THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Querying Incomplete Numerical Data: Between Certain and Possible Answers

Citation for published version:
Console, M, Libkin, L & Peterfreund, L 2023, Querying Incomplete Numerical Data: Between Certain and Possible Answers. in Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23). PODS '23, ACM Association for Computing Machinery, pp. 349-358, 42nd ACM Symposium on Principles of Database Systems , Seattle , Washington, United States, 18/06/23. https://doi.org/10.1145/3584372.3588660

Digital Object Identifier (DOI):
10.1145/3584372.3588660

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

OPEN ACCESS

Download date: 30. Apr. 2024

Querying Incomplete Numerical Data: Between Certain and Possible Answers

Marco Console
Sapienza, University of Rome
Rome, Italy
console@diag.uniroma1.it

Leonid Libkin
The University of Edinburgh
Edinburgh, United Kingdom
l@libk.in

Liat Peterfreund
LIGM, Université Gustave Eiffel, CNRS
Champs-sur-Marne, France
liat.peterfreund@univ-eiffel.fr

ABSTRACT

Queries with aggregation and arithmetic operations, as well as incomplete data, are common in real-world database, but we lack a good understanding of how they should interact. On the one hand, systems based on SQL provide ad-hoc rules for numerical nulls, on the other, theoretical research largely concentrates on the standard notions of certain and possible answers. In the presence of numerical attributes and aggregates, however, these answers are often meaningless, returning either too little or too much. Our goal is to define a principled framework for databases with numerical nulls and answering queries with arithmetic and aggregations over them.

Towards this goal, we assume that missing values in numerical attributes are given by probability distributions associated with marked nulls. This yields a model of probabilistic bag databases in which tuples are not necessarily independent since nulls can repeat. We provide a general compositional framework for query answering and then concentrate on queries that resemble standard SQL with arithmetic and aggregation. We show that these queries are measurable, and their outputs have a finite representation. Moreover, since the classical forms of answers provide little information in the numerical setting, we look at the probability that numerical values in output tuples belong to specific intervals. Even though their exact computation is intractable, we show efficient approximation algorithms to compute such probabilities.

CCS CONCEPTS

• **Theory of computation** → *Logic*; **Incomplete, inconsistent, and uncertain databases**; *Logic and databases*; • **Mathematics of computing** → *Probabilistic algorithms*; *Probabilistic representations*; • **Information systems** → **Incomplete data**; **Relational database model**.

KEYWORDS

Nulls; numerical attributes; aggregate queries; probabilistic databases; approximations; certain and possible answers

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PODS '23, June 18–23, 2023, Seattle, WA, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0127-6/23/06…$15.00
https://doi.org/10.1145/3584372.3588660

ACM Reference Format:
Marco Console, Leonid Libkin, and Liat Peterfreund. 2023. Querying Incomplete Numerical Data: Between Certain and Possible Answers. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23), June 18–23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3584372.3588660

1 INTRODUCTION

Handling incomplete data is a subject of key importance in databases, but the practical side of it has many well-known deficiencies [13–15], and the state-of-the-art in theoretical research has a rather narrow focus in terms of its applicability (see, e.g., [9] for a recent survey). In particular, systems based on SQL follow a rather specific approach to handling incomplete data where all types of incompleteness are replaced with a single null value, and specific rules are applied to those nulls depending on their usage (e.g., 3-valued logic in selection conditions with *true* tuples retained; 3-valued logic in constraint conditions with *non-false* constraints satisfied, syntactic equality of nulls for grouping and set operations). With these rules frequently leading to undesired behaviors, many practical applications attempt to impute incomplete data in a statistically meaningful way, i.e., resembling the distribution of other data values in a database [18]. While standard query evaluation can be used on the now complete data, the knowledge that the original data was incomplete is lost for the users, and they may take the answers as correct without the healthy dose of scepticism they deserve.

There is little that database research can offer to mitigate this problem, especially for numerical attribute values where imputation is most commonly used. For them, typical queries use arithmetic operations, aggregate functions, and numerical constraints. In fact, queries of this kind are very common in practice and form the absolute majority of standard benchmarks for database systems (e.g., TPC-H or TPC-DS). For these queries, though, we have little theoretical knowledge to rely on.

To explain why this is so, we observe that most of the existing theoretical research on answering queries over incomplete data concentrates on the notions of *certain* and *possible* answers. The former is often viewed as the holy grail of query answering, while the latter is often a fallback position in case we do not have enough certain answers to show. Both possible and certain answers, however, are often meaningless for queries with arithmetic and aggregation. Consider, for example, the following SQL query q

```
SELECT A, SUM(B) FROM R GROUP BY A WHERE B ≥ 2
```

on a relation $R(A, B)$ with tuples $(1, 1)$ and $(1, \texttt{Null})$, where `Null` denotes the SQL null value. If all we know about the attribute B is that it is an integer, then no answer is certain for q (in fact, as long as there is any uncertainty about the value that `Null` may take, the certain answer is empty). As for possible answers, then *every* tuple

$(1, c)$ for $2 \leq c \in \mathbb{N}$ is a possible answer, in cases where Null is interpreted as c (since the tuple $(1, 1)$ is excluded due to the WHERE clause). Hence, the usefulness of such answers is limited. A recent attempt to provide proper substitutes for these notions was made in [41] but the shape of answers presented is quite complex for the user to comprehend. Another approach, specifically focused on numerical attributes, was presented in [10], but it does not take into account aggregation and prior knowledge on the missing data. As for imputation, it would choose some value for Null, say 2 based on the other data in the database, and then simply return the tuple $(1, 2)$ as the definitive truth – without any hint to the user as to the doubts of the veracity of the answer.

But what about mixing the probabilistic approach of imputation with the classical certain/possible answers approach? For example, suppose that, in the above example, the possible values for Null are distributed according to the normal distribution with mean 2 and standard deviation 0.5. Then, with probability approximately 0.682, the value of the SUM aggregate will fall into the $[2.5, 3.5]$ interval. Unlike returning the value $(1, 3)$, giving no hint that data was incomplete in first place, this new type of answer makes it very clear that our knowledge is only partial. Also, it provides much more information than an empty certain answer (nothing is certain) and an infinite possible answer (absolutely everything is possible)

The goal of this paper is to define a model of incomplete numerical data and a framework for query answering that naturally support answers like the one above. As a first theoretical model of this kind, our approach will not cover the entire SQL Standard and all the numerical attributes in it. However, we shall cover a significant portion of the query language (essentially, the usual SELECT-FROM-WHERE-GROUP BY-HAVING queries) and use the real numbers $\mathbb{R}$ as an abstraction for numerical attributes, for the ease of reasoning about probability distributions.

Our first task will be to define the data model. The example above hints that we want to treat nulls as random variables, according to some distribution. In our model, for generality, we allow a real-valued random variable for each null, though, in a more down-to-earth approach, we could associate them with individual attributes (say, the height of a person that is usually assumed to follow a normal distribution [39]). Thus, unlike the usual probabilistic databases [38], our model deals with *uncountably infinite* probability spaces. Such databases were the focus of recent studies [8, 21, 22], which mainly focused on the crucial issue of measurability of query mappings. Those papers, however, left open a crucial issue: how to construct suitable probabilistic databases from real-world incomplete numerical data. As our first task, we fill this gap and provide such a model. Our construction is general enough so that queries could be viewed as measurable maps between probability spaces given by infinite probabilistic databases.

Then, we instantiate our framework to the case of databases with marked (or repeating) nulls [2, 27]. The fact that nulls can repeat in different tuples will take us out of the realm of tuple-independent probabilistic databases that have dominated the field of probabilistic data so far. In fact, the need for a model of "tuple-dependent" databases even transcends our immediate goal. Consider, for example, the query

```
SELECT * FROM R JOIN S ON R.A = S.A
```

with $R(A, B)$ having one fact $(1, \texttt{Null})$ and $S(A, C)$ containing $(1, 2)$ and $(1, 3)$. The end result has tuples $(1, \texttt{Null}, 2)$ and $(1, \texttt{Null}, 3)$. Since Null is defined by the same random variable in both tuples, the probability that a tuple, say $(1, 1, 2)$, is in the answer depends on the prior knowledge that another tuple, say $(1, 1, 3)$, is in the answer. Thus, if we want to have a compositional query language, i.e., the ability to query the results of other queries (taken for granted in languages like SQL), then we cannot rely on the tuple-independent model, as query results will be not be tuple-independent.

We next instantiate a query language. Following the long line of work on algebras with aggregates [23, 25, 30, 33, 34] we provide a simple yet powerful extension of relational algebra that captures key features of SQL queries such as arithmetic and aggregation. We then show that answers for these queries define measurable sets in our probabilistic model. Thus, a meaningful form of answers for queries in our extended algebra, and SQL, can be defined in our framework.

With that, we move to query answering. As hinted above, we first look at computing the probability that values in the output tuples belong to given intervals. We formulate this intuition as a computation problem as well as the corresponding decision problem, and study their complexity. For the computation problem, we prove that results are often transcendental, and even for simple distributions, that are guaranteed to yield rational numbers, the problem is ♯P-hard [3]. For the computation problem, we show that it is NP-hard.

This means that one needs to resort to approximations. We first show that the standard FPRAS (Fully Polynomial Randomized Approximation Scheme [3]) approximation does not exist modulo some complexity-theoretic assumptions. We do show however that, if we allow additive error guarantees, approximations exist and are computable in polynomial time. This is a reasonable relaxation since numbers we compute are probabilities, hence in the interval $[0, 1]$.

This development leaves two loose ends with respect to the feasibility of our approach. First, when a user asks a query, that user expects an output, not necessarily going tuple by tuple and asking for probabilities. Can there be a compact representation of the infinite answer space? We give the positive answer and explain how to construct finite encodings of infinite probabilistic databases that capture information in them up to a difference that has probability zero.

Second, our approximation algorithm creates several samples of the missing data by instantiating nulls in the input database. Even creating one such sample would be infeasible in real life, as it amounts to, essentially, creating a copy of the entire input data just to ask a query over it. To mitigate this issue, we show that sampling can be done smartly, i.e., encoded in a query that is expressible in our relational algebra. Thus, the sampling approximation algorithm can be expressed by a single query ran over the input database.

The remainder of the paper is organized as follows. We start by recalling relevant notions in databases and probability theory in Section 2. In Section 3, we present our framework of probabilistic databases and its instantiation by means of marked nulls with attached probability distributions. In Section 4, we introduce the query language RA* for databases with numerical entries. Answers for RA* in our framework are then introduced in Section 5 where

we also study their measurability and complexity of exact and approximate computation. We expand on approximation in Section 6, where we present an efficient approximation algorithm for RA* queries over probabilistic databases and show how it can be encoded within RA* queries. Finally, Section 7 shows that answers for RA* queries can be represented finitely. In Section 8 we conclude.

Related Work. The majority of probabilistic databases found in the literature are based on Possible World Semantics [38]. In these models, a probabilistic database (pdb) defines a probability space whose outcomes are complete database instances. Such pdbs are often based on *tuple-level* events, i.e., events defining whether a tuple belongs to a relation. A large body of work on the topic focuses on pdbs where the presence of each tuple is an independent event (*tuple-independent pdbs*) [12]. Even when tuples may define dependent events, such models are usually characterized by finitely-many events and, thus, are not well suited to represent uncertainty on infinite attribute domains. This is the case, for example, of pdbs based on c-tables where tuple variables are not allowed (see, e.g., [11]). To characterize attribute-level uncertainty, one could define events at the *attribute-level*, i.e., focus on the values that uncertain attributes can take. In the literature, however, such models often require attribute domains to be finite (e.g., [20]) or discrete (e.g., [31]) to avoid issues with measurability. To ensure tame behavior of the probability spaces thus defined, query languages considered are often limited to fragments of first-order logic.

The model of this paper is based on Possible World Semantics and attribute-level events. It differs from existing work in that attributes of our pdbs have *continuous domains*, and their events are defined by continuous probability distributions. Moreover, we study query answering in a rich language that is close to actual SQL. Ours is the first formal framework of this kind, to the best of our knowledge.

Probabilistic databases with continuous attribute domains have been studied in *database systems literature* [26, 28, 29, 31, 36]. In this line of work, systems either consider only specific classes of queries or resort to ad-hoc semantics of incompleteness to achieve desired computational behavior. For example, the work in [29] focuses on the expected values and moments of answers, and aggregation is handled via sampling. In [28], query answers are defined directly as the result of a Monte Carlo process while, in [36], answers are defined via a direct manipulation of the probability density functions involved using techniques of [37]. A further interesting example is the work in [17] where attribute-level uncertainty comes in the shape of continuous intervals, and query answers are defined as approximations of classical certain answers. Despite these differences, our formal framework could pave the way for a deeper understanding of the behavior of such systems.

2 BACKGROUND

2.1 Incomplete Databases

Schemas and instances. We assume a countable set $\mathcal{R}$ of *relation schemas*. Each relation schema $R \in \mathcal{R}$ has an associated *arity* denoted by $\texttt{arity}(R)$. A *schema* $\mathcal{S}$ is a set of relation schemas.

We work with the model of marked nulls, i.e., we assume a countable set $\mathsf{Null} := \{\bot_1, \bot_2, \ldots\}$ of *nulls*. The entries in databases come from the set $\mathbb{R} \cup \mathsf{Null}$. That is, we assume that numerical values come from $\mathbb{R}$; since non-numerical values can be enumerated in an arbitrary way, e.g., as $1, 2, \ldots$, we assume without loss of generality that all non-null entries come from $\mathbb{R}$.

A *tuple* of arity k is an element in $(\mathbb{R} \cup \mathsf{Null})^k$. We follow SQL's bag semantics, that is, a *relation* $\mathbf{R}$ over a relation schema $\mathcal{R}$ is a finite multi-set (bag) of tuples with arity $\texttt{arity}(R)$. An *incomplete database* $\mathbf{D}$, or just *database* for short, over a schema $\mathcal{S}$ consists of relations $\mathbf{R}^{\mathbf{D}}$ over each relation schema $\mathcal{R}$ in $\mathcal{S}$. We define $\mathsf{adom}(\mathbf{D})$ as the set of all entries of $\mathbf{D}$, and $\mathsf{Null}(\mathbf{D})$ as all entries of $\mathbf{D}$ that are also in Null. A database $\mathbf{D}$ is *complete* if all its relations instances are *complete*, that is, do not include nulls; Or, alternatively, if $\mathsf{Null}(\mathbf{D}) = \emptyset$.

A *database query* q over database schema $\mathcal{S}$ is a function from databases over $\mathcal{S}$ into databases over $\mathcal{S}'$, for some schema $\mathcal{S}'$ that consists of a single relation symbol. We write $q : \mathcal{S} \to \mathcal{S}'$ when schemas need to be specified explicitly.

Valuations. Valuations assign values to nulls. Formally, a *valuation* $v : \mathsf{Null} \to \mathbb{R}$ is a partial mapping, whose *domain*, i.e., the subset of Null on which it is defined, is denoted by $\mathsf{dom}(v)$. We lift valuations to databases by defining $v(\mathbf{D})$ as the database that is obtained from $\mathbf{D}$ by replacing every null entry $\bot$ with $v(\bot)$.

2.2 Probability Theory

We present the basic notions of probability we use.

Probability spaces. A *σ-algebra* on a set X is a family Σ of subsets of X such that $X \in \Sigma$ and Σ is closed under complement and countable unions. If Σ' is a family of subsets of X, then the σ-algebra *generated by* Σ', denoted by $\sigma(\Sigma')$, is the smallest σ-algebra on X containing Σ'. A *measurable space* is a pair (X, Σ) with X an arbitrary set and Σ a σ-algebra on X. Subsets of X are called *Σ-measurable* if they belong to Σ. A *probability measure* on (X, Σ) is a function $P : \Sigma \to [0, 1]$ with $P(\emptyset) = 0$, $P(X) = 1$ and $P(\cup_{i=1}^{n} \sigma_i) = \sum_{i=1}^{n} P(\sigma_i)$ for every sequence $\sigma_1, \ldots, \sigma_n$ of disjoint measurable sets. A measurable space equipped with a probability measure is called a *probability space*. When $X := \mathbb{R}$, we implicitly assume that Σ is the Borel σ-algebra Σ_B (i.e., the one generated by open intervals (a, b) for $a < b \in \mathbb{R}$), and hence, we only specify the density function that suffices to determine the probability space.

Measurable mappings. If (X, Σ) and (X', Σ') are measurable spaces, and $\phi : X \to X'$ is a mapping then we say that ϕ is *(Σ, Σ')-measurable* (or simply measurable if Σ, Σ' are clear from context) if the preimage ϕ^{-1} under ϕ of every Σ'-measurable set is Σ-measurable.

For additional details, we refer the reader to [35, Section 5].

3 QUERYING AND INSTANTIATING PROBABILISTIC DATABASES

Our model of probabilistic databases describes all possible data configurations together with a probability measure defining how likely they are to occur. This intuition is captured by the following notion.

Definition 1. A *probabilistic database (PDB)* $\mathcal{D}$ over a database schema $\mathcal{S}$ is a probability space whose domain is a set of complete databases over $\mathcal{S}$.

Notice that, similarly to [22], the domain of a PDB $\mathcal{D}$ may be uncountable due to the fact that $\mathcal{D}$ uses reals as entries. However,

in most models of probabilistic databases known in the literature, one assumes *tuple independence*, i.e., the presence of each tuple is an independent event. This is not the case for the PDBs presented in Definition 1 where no restriction is imposed on the probability distribution.

3.1 Querying PDBs

To define queries for PDBs, we start from the standard notion of queries for complete databases. Let q be a query over a schema $\mathcal{S}$, and let $\mathcal{D} := (X_\mathcal{D}, \Sigma_\mathcal{D}, P_\mathcal{D})$ be a probabilistic database over $\mathcal{S}$. For a query q, we write $X_{q,\mathcal{D}}$ to denote the set $\{q(\mathbf{D}) \mid \mathbf{D} \in X_\mathcal{D}\}$ of all possible answers for q over $X_\mathcal{D}$. We equip $X_{q,\mathcal{D}}$ with the σ-algebra $\Sigma_{q,\mathcal{D}}$ generated by the family $\{A \subseteq X_{q,\mathcal{D}} \mid q^{-1}(A) \in \Sigma_\mathcal{D}\}$. In other words, $\Sigma_{q,\mathcal{D}}$ is the σ-algebra generated by those images of q that are measurable in $\mathcal{D}$. For each element σ of $\Sigma_{q,\mathcal{D}}$, we set $P_{q,\mathcal{D}}(\sigma) := P_\mathcal{D}(q^{-1}(\sigma))$, provided the latter is defined.

Proposition 2. *Given a PDB $\mathcal{D}$ over a schema $\mathcal{S}$, and a query q over $\mathcal{S}$,*

- *q is $(\Sigma_\mathcal{D}, \Sigma_{q,\mathcal{D}})$-measurable; and*
- *$P_{q,\mathcal{D}}$ is defined over every element of $\Sigma_{q,\mathcal{D}}$ and is a probability measure for $(X_{q,\mathcal{D}}, \Sigma_{q,\mathcal{D}})$.*

In view of Proposition 2, we can define the probability space for the answers to q over $\mathcal{D}$ as follows.

Definition 3. The *answer space of q over $\mathcal{D}$* is defined as the probabilistic database $q(\mathcal{D}) := (X_{q,\mathcal{D}}, \Sigma_{q,\mathcal{D}}, P_{q,\mathcal{D}})$.

We point out next that, contrary to tuple-independent probabilistic databases, our model preserves composability of query languages.[1] We say that a pair (q, q') of queries is *composable* if $q : \mathcal{S} \to \mathcal{S}'$ and $q' : \mathcal{S}' \to \mathcal{S}''$. In this case, we define their *composition* $(q' \circ q)(\mathbf{D}) := q'(q(\mathbf{D}))$. Can we lift composability to PDBs? The following proposition shows that the answer to this question is positive.

Proposition 4. *Let (q, q') be composable queries. Then $(q' \circ q)(\mathcal{D}) = q'(q(\mathcal{D}))$ for every PDB $\mathcal{D}$ and query q for $\mathcal{D}$.*

Note that composability is a consequence of the answer space definition, and is independent of the query language.

3.2 PDBs from Incomplete Databases

We now show how the PDBs we introduced in Definition 1 can be instantiated naturally via incomplete databases.

To associate an incomplete database $\mathbf{D}$ with a PDB, we equip each $\perp_i \in \mathsf{Null}$ with a probability density function P_i.

We assume without loss of generality that $\mathsf{Null}(\mathbf{D}) := \{\perp_1, \ldots, \perp_n\}$, and we then define the *measurable space of valuations* (X_V, Σ_V) of $\mathsf{Null}(\mathbf{D})$ such that X_V is the set of all valuations v with $\mathrm{dom}(v) = \mathsf{Null}(\mathbf{D})$, and Σ_V is the σ-algebra generated by the family consisting of the sets

$$V(\sigma_1, \ldots, \sigma_n) := \{v \mid v(\perp_i) \in \sigma_i\}$$

for each $\sigma_1, \ldots, \sigma_n$ in the Borel σ-algebra Σ_B. Notice that the range of the valuations of each such set is a Cartesian product of sets in Σ_B. The events defined by these sets, i.e., $v(\perp_i) \in \sigma_i$, for each $i = 1, \ldots, n$, are assumed to be mutually independent.

[1] Note that even the simplest operations, e.g., computing union of a relation with itself, results in a violation of tuple-independence.

$$\begin{array}{lcl} q & := & R \mid \pi_{\$i_1,\ldots,\$i_k}(q) \mid \sigma_\theta(q) \mid q \times q \mid q \cup q \mid q \setminus q \\ & & \mid \textsc{Apply}_f(q) \mid \Sigma^{\$j}_{\$i_1,\ldots,\$i_k}(q) \\ \theta & := & \$i = \$j \mid \$i < \$j \\ f & \in & \mathrm{RAT}[\$1, \$2, \ldots], \quad k \geq 0 \end{array}$$

Figure 1: RA* Queries

To reflect this, we would like to define a probability function P_V for which the following requirement holds:

$$P_V(V(\sigma_1, \ldots, \sigma_n)) = \Pi_{i=1}^n P_i(\sigma_i) \tag{1}$$

As it turns out, this requirement determines a unique probability measure for (X_V, Σ_V).

Theorem 5. *There exists a unique probability measure P_V for the measurable space of valuations of* $\mathsf{Null}(\mathbf{D})$ *that satisfies Equation (1).*

From now on, we refer to (X_V, Σ_V, P_V) as the *valuation space* of $\mathsf{Null}(\mathbf{D})$. The valuation space, however, does not define a PDB, since multiple valuations may define the same complete database.[2]

To overcome this, we define the PDB of $\mathbf{D}$ as follows. Let $X_\mathbf{D}$ be the set of all possible instantiations of $\mathbf{D}$, i.e., $X_\mathbf{D} := \{v(\mathbf{D}) \mid \mathrm{dom}(v) = \mathsf{Null}(\mathbf{D})\}$, and let $\chi : X_V \to X_\mathbf{D}$ be the mapping such that $\chi(v) := v(\mathbf{D})$. We set

$$\Sigma_\mathbf{D} := \{A \mid \chi^{-1}(A) \in \Sigma_V\}.$$

Proposition 6. *$\Sigma_\mathbf{D}$ is a σ-algebra on $X_\mathbf{D}$ for every incomplete database $\mathbf{D}$.*

With Proposition 6 in place, we can move to define a probability measure for the measurable space $(X_\mathbf{D}, \Sigma_\mathbf{D})$. To this end, we define $P_\mathbf{D} : \Sigma_\mathbf{D} \to [0, 1]$ by setting, for each $A \in \Sigma_\mathbf{D}$,

$$P_\mathbf{D}(A) := P_V(\chi^{-1}(A))$$

Proposition 7. *$P_\mathbf{D}$ is a probability measure for $(X_\mathbf{D}, \Sigma_\mathbf{D})$ for every incomplete database $\mathbf{D}$.*

As a consequence of the previous results, we now have a well-defined concept of the probabilistic space of $\mathbf{D}$. We record this in the following definition.

Definition 8. The *PDB of* $\mathbf{D}$ is $(X_\mathbf{D}, \Sigma_\mathbf{D}, P_\mathbf{D})$.

With a slight abuse of notation, from now on, we shall sometimes refer to the PDB of $\mathbf{D}$, simply as $\mathbf{D}$.

4 QUERY LANGUAGE: EXTENDED RELATIONAL ALGEBRA

To analyze our framework, we need to focus on a concrete query language. Our choice is a compact language that has sufficient power to express standard SQL queries with arithmetic and aggregation, similar to [33]. This language, RA* is an extended form of relation algebra and its syntax is presented in Figure 1. Due to space limitations, the full formal semantics is in the Appendix; here

[2] For example, given a Unary relation that consists of two nulls, for any valuation there is a symmetric one (that replaces one nulls with the other) that defines the same database.

we introduce the main constructs and explain the expressiveness of the language.

Like SQL, the queries of RA* are interpreted under *bag semantics*, i.e., relations may have multiple occurrences of the same tuple.

The first line of the Figure 1 contains the standard operations of Relational Algebra. We use the unnamed perspective [1] and refer to attributes by their positions in the relation, i.e., $\$1, \$2$, etc, for the first, second, etc attribute of a relation. Projection and selection keep duplicates; Cartesian product $\times$ multiplies occurrences, $\cup$ as SQL's UNION ALL adds them, and $\setminus$ as SQL's EXCEPT ALL subtracts the number of occurrences, as long as it is non-negative.

The operation $\textsc{Apply}_f(q)$ takes a function $f \in \text{RAT}[\$1, \$2, \ldots]$, which is a rational function over variables $\$i$, $i \in \mathbb{N}$, i.e., a ratio of two polynomials with real coefficients. It then adds a column to a relation with the result of this function for each tuple in the result of q. For example, if $f(\$1, \$2) = (\$1)^2/\$2$, then $\textsc{Apply}_f(R)$ turns a binary relation R into a ternary relation that consists of tuples $(a, b, a^2/b)$ for every occurrence of (a, b) in R.

The operation $\sum^{\$j}_{\$i_1,\ldots,\$i_k}(q)$ is grouping with summation aggregation; intuitively, this is SQL's SELECT $\$i_1, \ldots, \$i_k$, SUM($\$j$) FROM q GROUP BY $\$i_1, \ldots, \$i_k$. When $k = 0$, we group by the empty set, thus obtaining a query that returns a unique value who is the sum of all values in column j.

While we have chosen a rather minimalistic language for the convenience of presentation and proofs, RA* actually packs considerable expressive power. For example, it can express all of the following:

- Arbitrary conditions $f\ \omega\ g$ where $\omega \in \{=, \neq, <, >, \leq, \geq\}$ and f, g are polynomials in variables $\$1, \$2$, etc. Indeed values of f, g can be attached as new attributes, and those can be compared with = and < comparisons; other comparisons are expressed by means of set operations.
- All SQL aggregates MIN, MAX, AVG, SUM, COUNT. Indeed, MIN and MAX can be expressed in standard relational algebra. To express count, we can attach an attribute with constant value 1 and add up those values. For example, SQL's SELECT A, COUNT(B) FROM R GROUP BY A for a two-attribute relation is expressed as $\sum^{\$3}_{\$1}(\textsc{Apply}_1(R))$. To express multiple aggregates, one needs to take the product of the input relation with the result of the first aggregation to restore duplicates, and then perform another aggregation. And finally average can be expressed as the ratio of SUM and COUNT.
- Duplicate elimination is encoded in grouping: one needs to add an aggregate and project out grouping attributes. For example, to eliminate duplicates from a single-attribute relation R, one would write $\pi_{\$1}\Big(\sum^{\$2}_{\$1}(\textsc{Apply}_1(R))\Big)$.
- Conditions involving subqueries, as in IN and EXISTS conditions used in SQL's WHERE. This is already doable in relational algebra [24, 42].

Thus, we see that despite its simplicity, the language is significantly expressive, as it captures the essence of SELECT-FROM-WHERE-GROUP BY-HAVING queries of SQL.

5 QUERY ANSWERING

With the query language in place, we now turn to query answering. To do so, we focus on specific subsets of the answer space; essentially we ask, as was outlined in the introduction, whether values in output tuples belong to specific intervals. We want to know the probability of such events. In this section we show that such probabilities are well defined, and look at the complexity of their exact computation (which will be high, thus leading us to approximation algorithms in the following section).

5.1 Target Sets of the Answer Space

As we already explained, when dealing with numerical values, returning certain or possible tuples of constants is not very informative. Instead we want to settle for providing *intervals* of values. These intervals may be fixed, say $[1, 2]$, or even defined by functions on nulls in a database, say $[\$1 + 3, \$1 + 4]$.

Formally, we define *interval tuples* as tuples $\bar{a}$ whose entries are open, or closed, or half-open half-closed intervals of one of the forms $(x, y), [x, y], [x, y), (x, y]$ where $x, y \in \text{RAT}[\perp_1, \perp_2, \ldots] \cup \{\pm\infty\}$. We denote by $\text{Null}(\bar{a})$ the set of all nulls that appear in the entries of $\bar{a}$, and by $v(\bar{a})$ the tuple obtained from $\bar{a}$ by replacing each $\perp_i$ with $v(\perp_i)$. In such a tuple, all intervals are grounded, i.e. their endpoints are real numbers.

We say that a tuple of constants $\bar{t} := (t_1, \ldots, t_n)$ is *consistent* with $v(\bar{a})$ if $t_i \in v(a_i)$ for each i, that is, t_i is a value within the interval $v(a_i)$. For the rest of the section we use the symbol $\circ$ to denote one of the usual comparisons, i.e., $<, =$, or $>$. We write $\#(v(\bar{a}), A) \circ \mathsf{k}$ when the number of tuples in the bag A that are consistent with $v(\bar{a})$ is $\circ\mathsf{k}$. With this, we can move to the following key definition.

Definition 9.

$$\text{out}_{q,\mathbf{D},\circ}(\bar{a}, \mathsf{k}) := \left\{ q(v(\mathbf{D})) \in X_{q,\mathbf{D}} \;\middle|\; \begin{array}{l} \text{dom}(v) = \text{Null}(\mathbf{D}), \\ \#(v(\bar{a}), q(v(\mathbf{D}))) \circ \mathsf{k} \end{array} \right\}$$

Intuitively, $\text{out}_{q,\mathbf{D},\circ}(\bar{a}, \mathsf{k})$ is the set that consists of all those databases in $X_{q,\mathbf{D}}$ such that each contains $\circ\mathsf{k}$ tuples consistent with $\bar{a}$. Note that $X_{q,\mathbf{D}}$ is the domain of the answer space of q over $\mathbf{D}$ as defined in Definition 3. We call $\text{out}_{q,\mathbf{D},\circ}(\cdot, \cdot)$ the *target mapping* for q over $\mathbf{D}$. Target mappings capture information about the query answer space, and we next see how to compute them.

5.2 Measurability of Target Sets

The target mapping can provide us with information about the answer space. In particular, we can compute how likely it is that tuples consistent with $\bar{a}$ appear in the answer (at most, at least or exactly) k times. This is captured by the following computational problem.

LIKELIHOOD$[q, \circ, \mathsf{k}]$	
PARAMETERS:	A query $q \in \text{RA}^*$, $\mathsf{k} \geq 0$, and $\circ \in \{<, =, >\}$
INPUT:	An incomplete database $\mathbf{D}$, and an interval tuple $\bar{a}$
PROBLEM:	Compute $P_{q,\mathbf{D}}(\text{out}_{q,\mathbf{D},\circ}(\bar{a}, \mathsf{k}))$

Is this problem even well-defined? In other words, is it always the case that $P_{q,\mathbf{D}}(\text{out}_{q,\mathbf{D},\circ}(\bar{a},\mathsf{k}))$ exists? The answer is not immediate since not every subset of $X_{q,\mathbf{D}}$ is measurable. We settle this issue with the following claim.

Theorem 10. *For every $q \in RA^*$, incomplete database $\mathbf{D}$, interval tuple $\bar{a}$, $\mathsf{k} \geq 0$, and $\circ \in \{<,=,>\}$, it holds that*

$$\text{out}_{q,\mathbf{D},\circ}(\bar{a},\mathsf{k}) \in \Sigma_{q,\mathbf{D}}$$

To obtain Theorem 10, we first observe that $\text{out}_{q,\mathbf{D},\circ}(\bar{a},\mathsf{k})$ is the union of some countable family of sets $\text{out}_{q,\mathbf{D},=}(\bar{a},\mathsf{k})$. Since each such set is definable by a first-order formula in the theory of the reals with $|\mathsf{Null}(\mathbf{D})| = n$ free variables, we can conclude that each set is the finite union of open sets in $\mathbb{R}^n$, which are Borel-measurable $\mathbb{R}^n$, and sets of smaller dimensions which have measure zero in $\mathbb{R}^n$ ([43]). The claim follows from the fact that the PDB defined by $\mathbf{D}$ is Borel-isomorphic to $\mathbb{R}^n$, that is, there exists a bijection between the two spaces (see, e.g., [16, Section 1] for more details).

5.3 Computing Likelihood$[q,\circ,\mathsf{k}]$ Exactly

Solving the computational problem Likelihood$[q,\circ,\mathsf{k}]$ exactly is a very challenging task that is unfeasible in practice even for simple queries. Let SPC be the fragment of RA* that allows only basic relations, selection, projection, and Cartesian product. We can prove that the Likelihood$[q,\circ,\mathsf{k}]$ is not rational even for queries in SPC and nulls defined by basic distributions. An *exponentially distributed* database is an incomplete database whose null values are annotated with exponential probability distributions $\lambda e^{-\lambda x}$, where λ is rational.

Proposition 11. *For each $\circ \in \{<,=,>\}$, there exists a Boolean query $q \in \mathsf{SPC}$ and a family of exponentially distributed databases $\{\mathbf{D}_\mathsf{k} \mid \mathsf{k} > 0 \in \mathbb{N}\}$ such that Likelihood$[q,\circ,\mathsf{k}](\mathbf{D}_\mathsf{k},())$ is a transcendental number.*

In other words, Proposition 11 implies that it may not be possible to even write down the exact output of Likelihood$[q,\circ,\mathsf{k}](\mathbf{D}_\mathsf{k},())$. Is it due to the density functions we chose? It turns out, that even when we consider very simple and well-behaved distributions the problem remains computationally challenging. Formally, a *uniform interval database* is an incomplete database whose nulls are annotated with uniform interval distributions with rational parameters. That is, their density function is a rational constant $1/(u-l)$ over an interval $[l,u]$, with $l < u$ both rational, and 0 everywhere else. We denote by Likelihood$_{int}[q,\circ,\mathsf{k}]$ the restriction of Likelihood$[q,\circ,\mathsf{k}]$ to input databases from the class of uniform interval databases.

Theorem 12. *For each $\circ \in \{<,=,>\}$, there exists a Boolean query $q \in \mathsf{SPC}$ such that Likelihood$_{int}[q,\circ,\mathsf{k}]$ is #P-hard, for each $\mathsf{k} \geq 0$.*

We recall that hard problems in the class #P ([3]) are at least as hard as the whole polynomial hierarchy [40]. From a practical perspective, it is often enough to check whether Likelihood$[q,\circ,\mathsf{k}]$ exceeds a given threshold. We thus define the decision version of Likelihood$[q,\circ,\mathsf{k}]$:

Threshold$[q,\circ,\mathsf{k},\delta]$	
Parameters:	A query $q \in \text{RA}^*$, $\mathsf{k} \geq 0$, a *threshold* $0 \leq \delta \leq 1$, and $\circ \in \{<,=,>\}$
Input:	An incomplete database $\mathbf{D}$, and a interval tuple $\bar{a}$
Problem:	Decide whether $P_{q,\mathbf{D}}(\text{out}_{q,\mathbf{D},\circ}(\bar{a},\mathsf{k})) > \delta$

The case Threshold$[q,<,0,\delta]$ is trivial, and it will not be considered in what follows. We call Threshold$_{int}[q,\circ,\mathsf{k},\delta]$ the restriction of Threshold$[q,\circ,\mathsf{k},\delta]$ to uniform interval databases only. Even this problem is not simple to solve, as the following theorem suggests.

Theorem 13. *For each $\circ \in \{<,=,>\}$, there exists a Boolean query $q \in RA^*$ such that Threshold$_{int}[q,\circ,\mathsf{k},0]$ is NP-hard.*

This has consequences also on approximation algorithms for Likelihood$[q,\circ,\mathsf{k}]$. A *Fully-Polynomial Randomised Approximation Scheme* (FPRAS) for $f : X \to Y$ [3] is an algorithm A with the following guarantees, for input $x \in X$ and $\varepsilon \in (0,1]$, :

- A runs in time polynomial in the size of x and $\frac{1}{\varepsilon}$; and
- A returns a random variable $A(x,\varepsilon)$ such that

$$P(|A(x,\varepsilon) - f(x)| \leq \varepsilon \cdot |f(x)|) \geq 0.75.$$

Of course 0.75 can be replaced by any constant > 0.5; here we follow the traditional definition of 0.75 as the chosen constant.

Corollary 14. *Unless RP = NP, there exists $q \in RA^*$ such Likelihood$_{int}[q,\circ,\mathsf{k}]$ admits no FPRAS.*

The class *RP* consists of decision problems that can be solved efficiently by a randomized algorithm with a bounded one-sided error see, e.g., [3]. It is commonly believed that $RP \neq NP$, since, otherwise, problems in NP would admit efficient ephratctical solutions.

Are there other approximation schemes that can lead to better results? We answer this in the next section.

6 APPROXIMATION

In this section, we present an efficient approximation scheme with additive error for Likelihood$[q,\circ,\mathsf{k}]$, hence answering positively the question posed at the end of the previous section.

To deal with the numerical nature of our problem, in what follows we assume an extension of the RAM model of computation with the following assumptions: values in $\mathbb{R}$ are represented with arbitrary precision in one single register, and basic arithmetic operations $(+,-,\cdot,\div)$ are performed in constant time. Note that this model is commonly used for numerical problems (e.g., [6]).

An *Additive Error Fully-Polynomial Randomised Approximation Scheme* (AFPRAS) for $f : X \to Y$ is an algorithm A with the following guarantees, for input $x \in X$ and $\varepsilon \in (0,1]$, :

- A runs in time polynomial in the size of x and $\frac{1}{\varepsilon}$; and
- A returns a random variable $A(x,\varepsilon)$ such that

$$P(|A(x,\varepsilon) - f(x)| \leq \varepsilon) \geq 0.75.$$

The difference between FPRASs discussed previously and AFPRASs is in that the latter is allowed an additive rather than multiplicative error, i.e., the difference between the value an AFPRAS returns and the actual value it approximates is fixed rather than proportional.

6.1 Sampling the Valuation Space

Our approximation scheme is obtained by sampling valuations of the input database using *samplers*, i.e., algorithms that return random variables distributed according to some predefined distribution (see, e.g., [5]). For commonly used probability distributions, samplers can be obtained using different techniques (see, e.g., [7, 32]).

Formally, a *sampler* for a probability space $\mathcal{S} = (X_S, \Sigma_S, P_S)$ is a randomized algorithm A that returns a random variable X such that $P(X \in \sigma) = P_S(\sigma)$, for each $\sigma \in \Sigma_{\mathcal{S}}$. Given an incomplete database $\mathbf{D}$, we say that $\mathbf{D}$ is *efficiently sampled* (E.S.) if, for every $\perp_i \in \mathsf{Null}(\mathbf{D})$ defined by the probability space N_i, there exists a efficient sampler for N_i that runs in time polynomial in the size of $\mathbf{D}$. In what follows, we will use $\mathtt{Sample}_\mathtt{i}$ for the result of an efficient sampler for N_i.

Using the notions introduced so far, Algorithm 1 defines an approximated sampler for the space of valuation of incomplete databases that is E.S.

Algorithm 1 `ValSampler`$[\mathbf{D}]$

Parameter: An E.S. database $\mathbf{D}$ with $|\mathsf{Null}(\mathbf{D})| = n$
Output: A valuation v for $\mathsf{Null}(\mathbf{D})$
for all $\perp_i \in \mathsf{Null}(\mathbf{D})$ **do**
  $s_i := \mathtt{Sample}_\mathtt{i}$
end for
Let $v : \mathsf{Null}(\mathbf{D}) \to \mathbb{R}$ s.t. $v(\perp_i) = s_i$, for each $\perp_i \in \mathsf{Null}(\mathbf{D})$;
return v;

The following claim formalizes the relevant properties of `ValSampler`$[\mathbf{D}]$.

Lemma 15. *For every incomplete database* $\mathbf{D}$, `ValSampler`$[\mathbf{D}]$ *is a sampler for the valuation space of* $\mathsf{Null}(\mathbf{D})$. *Moreover,* `ValSampler`$[\mathbf{D}]$ *runs in time polynomial in the size of* $\mathbf{D}$.

6.2 Approximation via Direct Sampling

For input E.S. databases, we can obtain an AFPRAS for Likelihood$[q, \circ, \mathsf{k}]$ using `ValSampler`$[\mathbf{D}]$. The idea behind our technique is the following. Assume $q \in \mathrm{RA}^*$, an incomplete database $\mathbf{D}$, and an interval tuple $\bar{a}$. The algorithm uses `ValSampler`$[\mathbf{D}]$ to sample a number of valuations v from the valuation space of $\mathsf{Null}(\mathbf{D})$, and computes the average number of such v for which $\#(v(\bar{a}), q(v(\mathbf{D}))) \circ \mathsf{k}$ holds. Using well-known bounds on the expected values of a sum of independent random variables, we can prove that, with high probability, the value of this average is close enough to Likelihood$[q, \circ, \mathsf{k}](\mathbf{D}, \bar{a})$. We formalize this intuition in Algorithm 2.

Lemma 16. *For every incomplete E.S. database* $\mathbf{D}$, *interval tuple* $\bar{a}$ *and* $\varepsilon \in (0, 1]$ *it holds that*

$$P(|\mathtt{LikeApx}[q, \circ, \mathsf{k}](\mathbf{D}, \bar{a}, \varepsilon) - \textsc{Likelihood}[q, \circ, \mathsf{k}](\mathbf{D}, \bar{a})| \leq \varepsilon) \geq 0.75$$

We are finally ready to formally conclude the existence of an AFPRAS for Output Tuple Likelihood:

Theorem 17. *For every* $q \in RA^*$, *comparison* $\circ \in \{<, =, >\}$, *and integer* $\mathsf{k} \geq 0$, *the algorithm* `LikeApx`$[q, \circ, \mathsf{k}]$ *is an AFPRAS for* Likelihood$[q, \circ, \mathsf{k}]$ *over input databases that are E.S.*

Algorithm 2 `LikeApx`$[q, \circ, \mathsf{k}]$

Parameters: $q \in \mathrm{RA}^*$, $\circ \in \{<, =, >\}$, an integer $\mathsf{k} \geq 0$.
Input: An E.S. database $\mathbf{D}$, an interval tuple $\bar{a}$, $\varepsilon \in (0, 1]$.
Output: A number between 0 and 1
Let $\gamma := \lceil \varepsilon^{-2} \rceil$;
`count` := 0
for all `i` $= 1, \ldots, \gamma$ **do**
  v := `ValSampler`$[\mathbf{D}]$
  if $\#(v(\bar{a}), q(v(\mathbf{D}))) \circ \mathsf{k}$ **then**
    `count` := `count` + 1
  end if
end for
return $\frac{\mathtt{count}}{\gamma}$

6.3 Encoding Sampling into Queries

Our approximation algorithm `LikeApx`$[q, \circ, \mathsf{k}]$ requires a separate instantiation of the input database $\mathbf{D}$ for each sample. Practical implementations of the algorithm would need to generate these instantiations directly inside the RDBMS hosting $\mathbf{D}$, thus leading to an overhead that is unacceptable in practical scenarios. Instead of explicitly generating this data, however, it is possible to encode a full run of `LikeApx`$[q, \circ, \mathsf{k}]$ into a query to be executed directly over $\mathbf{D}$. In this section we present such construction.

First, we introduce some notation. To keep notation compact, in all following statements of our results, we let q be a generic query in RA^*, $\mathbf{D}$ be an incomplete database, $\bar{a}$ be an interval tuple, k be a positive integer, and $\circ$ be an element of $\{<, =, >\}$. Moreover, to evaluate RA^* queries directly over incomplete databases, we will implicitly assume *naive evaluation* [19], i.e., we will treat nulls as standard constants.

Theorem 18. *For every* $\epsilon \in (0, 1]$, *there exists an RA* query* apx_ε *such that*

$$P(|\mathsf{apx}_\varepsilon(\mathbf{D}) - \textsc{Likelihood}[q, \circ, \mathsf{k}](\mathbf{D}, \bar{a})| \leq \varepsilon) \geq 0.75.$$

The construction of apx_ε is done in several steps that we proceed to describe. We first show that we can compile a specific valuation v into q, that is, we can rewrite q to obtain $\tilde{q}_v$ such that for every input $\mathbf{D}$ we have

$$\tilde{q}_v(\mathbf{D}) = q(v(\mathbf{D}))$$

Using the COUNT aggregation, in particular the one corresponding with SQL's COUNT$(*)$, we can obtain q_v such that, if $\sharp(\bar{t}, \tilde{q}_v(\mathbf{D})) = k'$, then $(\bar{t}, k')$ appears once in $q_v(\mathbf{D})$. We do so by setting

$$q_v := \mathrm{COUNT}_{\$1,\$2,\ldots,\$n}(\tilde{q}_v)$$

Since we are interested in the total number of tuples of $q(v(\mathbf{D}))$ that are consistent with $\bar{a} := (a_1, \ldots, a_n)$, we set

$$q_{v,\bar{a}} := \pi_{\$n+1}\left(\sigma_{\$1 \in v(a_1) \wedge \cdots \wedge \$n \in v(a_n)}\,(q_v)\right)$$

where $\$j \in v(a_j)$ is the condition requiring that $\$j$ is within the interval $v(a_j)$. Then, to check whether this multiplicity is $\circ k$, we sum the multiplicities that $q_{v,\bar{a}}$ returns and select upon the condition $\$1 \circ k$. Finally, we project the result on the empty set, thus obtaining

the Boolean query

$$q_{v,\bar{a},\circ k} := \pi_{\emptyset}\left(\sigma_{\$1\circ k}\left(\sum_{\emptyset}^{\$1} q_{v,\bar{a}}\right)\right)$$

This query returns true, i.e., one occurrence of the empty tuple if $q(v(\mathbf{D}))$ contains $\circ k$ tuples consistent with $\bar{a}$, and false, i.e., the empty bag, otherwise.

Finally, let $V = \{v_1, \ldots, v_\gamma\}$ be the results of running the algorithm `ValSampler`$[\mathbf{D}]$ for $\gamma = \lceil \varepsilon^{-2} \rceil$ times. We define

$$\mathsf{apx}_\varepsilon := \mathrm{AVG}_{\emptyset}\left(\bigcup_{j=1}^{\gamma} q_{v_j,\bar{a},\circ k}\right).$$

and show that it indeed has the desired properties. Moreover, we can show that it can be efficiently computed.

Proposition 19. *apx_ε can be computed in time polynomial in the size of q, $\mathbf{D}$, and $\bar{a}$, and the value ε^{-1}.*

The construction presented so far can be adapted to return the possible answers of $q(\mathbf{D})$ along with their approximated probability. To this end, we define

$$\mathtt{compute}_\varepsilon := \textsc{Apply}_{\frac{\$1}{\gamma}}\left(\mathrm{COUNT}_{\$1,\ldots,\$n+1}\left(\bigcup_{j=1}^{\gamma} q_{v_j}\right)\right)$$

Recall that $\bigcup_{j=1}^{\gamma} q_{v_j}$ returns tuples of the form $(\bar{t}, k')$ such that $\bar{t}$ occurs k' times in $q(v_j(\mathbf{D}))$. Thus, $\mathrm{COUNT}_{\$1,\ldots,\$n+1}\left(\bigcup_{j=1}^{\gamma} q_{v_j}\right)$ results in tuples of the form $(\bar{t}, k', k'')$ with the same guarantees as before and, in addition, k'' is the number of valuations v amongst $v_1, \ldots, v_\gamma$ for which $\bar{t}$ occurs k' times in $q(v(\mathbf{D}))$. We can show that each such tuple represents a possible answer for q over $\mathbf{D}$. In the next theorem, by a slight abuse of notation, we refer to a tuple $(c_1, \ldots, c_n)$ as the interval tuple $([c_1, c_1], \ldots, [c_n, c_n])$.

Theorem 20. *Let $\epsilon \in (0, 1]$. Then, for every tuple $(\bar{c}, b, p)$ in $\mathtt{compute}_\varepsilon(\mathbf{D})$,*

$$P(|p - \textsc{Likelihood}[q, =, b](\mathbf{D}, \bar{c})| \leq \varepsilon) \geq 0.75$$

7 FINITE REPRESENTATION OF THE ANSWER SPACE

Additionally to the encoding technique presented previously, a desirable feature that may hint that our model can be deployed in practical use-cases is the ability to represent the answer space of RA* queries in a finite manner. This technique allows us to analyze fully the answer space of a given query which is fundamental in decision-support scenarios.

Do incomplete databases suffice for instantiating the answer space? Let us investigate the answer space of $\sigma_{\$i<\$j}$ over $\mathcal{D}$ – the PDB of $\mathbf{R} := \{\!\{(\bot_1, \mu)\}\!\}$, where $\{\!\{\cdot\}\!\}$ brackets are used for bags. If $\bot_1$ is distributed according to, e.g., the normal distribution with mean μ, then the domain of the answer space contains the empty relation along with instantiations of $\mathbf{R}$ for which $\bot_1 > \mu$. Therefore, there is no incomplete database that instantiates precisely the answer space. Similarly to [27] which dealt with the same issue for non-probabilistic databases, these considerations lead to a notion of *conditional* worlds.

7.1 Conditional Worlds

The idea behind conditional worlds is to separate the answer space to sub-spaces, each instantiatable by an incomplete database, along with a condition that indicates when it is valid. For the previous example, the answer space can be represented by the empty relation with the condition $\bot_1 < \mu$, and by the relation that consists of $(\bot_1, \mu)$ attached with $\bot_1 > \mu$. Note that here and throughout this section, we assume that the probability of each null to be equal to a constant is zero.

Definition 21. A probability distribution $(\mathsf{dom}, \Sigma, P)$ is said to be *singleton unlikely* if $P(\{c\}) = 0$ for every $c \in \mathsf{dom}$.

This is a very mild restriction since all continuous random variables enjoy this property [35].

To formally define conditional worlds we need to define conditional databases. Intuitively, conditional databases are pairs of (database, condition) such that database is valid whenever the condition holds. To carry full information on the computation and to express the conditions, we extend the domain. We define an *arithmetic database* as a database whose active domain is $\mathrm{RAT}[\bot_1, \bot_2, \ldots]$, that is, ratios of polynomials with real coefficients over the nulls.

Definition 22. A *conditional database* is a pair $(\mathbf{A}, c_{\mathbf{A}})$ where

- $\mathbf{A}$ is an arithmetic database, and
- $c_{\mathbf{A}}$ is a set of *arithmetic conditions* which are elements of the form $e < 0$ where $e \in \mathrm{RAT}[\bot_1, \bot_2, \ldots]$; For simplicity, we refer to $e < 0$ as e.

We denote the set of all nulls that appear in $\mathbf{A}$ and in $c_{\mathbf{A}}$ by $\mathsf{Null}(\mathbf{A}, c_{\mathbf{A}})$.

To specify when an arithmetic condition holds, for an arithmetic condition c, and a valuation v, we set $v(c) := \mathbf{t}$ if $v(e) < 0$ for every $e \in c$; otherwise, we set $v(c) := \mathbf{f}$.

Definition 23. A *conditional world* C is a set $\{(\mathbf{A}_1, c_1), \ldots, (\mathbf{A}_m, c_m)\}$ of conditional databases such that

- $P_V(\{v \mid v(c_1) = \mathbf{t} \vee \cdots \vee v(c_m) = \mathbf{t}\}) = 1$, and
- $\{v \mid v(c_i) = \mathbf{t} \wedge v(c_j) = \mathbf{t}\} = \emptyset$ for every $i \neq j$

where $(\mathsf{dom}_V, \Sigma_V, P_V)$ is the valuation space of $\mathsf{Null}(C)$ defined as the union $\bigcup_{i=1}^{m} \mathsf{Null}(\mathbf{A}_i, c_i)$.

In the previous definition it may be the case that $U := \{v \mid v(c_1) = \mathbf{t} \vee \cdots \vee v(c_m) = \mathbf{t}\} \subsetneq \mathsf{dom}_V$, which implies that $P_V(\mathsf{dom}_V \setminus U) = 0$.

7.2 Interpreting Conditional Worlds as PDBs

In the next section we will explain how conditional worlds can instantiate the answer space, but before we show how to interpret these worlds as PDBs.

Definition 24. The *probabilistic interpretation* $(\mathsf{dom}_C, \Sigma_C, P_C)$ *of a conditional world* $C := \{(\mathbf{A}_1, c_1), \ldots, (\mathbf{A}_m, c_m)\}$ is defined by:

$$\mathsf{dom}_C := \bigcup_{i=1}^{m} \{v(\mathbf{A}_i) \mid \mathsf{dom}(v) = \mathsf{Null}(C)\}$$

$$\Sigma_C := \{A \mid \chi^{-1}(A) \in \Sigma_V\}$$

$$P_C(A) := P_V(\chi^{-1}(A))$$

where $(\mathrm{dom}_V, \Sigma_V, P_V)$ is the valuation space of $\mathsf{Null}(C)$, and $\chi : \mathrm{dom}_V \to \mathrm{dom}_C$ is defined by

$$\chi(v) := \begin{cases} v(\mathbf{A}_1) & \text{if } v(c_1) = \mathbf{t} \\ \vdots & \vdots \\ v(\mathbf{A}_m) & \text{otherwise} \end{cases}$$

Note that χ is well-defined due to Definition 23, and hence the probabilistic interpretation is also well-defined.

Before proceeding, we show that, as its name hints, the probabilistic interpretation is indeed a PDB.

Lemma 25. *For every conditional world C, the probabilistic interpretation of C is a PDB.*

7.3 Lifting Queries to Conditional Worlds

To show how conditional worlds instantiate the answer space, we need to show how queries can be lifted from complete databases to conditional worlds. For all queries, except that of inequality selections, the definitions are rather straightforward.

The intuition behind $\sigma_{\$i<\$j}(C)$ is that the result of evaluating an inequality selection on $C := \{(\mathbf{A}_1, c_1), \ldots, (\mathbf{A}_m, c_m)\}$ depends on which tuples are not filtered out. A tuple $\bar{t} := (t_1, \ldots, t_n)$ is filtered out if the condition $t_i - t_j < 0$ is not satisfied. Thus, conditions that may affect the result are the conditions in

$$C(i, j) := \{t_i - t_j < 0 \mid (t_1, \ldots, t_n) \in \cup_{i=1}^{m} \mathbf{A}_m\}$$

and each subset of satisfied conditions corresponds with a subset of the possible answers. We define the condition c_B of a subset $B \subseteq C(i, j)$ in a way it ensures that (1) all tuples that satisfy the conditions in B remains, and (2) all those that do not are filtered out. Thus, we set $c_B := B \cup \{-b \mid b \in C(i, j) \setminus B\}$. The first element in the union ensures (1) and the second (2). With this notation we can present the full definition.

Definition 26. We define the semantics $q(C)$ of queries q on conditional worlds C inductively as follows:

$$\begin{aligned} C \circ C' &:= \{(\mathbf{A} \circ \mathbf{A}', c \cup c') \mid (\mathbf{A}, c) \in C, (\mathbf{A}', c') \in C'\} \\ q(C) &:= \{(q(\mathbf{A}), c), \mid (\mathbf{A}, c) \in C\} \\ \sigma_{\$i<\$j}(C) &:= \{(\sigma_B(\mathbf{A}), c \cup c_B \mid (\mathbf{A}, c) \in C,\ B \subseteq C(i, j)\} \end{aligned}$$

with $\circ \in \{\times, \cup, \setminus\}$, $q := R \mid \pi_{\$i_1,\ldots,\$i_k}(q) \mid \sigma_\theta(q) \mid \textsc{Apply}_f(q) \mid \sum^{\$j}_{\$i_1,\ldots,\$i_k}(q)$, while setting $\sigma_\emptyset(\mathbf{A}) := \emptyset$, and all operators defined on complete databases whose active domain is extended to $\mathrm{RAT}[\bot_1, \bot_2, \ldots]$.

To show that this inductive definition is indeed well defined, it suffices to show compositionality of conditional worlds:

Lemma 27. *$q(C)$ is a conditional world for every conditional world C and query q.*

Example 1. Assume that C consists of $(\mathbf{A}, c)$ where $\mathbf{A} = \{\!\{(\bot_1, 0), (\bot_1, \bot_1), (\bot_3, \bot_1 + \bot_3)\}\!\}$. To compute $\sigma_{\$1<\$2}(C)$, we first notice that $C(1, 2) = \{\bot_1, -\bot_1\}$. There are, therefore, four subsets of $C(1, 2)$, and we obtain

$$\begin{aligned} \sigma_{\$1<\$2}(C) = \{&(\emptyset, c \cup \{-\bot_1, \bot_1\}), (\sigma_{\bot_1}(\mathbf{A}), c \cup \{\bot_1\}), \\ &(\sigma_{-\bot_1}(\mathbf{A}), c \cup \{-\bot_1\}), (\sigma_{\bot_1,-\bot_1}(\mathbf{A}), c \cup \{\bot_1, -\bot_1\})\} \end{aligned}$$

Note that each condition that contains $\{-\bot_1, \bot_1\}$ never holds. And, indeed, their corresponding databases cannot be instantiated to any of the possible answers.

7.4 Answer Spaces as Conditional Worlds

We now show that we can instantiate a probability space that is almost similar to the answer space using the lifting we presented in the previous section. But before, what does it mean to be almost similar? A probability space $(\mathrm{dom}, \Sigma, P)$ is said to be a *trivial extension* of a probability space $(\mathrm{dom}', \Sigma', P')$ if $\mathrm{dom} \supseteq \mathrm{dom}'$, $\Sigma \supseteq \Sigma'$, and $P'(\sigma) = P(\sigma)$ for every $\sigma \in \Sigma'$.

Theorem 28. *For every incomplete database $\mathbf{D}$ and query q, if each null in $\mathsf{Null}(\mathbf{D})$ has a singleton-unlikely distribution then the answer space of q over $\mathbf{D}$ is a trivial extension of $q(C)$ where $C := \{(\mathbf{D}, -1)\}$.*

The choice of C is due to the fact that its probabilistic interpretation is similar to the PDB of $\mathbf{D}$.

8 CONCLUSION

We presented a novel model for incomplete numerical data based on tuple-dependent infinite probabilistic databases. We studied query answering in a language that captures key features of SQL, looked at exact and approximate answers, and showed how to represent query outputs. As for future work, we would like to extend our model to other SQL data-types, especially discrete domains. We would like to study other forms of approximations, in particular direct sampling from the conditional world that represents the output. Such representations may be optimized deploying ideas from cylindrical algebraic decomposition [4]. From a practical standpoint, we would like to implement our theoretical algorithms using real-world RBDMSs, and test their performances with real data.

ACKNOWLEDGMENTS

This work was supported by a Leverhulme Trust Research Fellowship; by EPSRC grants N023056 and S003800; by Agence Nationale de la Recherche project ANR-21-CE48-0015 (Verigraph); by MUR under the PRIN 2017 project "HOPE" (prot. 2017MMJJRE), by the EU under the H2020-EU.2.1.1 project TAILOR, grant id. 952215, and by MUR under the PNRR project PE0000013-FAIR.

REFERENCES

[1] Abiteboul, S., Hull, R., and Vianu, V. *Foundations of Databases.* Addison-Wesley, 1995.
[2] Abiteboul, S., Kanellakis, P., and Grahne, G. On the representation and querying of sets of possible worlds. *Theoretical Computer Science 78*, 1 (1991), 158–187.
[3] Arora, S., and Barak, B. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.
[4] Basu, S., and González-Vega, L., Eds. *Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science, Proceedings of a DIMACS Workshop, Piscataway, NJ, USA, March 12-16, 2001* (2001), vol. 60 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, DIMACS/AMS.
[5] Blum, A., Hopcroft, J., and Kannan, R. *Foundations of data science.* Cambridge University Press, 2020.
[6] Blum, L., Cucker, F., Shub, M., and Smale, S. *Complexity and real computation.* Springer Science & Business Media, 1998.
[7] Box, G., and Muller, M. A note on the generation of random normal deviates. *Ann. Math. Statist. 29* (1958), 610–611.
[8] Carmeli, N., Grohe, M., Lindner, P., and Standke, C. Tuple-independent representations of infinite probabilistic databases. In *PODS'21: Proceedings of the 40th*

ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021 (2021), L. Libkin, R. Pichler, and P. Guagliardo, Eds., ACM, pp. 388–401.

[9] Console, M., Guagliardo, P., Libkin, L., and Toussaint, E. Coping with incomplete data: Recent advances. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020* (2020), D. Suciu, Y. Tao, and Z. Wei, Eds., ACM, pp. 33–47.

[10] Console, M., Hofer, M. F. J., and Libkin, L. Queries with arithmetic on incomplete databases. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020* (2020), D. Suciu, Y. Tao, and Z. Wei, Eds., ACM, pp. 179–189.

[11] Dalvi, N. N., Ré, C., and Suciu, D. Queries and materialized views on probabilistic databases. *J. Comput. Syst. Sci. 77*, 3 (2011), 473–490.

[12] Dalvi, N. N., and Suciu, D. Efficient query evaluation on probabilistic databases. *VLDB J. 16*, 4 (2007), 523–544.

[13] Date, C. J. *Database in Depth - Relational Theory for Practitioners.* O'Reilly, 2005.

[14] Date, C. J. A critique of Claude Rubinson's paper 'Nulls, three-valued logic, and ambiguity in SQL: critiquing Date's critique'. *SIGMOD Record 37*, 3 (2008), 20–22.

[15] Date, C. J., and Darwen, H. *A Guide to the SQL Standard.* Addison-Wesley, 1996.

[16] Durrett, R. *Probability: theory and examples*, vol. 49. Cambridge university press, 2019.

[17] Feng, S., Glavic, B., Huber, A., and Kennedy, O. A. Efficient uncertainty tracking for complex queries with attribute-level bounds. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021* (2021), G. Li, Z. Li, S. Idreos, and D. Srivastava, Eds., ACM, pp. 528–540.

[18] Gao, Y., and Miao, X. *Query Processing over Incomplete Databases.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018.

[19] Gheerbrant, A., Libkin, L., and Sirangelo, C. Naïve evaluation of queries over incomplete databases. *ACM Trans. Database Syst. 39*, 4 (2014), 31:1–31:42.

[20] Green, T. J., and Tannen, V. Models for incomplete and probabilistic information. *IEEE Data Eng. Bull. 29*, 1 (2006), 17–24.

[21] Grohe, M., Kaminski, B. L., Katoen, J., and Lindner, P. Probabilistic data with continuous distributions. *SIGMOD Rec. 50*, 1 (2021), 69–76.

[22] Grohe, M., and Lindner, P. Infinite probabilistic databases. *Log. Methods Comput. Sci. 18*, 1 (2022).

[23] Grumbach, S., and Milo, T. Towards tractable algebras for bags. *J. Comput. Syst. Sci. 52*, 3 (1996), 570–588.

[24] Guagliardo, P., and Libkin, L. A formal semantics of SQL queries, its validation, and applications. *Proc. VLDB Endow. 11*, 1 (2017), 27–39.

[25] Hella, L., Libkin, L., Nurmonen, J., and Wong, L. Logics with aggregate operators. *Journal of the ACM 48*, 4 (2001), 880–907.

[26] Huang, J., Antova, L., Koch, C., and Olteanu, D. Maybms: a probabilistic database management system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009* (2009), U. Çetintemel, S. B. Zdonik, D. Kossmann, and N. Tatbul, Eds., ACM, pp. 1071–1074.

[27] Imielinski, T., and Lipski, W. Incomplete information in relational databases. *Journal of the ACM 31*, 4 (1984), 761–791.

[28] Jampani, R., Xu, F., Wu, M., Perez, L. L., Jermaine, C. M., and Haas, P. J. MCDB: a monte carlo approach to managing uncertain data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008* (2008), J. T. Wang, Ed., ACM, pp. 687–700.

[29] Kennedy, O., and Koch, C. PIP: A database system for great and small expectations. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA* (2010), F. Li, M. M. Moro, S. Ghandeharizadeh, J. R. Haritsa, G. Weikum, M. J. Carey, F. Casati, E. Y. Chang, I. Manolescu, S. Mehrotra, U. Dayal, and V. J. Tsotras, Eds., IEEE Computer Society, pp. 157–168.

[30] Klug, A. C. Equivalence of relational algebra and relational calculus query languages having aggregate functions. *J. ACM 29*, 3 (1982), 699–717.

[31] Lakshmanan, L. V. S., Leone, N., Ross, R. B., and Subrahmanian, V. S. Probview: A flexible probabilistic database system. *ACM Trans. Database Syst. 22*, 3 (1997), 419–469.

[32] L'Ecuyer, P. Non-uniform random variate generations. In *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Springer, 2011, pp. 991–995.

[33] Libkin, L. Expressive power of SQL. *Theor. Comput. Sci. 296*, 3 (2003), 379–404.

[34] Libkin, L., and Wong, L. Query languages for bags and aggregate functions. *J. Comput. Syst. Sci. 55*, 2 (1997), 241–272.

[35] Loeve, M. *Probability theory.* Courier Dover Publications, 2017.

[36] Singh, S., Mayfield, C., Mittal, S., Prabhakar, S., Hambrusch, S. E., and Shah, R. Orion 2.0: native support for uncertain data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008* (2008), J. T. Wang, Ed., ACM, pp. 1239–1242.

[37] Singh, S., Mayfield, C., Shah, R., Prabhakar, S., Hambrusch, S. E., Neville, J., and Cheng, R. Database support for probabilistic attributes and tuples. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico* (2008), G. Alonso, J. A. Blakeley, and A. L. P. Chen, Eds., IEEE Computer Society, pp. 1053–1061.

[38] Suciu, D., Olteanu, D., Re, C., and Koch, C. *Probabilistic Databases.* Morgan&Claypool Publishers, 2011.

[39] Tanner, J. M. *A History of the Study of Human Growth.* Cambridge University Press, 1981.

[40] Toda, S. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput. 20*, 5 (1991), 865–877.

[41] Toussaint, E., Guagliardo, P., and Libkin, L. Knowledge-preserving certain answers for sql-like queries. In *KR* (2020), D. Calvanese, E. Erdem, and M. Thielscher, Eds., pp. 758–767.

[42] Van den Bussche, J., and Vansummeren, S. Translating sql into the relational algebra. *Course notes, Hasselt University and Université Libre de Bruxelles* (2009).

[43] Van den Dries, L. *Tame topology and o-minimal structures*, vol. 248. Cambridge university press, 1998.