol{\Sigma}) \leq H(\mathbf{D} \mid \boldsymbol{\Sigma}) \leq H(\mathbf{D})) \\ &\geq 1 - \frac{|\mathbf{D}|}{n} = 1 - o(1) \end{aligned}$$

where in the last inequality we use the fact that $\mathbf{X}$ is uniformly chosen from a domain of size 2^n and hence $H(\mathbf{X}) = n$ and $H(\mathbf{D}) \leq |\mathbf{D}|$ (both by Claim A.1-(1)). ∎

Claim 3.4. *Suppose* $|\mathbf{D}| + |\mathbf{J}| = o(n)$ *; then* $I(\boldsymbol{\theta}; \mathbf{S}^J \mid \mathbf{D}, \mathbf{Q}, \mathbf{I}) \leq 0.9$.

Proof. We first define some notation. For a fixed D and Q, we use $J = \texttt{exam}(D, Q)$ to denote the unique set of examined indices (recall that the compressing scheme is deterministic over the distribution $\mathcal{D}$). For a triple $T := (D, Q, i)$ as an assignment to $(\mathbf{D}, \mathbf{Q}, \mathbf{I})$, we say that the tuple is *good* if $i \notin J$, where $J = \texttt{exam}(D, Q)$. We use the set $\mathcal{X}$ to denote the set of all valid tuples T and the set $\mathcal{X}_g$ to denote the set of all good tuples. Using this notation,

$$I(\boldsymbol{\theta}; \mathbf{S}^J \mid \mathbf{D}, \mathbf{Q}, \mathbf{I}) = \sum_{T \in \mathcal{X}} \Pr((\mathbf{D}, \mathbf{Q}, \mathbf{I}) = T) \cdot I(\boldsymbol{\theta}; \mathbf{S}^J \mid (\mathbf{D}, \mathbf{Q}, \mathbf{I}) = T)$$

We decompose the above summands into two parts; one over $\mathcal{X}_g$ and one over $\mathcal{X} \setminus \mathcal{X}_g$. We first argue that for a tuple $T \in \mathcal{X}_g$, $I(\boldsymbol{\theta}; \mathbf{S}^J \mid (\mathbf{D}, \mathbf{Q}, \mathbf{I}) = T) = 0$. This is because conditioned on $(\mathbf{D}, \mathbf{Q}, \mathbf{I}) = (D, Q, i)$, by Claim 3.2, $\boldsymbol{\theta} = \mathbf{X}_i$ and $\mathbf{X}_i$ is independent of $\mathbf{X} \setminus \mathbf{X}_i$ and $\boldsymbol{\Sigma}$, and hence $\mathbf{X}_i$ is independent of $\mathbf{S}^J$ as well whenever $i \notin J$. For a tuple $T \notin \mathcal{X}_g$, we simply use the upper bound $I(\boldsymbol{\theta}; \mathbf{S}^J \mid (\mathbf{D}, \mathbf{Q}, \mathbf{I}) = T) \leq H(\boldsymbol{\theta} \mid (\mathbf{D}, \mathbf{Q}, \mathbf{I}) = T) \leq |\boldsymbol{\theta}|$, where $|\boldsymbol{\theta}| = 1$. Consequently,

$$\begin{aligned}
I(\boldsymbol{\theta}; \mathbf{S}^J \mid \mathbf{D}, \mathbf{Q}, \mathbf{I}) &= \sum_{T \notin \mathcal{X}_g} \Pr((\mathbf{D}, \mathbf{Q}, \mathbf{I}) = T) \cdot I(\boldsymbol{\theta}; \mathbf{S}^J \mid (\mathbf{D}, \mathbf{Q}, \mathbf{I}) = T) \\
&\leq \sum_{T \notin \mathcal{X}_g} \Pr((\mathbf{D}, \mathbf{Q}, \mathbf{I}) = T) \cdot |\boldsymbol{\theta}| \\
&= \Pr((\mathbf{D}, \mathbf{Q}, \mathbf{I}) \notin \mathcal{X}_g) && (|\boldsymbol{\theta}| = 1)
\end{aligned}$$

We now show that $\Pr((\mathbf{D}, \mathbf{Q}, \mathbf{I}) \notin \mathcal{X}_g) \leq 0.9$. Let μ be the distribution of $\mathbf{I}$ (i.e., a uniform distribution on $[n]$) and for any D and Q, $\nu_{D,Q}$ be the distribution of $\mathbf{I} \mid \mathbf{D} = D, \mathbf{Q} = Q$. Fix any set J of size $o(n)$; it is clear that under μ, $\Pr_\mu(\mathbf{I} \in J) = o(1)$. We show that the total variation distance of μ and $\nu_{D,Q}$ (for "typical" choices of D and Q) is bounded away from 1 and hence $\Pr_{\nu_{D,Q}}(\mathbf{I} \in J) < 1$ as well (using Claim A.4). To achieve the bound on the total variation distance, we instead bound the KL-divergence of $\nu_{D,Q}$ from the uniform distribution μ (in expectation over the choice of D and Q) and then use Pinsker's inequality (Claim A.3) to bound the total variation distance between these distributions. We have,

$$\begin{aligned}
\mathrm{E}_{D,Q}\left[\mathrm{D}(\nu_{D,Q} \parallel \mu)\right] &= I(I; D, Q) = H(\mathbf{I}) - H(\mathbf{I} \mid \mathbf{D}, \mathbf{Q}) && \text{(by Claim A.2)} \\
&= \log n - \sum_{Q \in \mathcal{F}} \Pr(\mathbf{Q} = Q) \cdot H(\mathbf{I} \mid \mathbf{D}, \mathbf{Q} = Q) \\
&= \log n - \sum_{Q \in \mathcal{F}} \frac{1}{n} \cdot H(\boldsymbol{\Sigma}(Q) \mid \mathbf{D}) \\
& && (I = \boldsymbol{\Sigma}(Q) \text{ and } \boldsymbol{\Sigma}(Q) \perp \mathbf{Q} = Q, \mathbf{D} \perp \mathbf{Q} = Q) \\
&\leq \log n - \frac{H(\boldsymbol{\Sigma} \mid \mathbf{D})}{n} && \text{(By subadditivity of entropy, Claim A.1-(4))} \\
&\leq \log n - \frac{|\boldsymbol{\Sigma}| - |\mathbf{D}|}{n} && (H(\boldsymbol{\Sigma} \mid \mathbf{D}) \geq |\boldsymbol{\Sigma}| - |\mathbf{D}|, \text{ Claim A.1-(1)}) \\
&= \log n - \frac{\log(n!) - o(n)}{n} && (|\boldsymbol{\Sigma}| = \log n!, |\mathbf{D}| = o(n)) \\
&\leq \log n - \frac{n \log n - n \log e + O(\log n) - o(n)}{n} \\
& && \text{(by Stirling approximation of } n!) \\
&= \log e + o(1)
\end{aligned}$$

We now have,

$$\begin{aligned}
\mathrm{E}_{D,Q}\left[|\nu_{D,Q}-\mu|\right] &\le \mathrm{E}_{D,Q}\left[\sqrt{\frac{1}{2}\cdot \mathrm{D}(\nu_{D,Q}\parallel \mu)}\right] && \text{(Pinsker's inequality, Claim A.3)}\\
&\le \sqrt{\frac{1}{2}\cdot \mathrm{E}_{D,Q}\left[\mathrm{D}(\nu_{D,Q}\parallel \mu)\right]} && \text{(Convexity of } \sqrt{\cdot}\text{)}\\
&\le \sqrt{\frac{\log e}{2}+o(1)} < 0.85
\end{aligned}$$

Fix any pair (D,Q), by Claim A.4,

$$\Pr_{\nu_{D,Q}}\left(i\in \texttt{exam}(D,Q)\right) \le \Pr_{\mu}(i\in \texttt{exam}(D,Q)) + |\nu_{D,Q}-\mu|$$

By taking expectation over (D,Q),

$$\begin{aligned}
\mathrm{E}_{D,Q}\left[\Pr_{\nu_{D,Q}}\left(i\in \texttt{exam}(D,Q)\right)\right] &\le \mathrm{E}_{D,Q}\left[\Pr_{\mu}(i\in \texttt{exam}(D,Q))\right] + \mathrm{E}_{D,Q}\left[|\nu_{D,Q}-\mu|\right]\\
&\le o(1)+0.85 \le 0.9
\end{aligned}$$

Noting that the LHS in the above equation is equal $\Pr((\mathbf{D},\mathbf{Q},\mathbf{I})\notin \mathcal{X}_g)$ finalizes the proof. ∎

By plugging in the values from the above two claims, we have

$$H_2(\delta) \ge H(\boldsymbol{\theta}\mid \mathbf{D},\mathbf{Q},\mathbf{I}) - I(\boldsymbol{\theta};\mathbf{S}^J\mid \mathbf{D},\mathbf{Q},\mathbf{I}) \ge 1-o(1)-0.9 = 0.1-o(1)$$

which is in contradiction with the fixed value of $H_2(\delta)\sim 0.08$ (for $\delta=0.01$). Hence, $|\mathbf{D}|+|\mathbf{J}|=\Omega(n)$, implying that the storage requirement or examination requirement of the scheme has to be of size $\Omega(n)$. The bound of $2^{\Omega(k)}$ also follows from the fact that $n=2^{\Omega(k)}$ in this construction.

∎

Notice that in the proof of Theorem 3.1, the constructed instances have a simple property (the union in each question is either $[n]$ or $[n]\setminus\{i\}$ for some i). We extract this useful property and provide the following stronger version of Theorem 3.1 as a corollary (to its proof). This stronger version is used to prove a lower bound for the average query in Theorem 4.1.

Corollary 3.5. *Suppose the given sets $S_1,\ldots,S_k$ in the* Coverage *problem are promised to have the property that for any set $\{i_1,\ldots,i_{\lfloor k/2\rfloor}\}\subseteq[k]$, $S_{i_1}\cup\ldots\cup S_{i_{\lfloor k/2\rfloor}}$ is either $[n]$ or $[n]\setminus\{j\}$ for some $j\in[n]$. The lower bound of Theorem 3.1 still holds for this promised version of the* Coverage *problem.*

4 Numerical Queries

In this section, we study provisioning of numerical queries, i.e., queries that output some (rational) number(s) given a set of tuples. In particular, we investigate aggregation queries including count, sum, average, and quantiles (therefore min, max, median, rank, and percentile), and as a first step towards provisioning database-supported machine learning, linear (ℓ_2) regression. We assume that the relevant attribute values are rational numbers of the form a/b where both a,b are integers in range $[-W,W]$ for some $W>0$.

4.1 The Count, Sum, and Average Queries

In this section, we study provisioning of the count, sum, and average queries, formally defined as follows. The answer to the count query is the number of tuples in the input instance. For the other two queries, we assume a relational schema with a binary relation containing two

attributes: an *identifier (key)* and a *weight*. We say that a tuple x is smaller than the tuple y, if the weight of x is smaller than the weight of y. Given an instance I, the answer to the sum query (resp. the average query) is the *total weights* of the tuples (resp. the *average weight* of the tuples) in I.

We first show that none of the count, sum, average queries can be provisioned both compactly and exactly, which motivates the ϵ-provisioning approach, and then briefly describe how to build a compact (ϵ, δ)-linear provisioning scheme for each of them. Our lower bound results for count, sum, and average queries are summarized in the following theorem.

Theorem 4.1. *Exact provisioning of any of the* count, sum, *or* average *queries requires sketches of size* $\min(2^{\Omega(k)}, \Omega(n))$ *bits.*

Proof. We provide a proof for each of the queries separately.

Count query. Given $\{S_1, \ldots, S_k\}$, where each S_i is a subset of $[n]$, we solve Coverage using a provisioning scheme for the count query. Define an instance I of a relational schema with a unary relation A, where $I = \{A(x)\}_{x \in [n]}$. Define a set H of k hypotheticals, where for any $i \in [k]$, $h_i(I) = \{A(x)\}_{x \in S_i}$. For any scenario $S = \{i_1, \ldots, i_s\}$, the count of $I|_S$ is n iff $S_{i_1} \cup \ldots \cup S_{i_s} = [n]$. Hence, any provisioning scheme for the count query solves the Coverage problem and the lower bound follows from Theorem 3.1.

Sum query. The lower bound of the sum follows immediately from the one for count by setting all weights to be 1.

Average query. For simplicity, in the following construction we will omit the id attribute of the tuples as the weights of the tuples are distinct and can be used to identify each tuple.

Given $\mathcal{S} = \{S_1, S_2, \ldots S_k\}$, where each S_i is a subset of $[n]$, with the promise that for any set $\{i_1, \ldots, i_{\lfloor k/2 \rfloor}\} \subseteq [k]$, $S_{i_1} \cup \ldots \cup S_{i_{\lfloor k/2 \rfloor}}$ is either $[n]$ or $[n] \setminus \{j\}$ for some $j \in [n]$, we want to solve this restricted Coverage problem. By Corollary 3.5, the restricted Coverage problem also needs a data structure of size $\min(\Omega(n), 2^{\Omega(k)})$ bits.

Define an instance I of the relational schema with a unary relation A, where $I = \{A(x)\}_{x \in [n]}$. Define a set H of k hypotheticals, where for any $i \in [k]$, $h_i(I) = \{A(x)\}_{x \in S_i}$. For any scenario $S = \{i_1, \ldots, i_{\lfloor k/2 \rfloor}\}$, the average weight of $I|_S$ is $\frac{n+1}{2}$ (resp. $\frac{n(n+1)/2 - j}{n-1}$ for some $j \in [n]$) if $S_{i_1} \cup \ldots \cup S_{i_{\lfloor k/2 \rfloor}}$ is equal to (resp. not equal to) $[n]$. The two values are equal iff $j = (n+1)/2$, which, if we assume n is even, could never happen. Therefore knowing the average value is enough to solve the restricted Coverage problem, and the lower bound follows. ∎

We further point out that if the weights can be both positive and negative, the sum (and average) cannot be compactly provisioned even approximately, and hence we will focus on ϵ-provisioning for *positive* weights.

Theorem 4.2. *Provisioning of the* sum *(and* average*) query approximately (up to any multiplicative factor) over the input instances with both positive and negative weights requires sketches of size* $\min(2^{\Omega(k)}, \Omega(n))$ *bits.*

Proof. We use a reduction from the Coverage problem. Suppose we are given a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots S_k\}$, where each S_i is a subset of $[n]$ in the Coverage problem. Consider the relational schema $\Sigma = \{A\}$ where A is binary and the second attribute of A is the weight. Let $I = \{A(1,1), A(2,1), \ldots, A(n,1), A(n+1,-n)\}$. We define k hypotheticals: for any $i \in [k]$, $h_i(I) = \{A(j,1) \mid j \in S_i\} \cup \{A(n+1,-n)\}$. For any set $\{i_1, \ldots, i_s\} \subseteq [k]$, let S be the scenario consisting of $h_{i_1}, \ldots, h_{i_s}$; then the total weight of $I|_S$ is zero iff $S_{i_1} \cup \ldots \cup S_{i_s} = [n]$. Therefore, any multiplicative approximation to the sum query would distinguish between the zero and non-zero cases, and hence solves the Coverage problem. The lower bound on the size of the sketch now follows from Theorem 3.1. ∎

We conclude this section by explaining the ϵ-provisioning schemes for the count, sum, and average queries. Formally,

Theorem 4.3 (ϵ-provisioning count)**.** *For any $\epsilon, \delta > 0$, there exists a compact (ϵ, δ)-linear provisioning scheme for the* count *query that creates a sketch of size $\widetilde{O}(\epsilon^{-2}k(k + \log(1/\delta)))$ bits.*

Theorem 4.4 (ϵ-provisioning sum & average)**.** *For instances with positive weights, for any $\epsilon, \delta > 0$, there exists compact (ϵ, δ)-linear provisioning schemes for the* sum *and* average *queries, with a sketch of size $\widetilde{O}(\epsilon^{-2}k^2 \log(n) \log(1/\delta) + k \log \log W)$ bits.*

We remark that the results in Theorems 4.3 and 4.4 are mostly direct application of known techniques and are presented here only for completeness.

The count query can be provisioned by using *linear sketches* for estimating the ℓ_0-norm (see, e.g., [31]) as follows. Consider each hypothetical $h_i(I)$ as an n-dimensional boolean vector $\mathbf{x}_i$, where the j-th entry is 1 iff the j-th tuple in I belongs to $h_i(I)$. For each $\mathbf{x}_i$, create a linear sketch (using $\widetilde{O}(\epsilon^{-2} \log n)$ bits of space) that estimates the ℓ_0-norm [31]. Given any scenario S, combine (i.e., add together) the linear sketches of the hypotheticals in S and use the combined sketch to estimate the ℓ_0-norm (which is equal to the answer of count).

Remark 4.5. *Note that we can directly use linear sketching for provisioning the* count *query since counting the duplicates once (as done by union) or multiple times (as done by addition) does not change the answer. However, this is* not *the case for other queries of interest like* quantiles *and* regression *and hence linear sketching is not directly applicable for them.*

Here, we also describe a self-contained approach for ϵ-provisioning the count query with a slightly better dependence on the parameter n ($\log \log n$ instead of $\log n$).

We use the following fact developed by [7] in the streaming model of computation to design our scheme. For a bit string $s \in \{0,1\}^+$, denote by $\textsc{trail}(s)$ the number of trailing 0's in s. Given a list of integers $A = (a_1, \ldots, a_n)$ from the universe $[m]$, a function $g : [m] \to [m]$, and an integer $t > 0$, the $\langle t, g \rangle$-$\textsc{trail}$ of A is defined as the list of the t smallest $\textsc{trail}(g(a_i))$ values (use binary expression of $g(a_i)$), where the duplicate elements in A are counted only once.

Lemma 4.6 ([7])**.** *Given a list $A = (a_1, \ldots, a_n)$, $a_i \in [m]$ with F_0 distinct elements, pick a* random pairwise independent hash function $g : [m] \to [m]$*, and let $t = \lceil 256\epsilon^{-2} \rceil$. If r is the largest value in the $\langle t, g \rangle$-$\textsc{trail}$ of A and $F_0 \geq t$, then with probability at least $1/2$, $t \cdot 2^r$ is a $(1 \pm \epsilon)$ approximation of F_0.*

We now define our (ϵ, δ)-linear provisioning scheme for the count query.

Compression algorithm for the count **query.** Given an input instance I, a set H of hypotheticals, and an $\epsilon > 0$:

1. Assign each tuple in I with a unique number (an identifier) from the set $[n]$.
2. Let $t = \lceil 256\epsilon^{-2} \rceil$. Pick $\lceil k + \log(1/\delta) \rceil$ random pairwise independent hash functions $\{g_j : [n] \to [n]\}_{i=1}^{\lceil k + \log(1/\delta) \rceil}$. For each hash function g_j, create a sub-sketch as follows.
   (a) Compute the $\langle t, g_j \rangle$-$\textsc{trail}$ over the identifiers of the tuples in each $h_i(I)$.
   (b) Assign a new identifier to any tuple that accounts for at least one value in the $\langle t, g_j \rangle$-$\textsc{trail}$ of any h_i, called the *concise identifier*.
   (c) For each hypothetical h_i, **record** each value in the $\langle t, g_j \rangle$-$\textsc{trail}$ along with the concise identifier of the tuple that accounts for it.

Extraction algorithm for the count **query.** Given a scenario S, for each hash function g_j, we use the concise identifiers to compute the union of the $\langle t, g_j \rangle$-$\textsc{trail}$ of the hypotheticals that are turned on by S. Let r be the t-th smallest value in this union, and compute $t \cdot 2^r$. Output the median of these $t \cdot 2^r$ values among all the hash functions.

We call a sketch created by the above compression algorithm a CNT-Sketch. For each hash function g_j and each $h_i(I)$, we record concise identifiers and the number of trailing 0's ($O(\log\log n)$ bits each) of at most t tuples. Since at most $t \cdot k$ tuples will be assigned with a concise identifier, $O(\log(t \cdot k))$ bits suffice for describing each concise identifier. Hence the total size of a CNT-Sketch is $\widetilde{O}(\epsilon^{-2}k \cdot (k + \log(1/\delta)))$ bits. We now prove the correctness of this scheme.

Proof of Theorem 4.3. Fix a scenario S; for any picked hash function g_i, since the t-th smallest value of the union of the recorded $\langle t, g_i\rangle$-trail, r, is equal to the t-th smallest value in the $\langle t, g_i\rangle$-trail of $I|_S$. Hence, by Lemma 4.6, with probability at least $1/2$, $t \cdot 2^r$ is a $(1 \pm \epsilon)$ approximation of $|I|_S|$. By taking the median over $\lceil k + \log(1/\delta)\rceil$ hash functions, the probability of failure is at most $\delta/2^k$. If we take union bound over all 2^k scenarios, with probability at least $1 - \delta$, all scenarios could be answered with a $(1 \pm \epsilon)$ approximation. ∎

We now state the scheme for provisioning the sum query; the schemes for the sum and the count queries together can directly provision the average query, which finalizes the proof of Theorem 4.4. We use and extend our CNT-Sketch to ϵ-provision the sum query.

Compression algorithm for the sum query. Given an instance I, a set H of hypotheticals, and two parameters $\epsilon, \delta > 0$, let $\epsilon' = \epsilon/4$, $t = \lceil \log_{1+\epsilon'}(n/\epsilon')\rceil$, and $\delta' = \frac{\delta}{k(t+1)}$.

1. Let $p = \lceil \log_{(1+\epsilon')} W\rceil$ and for any $l \in [p]$, let $\overline{w}_l = (1+\epsilon')^l$. For each $l \in [p]$, define a set of k new hypotheticals $H_l = \{h_{l,1}, h_{l,2}, \ldots, h_{l,k}\}$, where $h_{l,i}(I) \subseteq h_i(I)$ and contains the tuples whose weights are in the interval $[\overline{w}_l, \overline{w}_{l+1})$.
2. For each hypothetical h_i, let w be the largest weight of the tuples in $h_i(I)$, and find the index γ such that $\overline{w}_\gamma \le w < \overline{w}_{\gamma+1}$. **Record** γ, and discard all the tuples in $h_i(I)$ with weight less than $\overline{w}_{\gamma-t}$.[a] Consequently, all the remaining tuples of $h_i(I)$ lie in the $(t+1)$ intervals $\{[\overline{w}_l, \overline{w}_{l+1})\}_{l=\gamma-t}^{\gamma}$. We refer to this step as the *pruning step*.
3. For each l, denote by $\widehat{H}_l$ the resulting set of hypotheticals after discarding the above small weight tuples from H_l (some hypotheticals might become empty). For each of the $\widehat{H}_l$ that contains at least one non-empty hypothetical, run the compression algorithm that creates a CNT-Sketch for I and $\widehat{H}_l$, with parameters ϵ' and δ'. **Record** each created CNT-Sketch.

[a] In case $\gamma < t$, no tuple needs to be discarded.

Extraction algorithm for the sum query. Given a scenario S, for any interval $[\overline{w}_l, \overline{w}_{l+1})$ with a recorded CNT-Sketch, compute the estimate of the number of tuples in the interval, denoted by $\widetilde{n}_l$. Output the summation of the values $(\overline{w}_{l+1} \cdot \widetilde{n}_l)$, for l ranges over all the intervals $[\overline{w}_l, \overline{w}_{l+1})$ with a recorded CNT-Sketch.

We call a sketch created by the above provisioning scheme a SUM-Sketch. Since for every hypothetical we only record the $(t+1)$ non-empty intervals, by an amortized analysis, the size of the sketch is $\widetilde{O}(\epsilon^{-2}k^2\log(n)\log(1/\delta) + k\log\log W)$ bits. We now prove the correctness of this scheme.

Proof of Theorem 4.4. For now assume that we do not perform the pruning step (line (2) of the compression phase). For each interval $[\overline{w}_l, \overline{w}_{l+1})$ among the $\lceil \log_{1+\epsilon'} W\rceil$ intervals, the CNT-Sketch outputs a $(1 \pm \epsilon')$ approximation of the total number of tuples whose weight lies in the interval. Each tuple in this interval will be counted as if it has weight $\overline{w}_{l+1}$, which

is a $(1+\epsilon')$ approximation of the original tuple weight. Therefore, we can output a $(1\pm\epsilon')^2$ approximation of the correct sum.

Now consider the original SUM-Sketch with the pruning step. We need to show that the total weight of the discarded tuples is negligible. For each hypothetical h_i, we discard the tuples whose weights are less than $\overline{w}_{\gamma-t}$, while the largest weight in $h_i(I)$ is at least $\overline{w}_\gamma$. Therefore, the total weight of the discarded tuples is at most

$$n\overline{w}_{\gamma-t} \leq \frac{nw_\gamma}{(1+\epsilon')^{\lceil \log_{(1+\epsilon')}(n/\epsilon')\rceil}} \leq \epsilon'\overline{w}_\gamma$$

Since whenever h_i is turned on by a given scenario, the sum of the weights is at least $\overline{w}_\gamma$, we lose at most ϵ' fraction of the total weight by discarding those tuples from $h_i(I)$. To see why we only lose an ϵ' fraction over all the hypotheticals (instead of $\epsilon' k$), note that at most n tuples will be discarded in the whole scenario, hence the n in the inequality $n\overline{w}_{\gamma-t} \leq \epsilon'\overline{w}_\gamma$ can be amortized over all the hypotheticals. ∎

4.2 The Quantiles Query

In this section, we study provisioning of the quantiles query. We again assume a relational schema with just one binary relation containing attributes identifier and weight. For any instance I and any tuple $x \in I$, we define the *rank* of x to be the number of tuples in I that are smaller than or equal to x (in terms of the weights). The output of a quantiles query with a given parameter $\phi \in (0,1]$ on an instance I is the tuple with rank $\lceil \phi \cdot |I| \rceil$. Finally, we say a tuple x is a $(1\pm\epsilon)$-approximation of a quantiles query whose correct answer is y, iff the rank of x is a $(1\pm\epsilon)$-approximation of the rank of y.

Similar to the previous section, we first show that the quantiles query admits no compact provisioning scheme for exact answer and then provide a compact ϵ-provisioning scheme for this query.

Theorem 4.7. *Exact provisioning of the* quantiles *query* even on disjoint hypotheticals *requires sketches of size* $\min(2^{\Omega(k)}, \Omega(n))$ *bits.*

In the quantiles query, the parameter ϕ may be given either already to the compression algorithm or only to the extraction algorithm. The latter yields an immediate lower bound of $\Omega(n)$, since by varying ϕ over $(0,1]$, one can effectively "reconstruct" the original database. However, we achieve a more interesting lower bound for the case when ϕ is given at to the compression algorithm (i.e., a fixed ϕ for all scenarios, e.g., setting $\phi = 1/2$ to find the *median*). An important property of the lower bound for quantiles is that, in contrast to all other lower bounds for numerical queries in this paper, this lower bound holds even for disjoint hypotheticals[2].

Proof of Theorem 4.7. Assume we want to prove the lower bound for any provisioning scheme for answering the median query (quantiles with fixed $\phi = 1/2$).

Let $N = 2^{k-1}$. We show how to encode a bit-string of length N into a database I with $n = \Theta(N)$ tuples and a set of $k+1$ hypotheticals such that given provisioned sketch of the median query for this instance, one can recover any bit of this string with constant probability. Standard information-theoretic arguments then imply that the sketch size must have $\Omega(N) = 2^{\Omega(k)} = \Omega(n)$ bits.

For any vector $\mathbf{v} = (v_1, v_2, \ldots, v_N) \in \{0,1\}^N$, define an instance $I_\mathbf{v}$ on a relational schema with a binary relation A whose second attribute is the weight. Let $I_\mathbf{v} = L \cup M \cup R$, where

[2]All other numerical queries that we study in this paper can be compactly provisioned for exact answer, when the hypotheticals are *disjoint*.

$L = \{A(x,0)\}_{x\in[N]}$, $M = \{A(N+x, 2x+v_x)\}_{x\in[N]}$, and $R = \{A(2N+x, W)\}_{x\in[2N]}$ (where W is the largest possible value of the weight). The weights of the tuples are ordered "$L < M < R$". The answers of all the scenarios will be in M: L is the set of "padding" tuples which shifts the median towards M, and R will be divided into disjoint hypotheticals with different sizes (basically size 2^{i-1} for the i-th hypothetical) where different scenarios over such set of hypotheticals allow us to output *all* the tuples in M.

Formally, define the set H of $(k+1)$ hypotheticals, where for any $i \in [k]$, $h_i(I_{\mathbf{v}}) = \{A(2N+x, W)\}_{x\in\{2^{i-1},\ldots,2^i-1\}}$ with 2^{i-1} tuples, and $h_{k+1} = L \cup M$. For any set $S' \subseteq [k]$, consider the scenario $S = S' \cup \{k+1\}$. Let $\gamma = \lceil \sum_{i\in S} |h_i(I_{\mathbf{v}})| /2 \rceil$. It is straightforward to verify that the median tuple of $I_{\mathbf{v}}|_S$ is $A(\gamma, 2\gamma + v_\gamma)$. By varying $\sum_{i\in S} |h_i(I_{\mathbf{v}})|$ from 1 to $2N$ (using the fact that size of hypotheticals are different powers of two), any tuple in M will be outputted and the vector $\mathbf{v}$ could be reconstructed. ∎

Note that one can extend this lower bound, by using an approach similar to Theorem 3.1, to provisioning schemes that are allowed a limited access to the original database after being given the scenario (see Section 3 for more details). We omit the details of this proof.

We now turn to prove the main theorem of this section, which argue the existence of a compact scheme for ϵ-provisioning the quantiles. We emphasize that the approximation guarantee in the following theorem is *multiplicative*.

Theorem 4.8 (quantiles). *For any $\epsilon, \delta > 0$, there exists a compact (ϵ, δ)-linear provisioning scheme for the* quantiles *query that creates a sketch of size $\widetilde{O}(k\epsilon^{-3} \log n \cdot (\log(n/\delta) + k)(\log W + k))$ bits.*

We should note that in this theorem the parameter ϕ is only provided in the extraction phase[3]. Our starting point is the following simple lemma first introduced by [27].

Lemma 4.9 ([27]). *For any list of* unique *numbers $A = (a_1, \ldots, a_n)$ and parameters $\epsilon, \delta > 0$, let $t = \lceil 12\epsilon^{-2} \log(1/\delta) \rceil$; for any* target rank *$r > t$, if we independently sample each element with probability t/r, then with probability at least $1 - \delta$, the rank of the t-th smallest sampled element is a $(1 \pm \epsilon)$-approximation of r.*

The proof of Lemma 4.9 is an standard application of the Chernoff bound and the main challenge for provisioning the quantiles query comes from the fact that hypotheticals overlap. We propose the following scheme which addresses this challenge.

Compression algorithm for the quantiles **query.** Given an instance I, a set H of hypotheticals, and two parameters $\epsilon, \delta > 0$, let $\epsilon' = \epsilon/5$, $\delta' = \delta/3$, and $t = \lceil 12\epsilon'^{-2}(\log(1/\delta') + 2k + \log(n)) \rceil$.

1. Create and **record** a CNT-Sketch for I and H with parameters ϵ' and δ'.
2. Let $\{r_j = (1+\epsilon')^j\}_{j=0}^{\lceil \log_{(1+\epsilon')} n \rceil}$. For each r_j, create the following sub-sketch individually.
3. If $r_j \leq t$, for each hypothetical h_i, **record** the r_j smallest chosen tuples in $h_i(I)$. If $r_j > t$, for each hypothetical h_i, choose each tuple in $h_i(I)$ with probability t/r_j, and **record** the $\lceil (1+3\epsilon') \cdot t \rceil$ smallest tuples in a list $T_{i,j}$. For each tuple x in the resulting list $T_{i,j}$, **record** its *characteristics vector* for the set of the hypotheticals, which is a k-dimensional binary vector $(v_1, v_2, \ldots, v_k)$, with value 1 on v_l whenever $x \in h_l(I)$ and 0 elsewhere.

[3]We emphasize that we gave a lower bound for the easier case in terms of provisioning (ϕ given at compression phase and disjoint hypotheticals), and an upper bound for the harder case (ϕ given at extraction phase and overlapping hypotheticals).

Extraction algorithm for the quantiles **query.** Suppose we are given a scenario S and a parameter $\phi \in (0,1]$. In the following, the rank of a tuple always refers to its rank in the sub-instance $I|_S$.

1. Denote by $\widetilde{n}$ the output of the CNT-Sketch on S. Let $\widetilde{r} = \phi \cdot \widetilde{n}$, and find the index γ, such that $r_\gamma \leq \widetilde{r} < r_{\gamma+1}$.
2. If $r_\gamma \leq t$, among all the hypotheticals turned on by S, take the union of the recorded tuples and output the r_γ-th smallest tuple in the union.
3. If $r_\gamma > t$, from each h_i turned on by S, and each tuple x recorded in $T_{i,\gamma}$ with a characteristic vector $(v_1, v_2, \ldots, v_k)$, *collect* x iff for any $l < i$, either $v_l = 0$ or $h_l \notin S$. In other words, a tuple x recorded by h_i is taken only when among the hypotheticals that are turned on by S, i is the smallest index s.t. $x \in h_i(I)$. We will refer to this procedure as the *deduplication*. Output the t-th smallest tuple among all the tuples that are collected.

We call a sketch created by the above compression algorithm a QTL-Sketch. It is straightforward to verify that the size of QTL-Sketch is as stated in Theorem 4.8. We now prove the correctness of the above scheme.

Proof of Theorem 4.8. Given an instance I and a set H of hypotheticals , we prove that with probability at least $1-\delta$, for every scenario S and every parameter $\phi \in (0,1]$, the output for the quantiles query on $I|_S$ is a $(1 \pm \epsilon)$-approximation. Fix a scenario S and a parameter $\phi \in (0,1]$; the goal of the extraction algorithm is to return a tuple with rank in range $(1 \pm \epsilon)$ of the *queried rank* $\lceil \phi \cdot |I|_S| \rceil$. Recall that in the extraction algorithm, $\widetilde{n}$ is the output of the CNT-Sketch. Therefore, $\widetilde{r} = \phi\widetilde{n}$ is a $(1 \pm \epsilon')$ approximation of the queried rank, and r_γ with $r_\gamma < \widehat{r} \leq (1+\epsilon')r_\gamma$ is a $(1 \pm 3\epsilon')$ approximation of the rank. In the following, we argue that the tuple returned by the extraction algorithm has a rank in range $(1 \pm \epsilon') \cdot r_\gamma$, and consequently is a $(1 \pm \epsilon)$-approximation answer to the quantiles query.

If $r_\gamma \leq t$, the r_γ-th smallest tuple of $I|_S$ is the r_γ-th smallest tuple of the union of the recorded tuples $T_{i,\gamma}$ for $i \in S$. Therefore, we obtain a $(1 \pm \epsilon)$ approximation in this case.

If $r_\gamma > t$, for any $i \in S$, define $\widehat{T}_i$ to be the list of all the tuples sampled from $h_i(I)$ in the compression algorithm (instead of maintaining the $(1+3\epsilon')t$ tuples with smallest ranks). Hence $T_{i,\gamma} \subseteq \widehat{T}_i$. If we perform the deduplication procedure on the union of the tuples in $\widehat{T}_i$ for $i \in S$ and denote the resulting list T^*, then every tuple in $I|_S$ has probability exactly t/r_γ to be taken into T^* (for any tuple, only the appearance in the smallest index $h_i(I)$ could be taken). Hence, by Lemma 4.9, with probability at least $1 - \delta'/2^{k+\log(n)}$, the rank of the t-th tuple of T^* is a $(1 \pm \epsilon')$-approximation of the rank r_γ. In the following, we assume this holds.

The extraction algorithm does not have access to $\widehat{T}_i$'s. Instead, it only has access to the list $T_{i,\gamma}$, which only contains the $(1+3\epsilon')t$ tuples of $\widehat{T}_i$ with the smallest ranks. We show that with high probability, the union of $T_{i,\gamma}$ for $i \in S$, contains the first t smallest tuples from T^*. Note that for any $i \in S$, if the largest tuple x in $\widehat{T}_i \cap T^*$ is in $T_{i,\gamma}$, then all tuples in $\widehat{T}_i \cap T^*$ are also in $T_{i,\gamma}$ (e.g. the truncation happens after the tuple x). Hence, we only need to bound the probability that x is truncated, which is equivalent to the probability of the following event: more than $(1+3\epsilon')t$ tuples in $h_i(I)$ are sampled in the compression phase for the rank r_γ (with probability t/r_γ).

Let L_i be the set of all the tuples in $h_i(I)$ which are smaller than x. Rank of x is less than or equal to the rank of the t-th smallest tuple in T^*, which is upper bounded by $(1+\epsilon')r_\gamma$. Hence, $|L_i| \leq (1+\epsilon)r_\gamma$. For any $j \in h_i(I)$, define a binary random variable y_j, which is equal to 1 iff the tuple with rank j is sampled and 0 otherwise. The expected number of the tuples that are sampled from L is then $E[\sum_{j \in L} y_j] = |L| \cdot (t/r_\gamma) \leq (1+\epsilon')t$.

Using Chernoff bound, the probability that more than $(1+3\epsilon')t$ tuples from L are sampled

is at most $\frac{\delta'}{2^{2k+\log(n)}}$. If we take union bound over all the hypotheticals in S, with probability at least $1 - \frac{\delta'}{2^{k+\log(n)}}$, for all h_i, the largest tuple in $\widehat{T}_i \cap T^*$ is in $T_{i,\gamma}$, which ensures that the rank of the returned tuple is a $(1 \pm \epsilon)$-approximation of the queried rank.

Finally, since there are only n different values for $\phi \in (0, 1]$ which results in different answers, applying union bound over all these n different values of ϕ and 2^k possible scenarios, with probability at least $(1 - 2\delta')$, the output of the extraction algorithm is a $(1 + \epsilon)$ approximation of the quantiles query. Since the failure probability of creating the CNT-Sketch is at most δ', with probability $(1 - 3\delta') = (1 - \delta)$ the QTL-Sketch successes. ∎

Extensions. By simple extensions of our scheme, many variations of the quantiles query can be answered, including outputting the rank of a tuple x, the percentiles (the rank of x divided by the size of the input), or the tuple whose rank is Δ larger than x, where $\Delta > 0$ is a given parameter. As an example, for finding the rank of a tuple x, we can find the tuples with ranks approximately $\{(1+\epsilon)^l\}$, $l \in [\lceil \log_{(1+\epsilon)} n \rceil]$, using the QTL-Sketch, and among the found tuples, output the rank of the tuple whose weight is the closest to the weight of x.

4.3 The Linear Regression Query

In this section, we study provisioning of the regression query (i.e., the ℓ_2-regression problem), where the input is a matrix $\mathbf{A}_{n\times d}$ and a vector $\mathbf{b}_{n\times 1}$, and the goal is to output a vector $\mathbf{x}$ that minimizes $\|\mathbf{Ax} - \mathbf{b}\|$ ($\|\cdot\|$ stands for the ℓ_2 norm). A $(1+\epsilon)$-approximation of the regression query is a vector $\widetilde{\mathbf{x}}$ such that $\|\mathbf{A}\widetilde{\mathbf{x}} - \mathbf{b}\|$ is at most $(1+\epsilon)\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|$.

The input is specified using a relational schema Σ with a $(d+2)$-ary relation R. Given an instance I of Σ with n tuples, we interpret the projection of R onto its first d columns, the $(d+1)$-th column, and the $(d+2)$-column respectively as the matrix $\mathbf{A}$, the column vector $\mathbf{b}$, and the identifiers for the tuples in R. For simplicity, we denote $I = (\mathbf{A}, \mathbf{b})$, assume that the tuples are ordered, and use the terms the i-th tuple of I and the i-th row of $(\mathbf{A}, \mathbf{b})$ interchangeably.

Notation. For any matrix $\mathbf{M} \in \mathbb{R}^{n\times d}$, denote by $\mathbf{M}_{(i)}$ the i-th row of $\mathbf{M}$, and by $\mathbf{U}_M \in \mathbb{R}^{n\times\rho}$ (where ρ is the rank of $\mathbf{M}$) the orthonormal matrix of the column space of $\mathbf{M}$ (see [29] for more details). Given an instance $I = (\mathbf{A}, \mathbf{b})$, and k hypotheticals, we denote for each hypothetical h_i the sub-instance $h_i(I) = (\mathbf{A}_i, \mathbf{b}_i)$. For any integer i, $\mathbf{e}_i$ denotes the i-th standard basis; hence, the i-th row of $\mathbf{M}$ can be written as $\mathbf{e}_i^T\mathbf{M}$.

The following theorem shows that the regression query cannot be compactly provisioned for exact answers and hence, we will focus on ϵ-provisioning.

Theorem 4.10. *Exact provisioning of the* regression *query, even when the dimension is $d = 1$, requires sketches of size $\min(2^{\Omega(k)}, \Omega(n))$ bits*

Proof. We show how to use the regression query to check whether the sum of a list of values is equal to 0, then, the same reduction used in Theorem 4.2 will complete the proof.

Given a list of values $a_1, a_2, \ldots, a_n$, create an instance of ℓ_2-regression problem with $d = 1$ where $\mathbf{A}_{n\times 1}^T = [1, 1, \ldots, 1]$ and $\mathbf{b}_{n\times 1}^T = [a_1, a_2, \ldots, a_n]$. We will show that $\sum_{i=1}^n a_i = 0$ iff $\arg\min_x \|\mathbf{A}x - \mathbf{b}\|^2$ is 0.

$$\|\mathbf{A}x - \mathbf{b}\|^2 = \sum_{i=1}^{n}(x - a_i)^2 = nx^2 - (2\sum_{i=1}^{n} a_i)x + \sum_{i=1}^{n} a_i^2$$

This parabola reaches its minimum when $x = (\sum_{i=1}^n a_i)/n$ and the result follows. ∎

Before continuing, we remark that if hypotheticals are disjoint, the regression query admits compact provisioning for exact answers, and hence the hardness of the problem again lies on the fact that hypotheticals overlap. To see this, consider the closed form solution:

$$\mathbf{x}_{opt} = (\mathbf{A}^T\mathbf{A})^{\dagger}\mathbf{A}^T\mathbf{b} \tag{1}$$

where † denotes the *Moore-Penrose pseudo-inverse* of a matrix [29]. Let $\mathbf{A}^T = [\mathbf{A}_1^T, \mathbf{A}_2^T, \ldots, \mathbf{A}_k^T]$ and $\mathbf{b}^T = [\mathbf{b}_1^T, \mathbf{b}_2^T, \ldots, \mathbf{b}_k^T]$, where for any $i \in [k]$, the $(\mathbf{A}_i, \mathbf{b}_i)$ pair corresponds to the sub-instance of the i-th hypotheticals, i.e., $h_i(I)$. Then

$$\mathbf{A}^T\mathbf{A} = \sum_{i=1}^{k} \mathbf{A}_i^T\mathbf{A}_i \quad \mathbf{A}^T\mathbf{b} = \sum_{i=1}^{k} \mathbf{A}_i^T\mathbf{b}_i$$

Therefore, we can design a provisioning scheme that simply records the $d \times d$ matrix $\mathbf{A}_i^T\mathbf{A}_i$ and the d dimensional vector $\mathbf{A}_i^T\mathbf{b}_i$ for each hypothetical. Then, for any given scenario $S \subseteq [k]$, the extraction algorithm computes $\sum_{i\in S} \mathbf{A}_i^T\mathbf{A}_i$ and $\sum_{i\in S} \mathbf{A}_i^T\mathbf{b}_i$, and obtains $\mathbf{x}_{opt}$ by using Equation (1).

We now turn to provide a provisioning scheme for the regression query and prove the following theorem, which is the main contribution of this section.

Theorem 4.11 (regression). *For any $\epsilon, \delta > 0$, there exists a compact (ϵ, δ)-linear provisioning scheme for the* regression *query that creates a sketch of size $\widetilde{O}(\epsilon^{-1}k^3 d \log(nW)(k + \log\frac{1}{\delta}))$ bits.*

Overview. Our starting point is a non-uniform sampling based approach (originally used for speeding up the ℓ_2-regression computation [36]) which uses a small sample to accurately approximates the ℓ_2-regression problem. Since the probability of sampling a tuple (i.e., a row of the input) in this approach depends on its relative importance which can vary dramatically when input changes, this approach is not directly applicable to our setting.

Our contribution is a *two-phase sampling* based approach to achieve the desired sampling probability distribution for *any* scenario. At a high level, we first sample and record a small number of tuples from each hypothetical using the non-uniform sampling approach; then, given the scenario in the extraction phase, we re-sample from the recorded tuples of the hypotheticals presented in the scenario. Furthermore, to rescale the sampled tuples (as needed in the original approach), we obtain the exact sampling probabilities of the recorded tuples by recording their relative importance in each hypothetical. Our approach relies on a monotonicity property of the relative importance of a tuple when new tuples are added to the original input.

RowSample. We first describe the non-uniform sampling algorithm. Let $\mathcal{P} = (p_1, p_2, \ldots, p_n)$ be a probability distribution, and $r > 0$ be an integer. Sample r tuples of $I = (\mathbf{A}, \mathbf{b})$ with replacement according to the probability distribution $\mathcal{P}$. For each sample, if the j-th row of $\mathbf{A}$ is sampled for some j, rescale the row with a factor $(1/\sqrt{rp_j})$ and store it in the sampling matrix $(\widetilde{\mathbf{A}}, \widetilde{\mathbf{b}})$. In other words, if the i-th sample is the j-th row of I, then $(\widetilde{\mathbf{A}}_{(i)}, \widetilde{\mathbf{b}}_{(i)}) = (\mathbf{A}_{(j)}, \mathbf{b}_{(j)})/\sqrt{rp_j}$. We denote this procedure by RowSample$(\mathbf{A}, \mathbf{b}, \mathcal{P}, r)$, and $(\widetilde{\mathbf{A}}, \widetilde{\mathbf{b}})$ is its output. The RowSample procedure has the following property [36] (see also [18, 38] for more details on introducing the parameter β).

Lemma 4.12 ([36]). *Suppose $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, $\boldsymbol{b} \in \mathbb{R}^n$, and $\beta \in (0, 1]$; $\mathcal{P} = (p_1, p_2, \ldots, p_n)$ is a probability distribution on $[n]$, and $r > 0$ is an integer. Let $(\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{b}})$ be an output of* RowSample$(\boldsymbol{A}, \boldsymbol{b}, \mathcal{P}, r)$, *and $\widetilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\widetilde{\boldsymbol{A}}\boldsymbol{x} - \widetilde{\boldsymbol{b}}\|$.*

If for all $i \in [n]$, $p_i \geq \beta \frac{\|e_i^T\boldsymbol{U}_A\|^2}{\sum_{j=1}^{n}\|e_j^T\boldsymbol{U}_A\|^2}$, and $r = \Theta(\frac{d\log d\log(1/\delta)}{\epsilon\cdot\beta})$, then with probability at least $(1-\delta)$, $\|\boldsymbol{A}\widetilde{\boldsymbol{x}} - \boldsymbol{b}\| \leq (1+\epsilon)\min_{\boldsymbol{x}}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|$.

The value $\|e_i^T \mathbf{U}_A\|^2$, i.e the square norm of the i-th row of $\mathbf{U}_A$, is also called the *leverage score* of the i-th row of $\mathbf{A}$. One should view the leverage scores as the "relative importance" of a row for the ℓ_2-regression problem (see [32] for more details). Moreover, using the fact that columns of $\mathbf{U}_A$ are orthonormal, we have $\sum_j \|e_j^T \mathbf{U}_A\|^2 = \rho$, where ρ is the rank of $\mathbf{A}$.

We now define our provisioning scheme for the regression query, where the compression algorithm performs the first phase of sampling (samples rows from each hypothetical) and the extraction algorithm performs the second (samples from the recorded tuples).

Compression algorithm for the regression **query.** Suppose we are given an instance $I = (\mathbf{A}, \mathbf{b})$ and k hypotheticals with $h_i(I) = (\mathbf{A}_i, \mathbf{b}_i)$ $(i \in [k])$. Let $t = \Theta(\epsilon^{-1} kd \log d \cdot (k + \log(1/\delta)))$, and define for each $i \in [k]$ a probability distribution $\mathcal{P}_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,n})$ as follows. If the j-th row of $\mathbf{A}$ is the l-th row of $\mathbf{A}_i$ (they correspond to the same tuple), let $p_{i,j} = \|e_l^T \mathbf{U}_{A_i}\|^2 / \rho$, where ρ is the rank of $\mathbf{A}_i$. If $\mathbf{A}_{(j)}$ does not belong to $\mathbf{A}_i$, let $p_{i,j} = 0$. Using the fact that for every $i \in [k]$, $\mathbf{U}_{A_i}$ is an orthonormal matrix, $\sum_{j=1}^n p_{i,j} = 1$. **Record** t independently chosen random permutations of $[k]$, and for each hypothetical h_i, create a sub-sketch as follows.

1. Sample t tuples of $h_i(I)$ with replacement, according to the probability distribution $\mathcal{P}_i$.
2. For each of the sampled tuples, assuming it is the j-th tuple of I, **record** the tuple along with its sampling rate in each hypothetical, i.e., $\{p_{i',j}\}_{i' \in [k]}$.

Extraction algorithm for the regression **query.** Given a scenario $S = \{i_1, \ldots, i_s\}$, we will recover from the sketch a matrix $\widetilde{\mathbf{A}}_{t \times d}$ and a vector $\widetilde{\mathbf{b}}_{t \times 1}$. For $l = 1$ to t:

1. Pick the l-th random permutation recorded in the sketch. Let γ be the first value in this permutation that appears in S.
2. If $(\mathbf{a}, b)$ is the l-th tuple sampled by the hypothetical h_γ, which is the j-th tuple of I, let $q_j = \sum_{i \in S} p_{i,j} / |S|$, using the recorded sampling rates.
3. Let $(\widetilde{\mathbf{A}}_{(l)}, \widetilde{\mathbf{b}}_{(l)}) = (\mathbf{a}, b) / \sqrt{tq_j}$. Return $\widetilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\widetilde{\mathbf{A}}\mathbf{x} - \widetilde{\mathbf{b}}\|$ (using any standard method for solving the ℓ_2-regression problem).

We call a sketch constructed above a REG-SKETCH. In order to show the correctness of this scheme, we need the following lemma regarding the monotonicity of leverage scores.

Lemma 4.13 (Monotonicity of Leverage Scores)**.** *Let $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{B} \in \mathbb{R}^{m \times d}$ be any matrix. Define matrix $\boldsymbol{C} \in \mathbb{R}^{(n+m) \times d}$ by* appending *rows of $\boldsymbol{B}$ to $\boldsymbol{A}$, i.e., $\boldsymbol{C}^T = [\boldsymbol{A}^T, \boldsymbol{B}^T]$. For any $i \in [n]$, if L_i is the leverage score of $\boldsymbol{A}_{(i)}$ and $\widehat{L}_i$ is the leverage score of $\boldsymbol{C}_{(i)}$, then $L_i \geq \widehat{L}_i$.*

Before providing the proof, we note that Lemma 4.13 essentially claims that adding more rows to a matrix $\mathbf{A}$ can only reduce the importance of any point originally in $\mathbf{A}$. Note that this is true even when the matrix $\mathbf{C}$ is formed by arbitrarily combining rows of $\mathbf{B}$ and $\mathbf{A}$ (rather than appending at the end).

Proof of Lemma 4.13. We use the following characterization of the leverage scores (see [9] for a proof). For any matrix $\mathbf{M} \in \mathbb{R}^{n \times d}$ and $i \in [n]$, the leverage score of $\mathbf{M}_{(i)}$ is equal

$$\min_{\mathbf{M}^T \mathbf{x} = \mathbf{M}_{(i)}} \|\mathbf{x}\|^2$$

Now, fix an $i \in [n]$, and let $\mathbf{x}^* = \arg\min_{\mathbf{A}^T\mathbf{x}=\mathbf{A}_{(i)}} \|\mathbf{x}\|^2$. By the above characterization of leverage scores, $L_i = \|\mathbf{x}^*\|^2$. Denote by $\mathbf{z}$, the vector in $\mathbb{R}^{n+m}$ where $\mathbf{z}^T = [\mathbf{x}^{*T}, \mathbf{0}]$. Then,

$$\mathbf{C}^T \cdot \mathbf{z} = \left[\mathbf{A}^T, \mathbf{B}^T\right] \cdot \mathbf{z} = \mathbf{A}^T\mathbf{x}^* = \mathbf{A}_{(i)} = \mathbf{C}_{(i)}$$

where the last two equalities are by the definitions of $\mathbf{x}^*$ and $\mathbf{C}$, respectively. Therefore,

$$\widehat{L}_i = \min_{\mathbf{C}^T\mathbf{x}=\mathbf{C}_{(i)}} \|\mathbf{x}\|^2 \leq \|\mathbf{z}\|^2 = \|\mathbf{x}^*\|^2 = L_i$$

where the inequality holds because $\mathbf{z}$ satisfies $\mathbf{C}^T\mathbf{z} = \mathbf{C}_{(i)}$. ∎

Proof of Theorem 4.11. Fix a scenario S and let $I|_S = (\widehat{\mathbf{A}}, \widehat{\mathbf{b}})$. It is straightforward to verify that, for any step $l \in [t]$, q_j (in line (2) of the extraction algorithm) is the probability that the j-th tuple of I is chosen, if the j-th tuple belongs to $I|_S$. Hence, assuming $\mathcal{P}'$ is the probability distribution defined by the q_js on rows of the $I|_S$, the extraction algorithm implements RowSample$(\widehat{\mathbf{A}}, \widehat{\mathbf{b}}, \mathcal{P}', t)$.

We will show that $q_j \geq \|e_l^T\mathbf{U}_{\widehat{A}}\|^2/k\widehat{\rho}$, where the l-th row of $\widehat{\mathbf{A}}$ is the j-th row of $\mathbf{A}$, and $\widehat{\rho}$ is the rank of $\widehat{A}$. Then, by Lemma 4.12 with β set to $1/k$, with probability at least $1 - \frac{\delta}{2^k}$, $\|\widehat{\mathbf{A}}\widetilde{\mathbf{x}} - \widehat{\mathbf{b}}\|$ is at most $(1+\epsilon)\min_{\mathbf{x}} \|\widehat{\mathbf{A}}\mathbf{x} - \widehat{\mathbf{b}}\|$; hence, the returned vector $\widetilde{\mathbf{x}}$ is a $(1+\epsilon)$-approximation. Applying a union bound over all 2^k scenarios, with probability at least $(1-\delta)$, our scheme ϵ-provisions the regression query.

We now prove that $q_j \geq \|e_l^T\mathbf{U}_{\widehat{A}}\|^2/k\widehat{\rho}$. Denote by $L_{i,j}$ (resp. $L_{S,j}$) the leverage score of the j-th tuple of I in the matrix $\mathbf{A}_i$ (resp. $\widehat{\mathbf{A}}$). Further, denote by ρ_i the rank of $\mathbf{A}_i$. Consequently, $p_{i,j} = L_{i,j}/\rho_i$, and our goal is to show $q_j \geq L_{S,j}/(k\widehat{\rho})$. Pick any $i^* \in S$ where $h_{i^*}(I)$ contains the j-th tuple of I, then:

$$q_j = \frac{\sum_{i\in S} p_{i,j}}{s} \geq \frac{p_{i^*,j}}{s} \geq \frac{L_{i^*,j}}{k\rho_i} \geq \frac{L_{S,j}}{k\widehat{\rho}}$$

For the last inequality, since $\mathbf{A}_i$ is a sub-matrix of $\widehat{\mathbf{A}}$, $\rho_i \leq \widehat{\rho}$ and the leverage score decreases due to the monotonicity (Lemma 4.13).

To conclude, the probabilities will be stored with precision $1/n$ (hence stored using $O(\log n)$ bits each), and the size of the sketch is straightforward to verify. ∎

5 Complex Queries

We study the provisioning of queries that combine logical components (relational algebra and Datalog), with grouping and with the numerical queries that we studied in Section 4.

We start by defining a class of such queries and their semantics formally. For the purposes of this paper, a *complex query* is a triple $\langle Q_L; G_{\overline{A}}; Q_N\rangle$ where Q_L is a relational algebra or Datalog query that outputs some relation with attributes $\overline{AB}$ for some $\overline{B}$, $G_{\overline{A}}$ is a *group-by* operation applied on the attributes $\overline{A}$, and Q_N is a numerical query that takes as input a relation with attributes $\overline{B}$. For any input I let $P = \Pi_{\overline{A}}(Q_L(I))$ be the $\overline{A}$-relation consisting of all the distinct values of the grouping attributes. We call the size of P the *number of groups* of the complex query. For each tuple $\overline{u} \in P$, we define $\Gamma_{\overline{u}} = \{\overline{v} \mid \overline{uv} \in Q_L(I)\}$. Then, the output of the complex query $\langle Q_L; G_{\overline{A}}; Q_N\rangle$ is a set of tuples $\{\langle \overline{u}, Q_N(\Gamma_{\overline{u}})\rangle \mid \overline{u} \in P\}$.

5.1 Positive, Non-Recursive Complex Queries

In the following, we give compact provisioning results for the case where the logical component is a *positive relational algebra* (i.e., SPJU) query. It will be convenient to assume a different,

but equivalent, formalism for these logical queries, namely that of *unions of conjunctive queries* (UCQs)[4]. We review quickly the definition of UCQs. A *conjunctive query* (CQ) over a relational schema Σ is of the form $ans(\overline{x}) \ :- \ R_1(\overline{x}_1), \ldots, R_b(\overline{x}_b)$, where atoms $R_1, \ldots, R_b \in \Sigma$, and the *size* of a CQ is defined to be the number of atoms in its body (i.e., b). A union of conjunctive query (UCQ) is a finite union of some CQs whose heads have the same schema.

In the following theorem, we show that for any complex query, where the logical component is a positive relational algebra query, compact provisioning of the numerical component implies compact provisioning of the complex query itself, provided the number of groups is not too large.

Theorem 5.1. *For any complex query $\langle Q_L; G_{\overline{A}}; Q_N \rangle$ where Q_L is a* UCQ*, if the numerical component Q_N can be compactly provisioned (resp. compactly ϵ-provisioned), and if the number of groups is bounded by $\text{poly}(k, \log n)$, then the query $\langle Q_L; G_{\overline{A}}; Q_N \rangle$ can also be compactly provisioned (resp. compactly ϵ-provisioned with the same parameter ϵ).*

Proof. Suppose Q_N can be compactly provisioned (the following proof also works when Q_N can be compactly ϵ-provisioned). Let b be the maximum size of the conjunctive queries in Q_L. Given an input instance I and a set H of k hypotheticals, we define a new instance $\widehat{I} = Q_L(I)$ and a set $\widehat{H}$ of $O(k^b)$ new hypotheticals as follows. For each subset $S \subseteq [k]$ of size at most b (i.e., $|S| \leq b$), define a hypothetical $\widehat{h}_S(\widehat{I}) = Q_L(I|_S)$ (though S is not a number, we still use it as an index to refer to the hypothetical $\widehat{h}_S$).

By our definition of the semantics of complex queries, the group-by operation partitions $\widehat{I}$ and each $\widehat{h}_S$ into $p = |\Pi_{\overline{A}}(\widehat{I})|$ sets. We treat each group individually, and create a sketch for each of them. To simplify the notation, we still use $\widehat{I}$ and $\widehat{H}$ to denote respectively the portion of the new instance, and the portion of each new hypothetical that correspond to, without loss of generality, the first group. In the following, we show that a compact provisioning scheme for Q_N with input $\widehat{I}$ and $\widehat{H}$ can be adapted to compactly provision $\langle Q_L; G_{\overline{A}}; Q_N \rangle$ for the first group. Since the number of groups p is assumed to be $\text{poly}(k, \log n)$, the overall sketch size is still $\text{poly}(k, \log n)$, hence achieving compact provisioning for the complex query.

Create a sketch for Q_N with input $\widehat{I}$ and $\widehat{H}$. For any scenario $S \in [k]$ (over H), we can answer the numerical query Q_N using the scenario $\widehat{S}$ (over $\widehat{H}$) where $\widehat{S} = \{S' \mid S' \subseteq S \ \& \ |S'| \leq b\}$. To see this, we only need to show that the input to Q_N remains the same, i.e., $Q_L(I|_S)$ is equal to $\widehat{I}|_{\widehat{S}}$. Each tuple t in $Q_L(I|_S)$ can be derived using (at most) b hypotheticals. Since any subset of S with at most b hypotheticals belongs to $\widehat{S}$, the tuple t belongs to $\widehat{I}|_{\widehat{S}}$. On the other hand, each tuple t' in $\widehat{I}|_{\widehat{S}}$ belongs to some $\widehat{h}_{S'}$ where $S' \in S$, and hence, by definition of $\widehat{h}_{S'}$, the tuple t is also in $Q_L(I|_S)$. Hence, $Q_L(I|_S) = \widehat{I}|_{\widehat{S}}$.

Consequently, any compact provisioning scheme for Q_N can be adapted to a compact provisioning scheme for the query $\langle Q_L; G_{\overline{A}}; Q_N \rangle$. ∎

Theorem 5.1 further motivates our results in Section 4 for numerical queries as they can be extended to these quite practical queries. Additionally, as an immediate corollary of the proof of Theorem 5.1, we obtain that any boolean UCQ (i.e., any UCQ that outputs a boolean answer rather than a set of tuples) can be compactly provisioned.

Corollary 5.2. *Any boolean UCQ can be compactly provisioned using sketches of size $O(k^b)$ bits, where b is the maximum size of each CQ.*

Remark 5.3. *Deutch et al. [17] introduced query provisioning from a practical perspective and proposed* boolean provenance *[23, 24, 30, 37] as a way of building sketches. This technique can also be used for compactly provisioning boolean UCQs.*

[4]Although the translation of an SPJU query to a UCQ may incur an exponential size blowup [3], in this paper, query (and schema) size are assumed to be constant. In fact, in practice, SQL queries often present with unions already at top level.

Proof Sketch. Given a query Q, an instance I and a set H of hypotheticals we compute a small sketch in the form of a boolean provenance expression.

Suppose that each tuple of an instance I is annotated with a distinct provenance token. The provenance annotation of the answer to a UCQ is a monotone DNF formula Δ whose variables are these tokens. Crucially, each term of Δ has fewer than b tokens where b is the size of the largest CQ in Q. Associate a boolean variable x_i with each hypothetical $h_i \in H$, $i \in [k]$. Substitute in Δ each token annotating a tuple t with the disjunction of the x_i's such that $t \in h_i(I)$ and with false otherwise. The result Δ' is a DNF on the variables $x_1, \ldots, x_k$ such that each of its terms has at most b variables; hence the size of Δ' is $O(k^b)$. Now Δ' can be used as a sketch if the extraction algorithm sets to true the variables corresponding to the hypotheticals in the scenario and to false the other variables. ∎

Remark 5.4. *A similar approach based on rewriting boolean provenance annotations, which are now general DNFs, can be used to provision UCQ$^\neg$s under disjoint hypotheticals. The disjointness assumption insures that negation is applied only to single variables and the resulting DNF has size* $O((2k+1)^b) = O(k^b)$.

We further point out that the exponential dependence of the sketch size on the query size (implicit) in Theorem 5.1 and Corollary 5.2 cannot be avoided even for CQs.

Theorem 5.5. *There exists a boolean conjunctive query Q of size b such that provisioning Q requires sketches of size* $\min(\Omega(k^{b-1}), \Omega(n))$ *bits.*

Proof of Theorem 5.5. Consider the following boolean conjunctive query Q_{EXP} defined over a schema with a unary relation A and a $(b-1)$-ary relation B.

$$Q_{\text{EXP}} \equiv ans() \ :- \ A(x_1), A(x_2), \ldots, A(x_{b-1}), \\ B(x_1, x_2, \ldots, x_{b-1})$$

We show how to encode a bit-string of length $N := \binom{k-1}{b-1}$ into a database I with $n = \Theta(N)$ tuples and a set of k hypotheticals such that given provisioned sketch of Q_{EXP}, one can recover any bit of this string with constant probability. Standard information-theoretic arguments then imply that the sketch size must have $\Omega(N) = \Omega(k^{b-1}) = \Omega(n)$ bits.

Let $(S_1, \ldots, S_N)$ be a list of all subsets of $[k-1]$ of size $b-1$. For any vector $\mathbf{v} \in \{0,1\}^N$, define the instance $I_{\mathbf{v}} = \{A(x)\}_{x \in [k-1]} \cup \{B(S_y)\}_{v_y = 1}$ (this is slightly abusing the notation: $B(S_y) = B(x_1, \ldots, x_{b-1})$ where $\{x_1, \ldots, x_{b-1}\} = S_y$), and a set $\{h_i\}_{i \in [k]}$ of k hypotheticals, where for any $i \in [k-1]$, $h_i(I) = \{A(i)\}$ and $h_k(I) = \{B(S_y)\}_{v_y = 1}$. To compute the i-th entry of $\mathbf{v}$, we can extract the answer to the scenario $S_i \cup \{k\}$ from the sketch and output 1 iff the answer of the query is true.

To see the correctness, $v_i = 1$ iff $B(S_i) \in h_k(I)$ iff $Q_{\text{EXP}}(I|_{S_i \cup \{k\}})$ is true. ∎

Note that one can extend this lower bound, by using an approach similar to Theorem 3.1, to provisioning schemes that are allowed a limited access to the original database after being given the scenario (see Section 3 for more details). We omit the details of this proof.

5.2 Adding Negation, Recursion, or HAVING

It is natural to ask (a) if Theorem 5.1 still holds when adding *negation* or *recursion* to the query Q_L (i.e. UCQ *with negation* and *recursive Datalog*, respectively), and (b) whether or not it is possible to provision queries in which logical operations are done *after* numerical ones. A typical example of a query in part (b) is a selection on aggregate values specified by a HAVING clause. Unfortunately, the answer to both questions is negative.

We first show that the answer to the question (a) is negative, i.e., there exists a *boolean conjunctive query with negation* ($\text{CQ}^\neg$) and a recursive Datalog query which require sketches of exponential size (in k) for any provisioning scheme. Formally,

Theorem 5.6. *Exact provisioning of (i) boolean CQ with negation or (ii) recursive Datalog (even without negation) queries requires sketches of size* $\min(2^{\Omega(k)}, \Omega(n))$.

Recall that a $\text{CQ}^\neg$ is a rule of the form

$$ans() \text{ :- } L_1, \ldots L_b$$

where the L_i's are positive or negative literals (atoms or negated atoms) that is subject to *range-restriction*: every variable occurring in the rule appears in at least one positive literal. We define the size of such a query to be the number of literals.

Define the following boolean $\text{CQ}^\neg$ over a schema with two unary relation symbols, named A and B:

$$Q_{\texttt{NOTSUB}}() \text{ :- } A(x), \neg B(x)$$

This query returns true on I iff there exists some x where $A(x) \in I$ and $B(x) \notin I$. Intuitively, if we view A, B as subsets of the active domain of I, it is querying whether or not "B is a subset of A". We use a reduction from the Coverage problem to prove the lower bound for $Q_{\texttt{NOTSUB}}$.

Proof of Theorem 5.6, Part (I). Suppose we are given a collection $\{S_1, \ldots, S_k\}$ of subsets of $[n]$ and we want to solve the Coverage problem using a provisioning scheme for $Q_{\texttt{NOTSUB}}$. Create the following instance I for the schema $\Sigma = \{A, B\}$, where for any $x \in [n]$, $A(x), B(x) \in I$. Define the set of of hypotheticals $H = \{h_1, h_2, \ldots, h_{k+1}\}$, where for any $i \in [k]$, $h_i(I) = \{B(x) \mid x \in S_i\}$ and $h_{k+1}(I) = \{A(x) \mid x \in [n]\}$.

For any set $\widehat{S} = \{i_1, \ldots, i_s\} \subseteq [k]$, under the scenario $S = \widehat{S} \cup \{k+1\}$, $Q_{\texttt{NOTSUB}}(I|_S)$ is true iff there exists $x \in [n]$ s.t. $B(x) \notin I|_S$ which is equivalent to $[n] \not\subseteq S_{i_1} \cup \ldots \cup S_{i_s}$. Therefore any provisioning scheme of $Q_{\texttt{NOTSUB}}$ solves the Coverage problem, and it follows from Theorem 3.1 that the sketch of such scheme must have size $\min(2^{\Omega(k)}, \Omega(n))$ bits. ∎

While adding negation extends UCQs in one direction, adding recursion to UCQ also results in another direction. This leads to recursive Datalog (without negation). Consider the st-connectivity query, which captures whether there is a path from the vertex s to the vertex t:

$$\begin{aligned} ans() &\text{ :- } T(\texttt{t}) \\ T(y) &\text{ :- } E(x, y), T(x) \\ T(\texttt{s}) & \end{aligned}$$

To prove a lower bound for the st-connectivity query, we again use a reduction from the Coverage problem, where in our construction, the only path from s to t has n edges, and hence only when all the n edges are presented in a scenario s will be connected to t.

Proof of Theorem 5.6, Part (II). Suppose we are given a collection $\{S_1, \ldots, S_k\}$ of subsets of $[n]$ and we want to solve the Coverage problem using a provisioning scheme for st-connectivity. Consider a graph $G(V, E)$ with vertex set $V = \{(s =)v_0, v_1, v_2, \ldots, v_n (= t)\}$, and edges $E(v_{j-1}, v_j)$, for all $j \in [n]$. The edge set E of the graph G is the input I to the provisioning scheme. Define the hypotheticals $H = \{h_1, h_2, \ldots, h_k\}$ where $h_i(I) = \{E(v_{j-1}, v_j)\}_{j \in S_i}$, for all $i \in [k]$.

For any scenario $S = \{i_1, \ldots, i_s\} \subseteq [k]$, $T(t)$ is true (i.e., s is connected to t) in $I|_S$ iff all n edges are in $I|_S$, which is equivalent to $S_{i_1} \cup \ldots \cup S_{i_s} = [n]$. Therefore, any provisioning

scheme for the st-connectivity query solves the Coverage problem, and it follows from Theorem 3.1 that the sketch of such scheme must have size $\min(2^{\Omega(k)}, \Omega(n))$ bits. ∎

Showing a negative answer to the question (b) in the beginning of this section is very easy. As we already showed in Section 4 (see, e.g., Theorem 4.1), there are numerical queries that do not admit compact provisioning for *exact* answer. One can simply verify that each of those queries can act as a counter example for question (b) by considering HAVING clauses that test the equality of the answer to the numerical part against an exact answer (e.g. testing whether the answer to count is n or not).

6 Comparison With a Distributed Computation Model

Query provisioning bears some resemblance to the following distributed computation model: k sites want to jointly compute a function, where each site holds only a portion of the input. The function is computed by a *coordinator*, who receives/sends data from/to each site. The goal is to design protocols with small amount of communication between the sites and the coordinator (see [15, 39], and references therein). In what follows, we highlight some key similarities and differences between this distributed computation model and our query provisioning framework.

In principle, any protocol where data are only sent from the sites to the coordinator (i.e., *one-way* communication), can be adapted into a provisioning scheme with a sketch of size proportional to the size of the transcript of the protocol. To see this, view each hypothetical as a site and record the transcripts as the sketch. Given a scenario S, the extraction algorithm acts as the coordinator and computes the function from the transcripts of the hypotheticals in S. The protocol for counting the number of distinct elements in the distributed model introduced in [14] is an example (which in fact also uses the streaming algorithm from [7] as we do in Lemma 4.6).

However, protocols in the distributed model usually involve back and forth (*two-way*) communication between the sites and the coordinator (e.g., the algorithms of [10, 41] for estimating quantiles in the distributed model). In general, the power of the distributed computation model with two-way communication is *incomparable* to the query provisioning framework. Specifically, for k sites/hypotheticals (even when the input is partitioned, i.e., no overlaps), there are problems where a protocol with $\text{poly}(k)$-bit transcripts exists but provisioning requires $2^{\Omega(k)}$-bit sketches. Conversely, there are problems where provisioning can be done using $\text{poly}(k)$-bit sketches but any distributed protocol requires $2^{\Omega(k)}$-bit transcripts. We make these observations precise below.

Another source of difference between our techniques and the ones used in the distributed computation model is the typical assumption in the latter that the input is *partitioned* among the sites in the distributed computation model (i.e. no overlaps; see, for instance, [15, 39, 41]). This assumption is in contrast to the hypotheticals with unrestricted overlap that we handle in our query provisioning framework. A notable exception to the no-overlaps assumption is the study of quantiles (along with various other aggregates) by [10]. However, as stated by the authors, they benefit from the coordinator sharing a summary of the whole data distribution to individual sites, which requires a back and forth communication between the sites and coordinator. As such the results of [10] can not be directly translated into a provisioning scheme. We also point out that the approximation guarantee we obtain for the quantiles query is stronger than the result of [10] (i.e. a relative vs additive approximation).

Finally, we use two examples to establish an exponential separation between the minimum sketch size in the provisioning model and the minimum transcript length in the distributed computation model, proving a formal separation between the two models.

Before describing the examples, we should note that, in the following, we assume that input tuples are made distinct by adding another column which contains *unique identifiers*,

but the problems and the operations will only be defined over the part of the tuples without the identifiers. For instance, 'two sets of tuples are disjoint' means '*after removing the identifiers of the tuples,* the two sets are disjoint'.

A problem with poly-size sketches and exponential-size transcripts. The following SetDisjointness problem is well-known to require transcripts of $\Omega(N)$ bits for any protocol [6, 35]: Alice is given a set S and Bob is given a set T both from the universe $[N]$, and they want to determine, with success probability at least $2/3$, whether S and T are disjoint. Similarly, if each of the k sites is given a set and they want to determine whether all their sets are disjoint, the size of the transcript is also lower bounded by $\Omega(N)$ bits. If we let $N = 2^k$, the distributed model requires $\Omega(2^k)$ bits of communication for solving this problem.

However, to provision the SetDisjointness query (problem), we only need to record the pairs of hypotheticals whose sets are not disjoint (hence $O(k^2)$ bits). The observation is that if a collection of sets are not disjoint, there must exist two sets that are also not disjoint, which can be detected by recording the non-disjoint hypothetical pairs.

A problem with exponential-size sketches and poly-size transcripts. Consider a relational schema with two unary relations A and B. Given an instance I, we let $a = \sum_{A(x)\in I} 2^x$; then, the problem is to determine whether $B(a) \in I$ or not. Intuitively, A 'encodes' the binary representation of a value a, and the problem is to determine whether a belongs to 'the set of values in B'.

Let $n = 2^k$, and $I = \left\{A(x)_{x\in[k]}\right\} \cup \{B(y)\}_{y\in[n]}$. Using a similar construction as Lemma 3.5, one can show that provisioning this problem requires a sketch of size $2^{\Omega(k)}$, even when the input instance is guaranteed to be a subset of I (i.e., the largest value of A is k).

However, in the distributed model, there is a protocol using a transcript of size $\text{poly}(k)$ bits: every site sends its tuple in A to the coordinator; the coordinator computes $a = \sum_{A(x)} 2^x$ and sends a to every site; a site response 1 iff it contains the tuple $B(a)$, and 0 otherwise.

7 Conclusions and Future Work

In this paper, we initiated a formal framework to study compact provisioning schemes for relational algebra queries, statistics/analytics including quantiles and linear regression, and complex queries. We considered provisioning for exact as well as approximate answers, and established upper and lower bounds on the sizes of the provisioning sketches.

The queries in our study include quantiles and linear regression queries from the list of in-database analytics highlighted in [28]. This is only a first step and the study of provisioning for other core analytics problems, such as variance computation, k-means clustering, logistic regression, and support vector machines is of interest.

Another direction for future research is the study of queries in which *numerical* computations follow each other (e.g., when the linear regression training data is itself the result of aggregations). Yet another direction for future research is an extension of our model to allow other kinds of hypotheticals/scenarios as discussed in [17] that are also of practical interest. For example, an alternative natural interpretation of hypotheticals is that they represent tuples to be *deleted* rather than retained. Hence the application of a scenario $S \subseteq [k]$ to I becomes $I|_S = I \setminus (\bigcup_{i\in S} h_i(I))$. Using our lower bound techniques, one can easily show that even simple queries like count or sum cannot be approximated to within any multiplicative factor under this definition. Nevertheless, it will be interesting to identify query classes that admit compact provisioning in the delete model or alternative natural models.

References

[1] Greenplum DB (Pivotal) http://pivotal.io/big-data/pivotal-greenplum-database.

[2] The MADlib Project. http://madlib.net.

[3] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[4] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29. ACM, 1996.

[5] Andrey Balmin, Thanos Papadimitriou, and Yannis Papakonstantinou. Hypothetical queries in an OLAP environment. In *VLDB*, pages 220–231, 2000.

[6] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. In *FOCS*, pages 209–218. IEEE, 2002.

[7] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *RANDOM*. Springer, 2002.

[8] Kenneth L Clarkson and David P Woodruff. Numerical linear algebra in the streaming model. In *STOC*, pages 205–214. ACM, 2009.

[9] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. *arXiv:1408.5099*, 2014.

[10] G. Cormode, S. Muthukrishnan, and W. Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *ICDE*, pages 20–31, 2006.

[11] Graham Cormode, Flip Korn, S Muthukrishnan, and Divesh Srivastava. Space-and time-efficient deterministic algorithms for biased quantiles over data streams. In *PODS*, pages 263–272. ACM, 2006.

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

[13] Graham Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *PODS*, pages 271–282, 2005.

[14] Graham Cormode, S Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. *ACM Transactions on Algorithms (TALG)*, 7(2):21, 2011.

[15] Graham Cormode, S Muthukrishnan, Ke Yi, and Qin Zhang. Optimal sampling from distributed streams. In *PODS*, pages 77–86. ACM, 2010.

[16] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[17] Daniel Deutch, Zachary G Ives, Tova Milo, and Val Tannen. Caravan: Provisioning for what-if analysis. In *CIDR*, 2013.

[18] Petros Drineas, Michael W Mahoney, and S Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *SODA*, pages 1127–1136. ACM, 2006.

[19] Petros Drineas, Michael W Mahoney, S Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249, 2011.

[20] Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

[21] Shahram Ghandeharizadeh, Richard Hull, and Dean Jacobs. Heraclitus: Elevating deltas to be first-class citizens in a database programming language. *ACM Trans. Database Syst.*, 21(3):370–426, 1996.

[22] Anna C Gilbert, Yannis Kotidis, S Muthukrishnan, and Martin J Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *PVLDB*, pages 454–465. VLDB Endowment, 2002.

[23] T.J. Green. Containment of conjunctive queries on annotated relations. *Theory Comput. Syst.*, 49(2), 2011.

[24] T.J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.

[25] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *ACM SIGMOD Record*, volume 30, pages 58–66. ACM, 2001.

[26] Michael B Greenwald and Sanjeev Khanna. Power-conserving computation of order-statistics over sensor networks. In *PODS*, pages 275–285. ACM, 2004.

[27] Anupam Gupta and Francis X Zane. Counting inversions in lists. In *SODA*, pages 253–254. Society for Industrial and Applied Mathematics, 2003.

[28] Joseph M. Hellerstein, Christopher Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleksander Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, and Arun Kumar. The madlib analytics library or MAD skills, the SQL. *PVLDB*, 5(12):1700–1711, 2012.

[29] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[30] T. Imielinski and W. Lipski. Incomplete information in relational databases. *J. ACM*, 31(4), 1984.

[31] Daniel M Kane, Jelani Nelson, and David P Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, pages 41–52. ACM, 2010.

[32] Michael W Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[33] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *ACM SIGMOD Record*, volume 27, pages 426–435. ACM, 1998.

[34] Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. In *The Computer Science and Engineering Handbook*, pages 141–161. 1997.

[35] Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

[36] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152. IEEE, 2006.

[37] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[38] David P Woodruff. Sketching as a tool for numerical linear algebra. *arXiv:1411.4357*, 2014.

[39] David P Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *STOC*, pages 941–960. ACM, 2012.

[40] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 222–227, 1977.

[41] Ke Yi and Qin Zhang. Optimal tracking of distributed heavy hitters and quantiles. *Algorithmica*, 65(1):206–223, 2013.

A Tools from Information Theory

We use basic concepts from information theory in our lower bound proof in Section 3. For a broader introduction to the field and proofs of the claims in this section, we refer the reader to the excellent text by Cover and Thomas [16].

In the following, we denote random variables using capital bold-face letters. We denote the *Shannon Entropy* of a random variable $\mathbf{A}$ by $H(\mathbf{A})$ and the *mutual information* of two random variables $\mathbf{A}$ and $\mathbf{B}$ by $I(\mathbf{A};\mathbf{B}) = H(\mathbf{A}) - H(\mathbf{A} \mid \mathbf{B}) = H(\mathbf{B}) - H(\mathbf{B} \mid \mathbf{A})$. For a real number $0 \leq x \leq 1$, we further use: $H_2(x) = x \log \frac{1}{x} + (1-x) \log \frac{1}{1-x}$ to denote the binary entropy function. Finally, we use S_A to denote the support of the random variable $\mathbf{A}$ and $|\mathbf{A}| := \log |S_A|$.

We use the following basic properties of entropy and mutual information.

Claim A.1. *Suppose* $\mathbf{A}$, $\mathbf{B}$, *and* $\mathbf{C}$ *are random variables and* $f : S_A \mapsto S_B$ *is a function;*

1. $0 \leq H(\mathbf{A}) \leq |A|$; $H(\mathbf{A}) = |A|$ *iff* $\mathbf{A}$ *is uniformly distributed over its support.*
2. Conditioning reduces entropy, *i.e,* $H(\mathbf{A} \mid \mathbf{B}) \leq H(\mathbf{A})$.
3. $I(\mathbf{A};\mathbf{B}) \geq 0$; *equality holds iff* $\mathbf{A}$ *and* $\mathbf{B}$ *are* independent.
4. Subadditivity of entropy: $H(\mathbf{A},\mathbf{B}) \leq H(\mathbf{A}) + H(\mathbf{B})$.
5. (Fano's inequality) *For a* binary *random variable* $\mathbf{B}$, *and a function* f *that predicts* $\mathbf{B}$ *based on* $\mathbf{A}$, *if* $\Pr(f(\mathbf{A}) \neq \mathbf{B}) = \delta$, *then* $H(\mathbf{B} \mid \mathbf{A}) \leq H_2(\delta)$.
6. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *are mutually independent conditioned on* $\mathbf{C}$, *then* $I(\mathbf{A};\mathbf{B} \mid \mathbf{C}) = 0$.
7. (Data processing inequality) $I(f(A);B) \leq I(A;B)$.

For two distributions μ and ν over the same probability space, the *Kullback-Leibler* divergence between μ and ν is defined as $\mathrm{D}(\mu \parallel \nu) = \mathrm{E}_{a\sim\mu}\left[\log \frac{\Pr_\mu(a)}{\Pr_\nu(a)}\right]$. For our proofs, we need the following basic relation between KL-divergence and mutual information.

Claim A.2. *For any two random variables* $\mathbf{A}$ *and* $\mathbf{B}$, *suppose* μ *denotes the distribution of* $\mathbf{A}$ *and* ν_B *denotes the distribution of* $\mathbf{A} \mid \mathbf{B} = B$ *for any* B *in the support of* $\mathbf{B}$; *then,* $E_B\left[D(\nu_B \parallel \mu)\right] = I(\mathbf{A};\mathbf{B})$.

We denote the *total variation distance* between two distributions μ and ν over the same probability space Ω by $|\mu - \nu| = \frac{1}{2} \cdot \sum_{x\in\Omega} \left|\Pr_\mu(x) - \Pr_\nu(x)\right|$. The Pinsker's inequality upper bounds the total variation distance between two distributions based on their KL-divergence as follows.

Claim A.3 (Pinsker's inequality)**.** *For any two distributions* μ *and* ν,

$$|\mu - \nu| \leq \sqrt{\frac{1}{2} \cdot D(\mu \parallel \nu)}$$

Finally,

Claim A.4. *Suppose* μ *and* ν *are two probability distributions for a random variable* $\mathbf{A}$ *and* J *is any fixed set in the domain of* $\mathbf{A}$; *then* $\Pr_\mu(\mathbf{A} \in J) \leq \Pr_\nu(\mathbf{A} \in J) + |\mu - \nu|$.