1 Overview

Uncertainty is prevalent in data analysis, no matter what the size of the data, the application domain, or the type of analysis. Common sources of uncertainty include missing values, sensor errors and noise, bias, outliers, mismatched data, and many more. If ignored, data uncertainty can result in hard to trace errors in analytical results, which in turn can have severe real world implications like unfounded scientific discoveries, financial damages, or even effects on people's physical well-being (e.g., medical decisions based on incorrect data).

`survey`

	AgeRange	Zip	Gender	Commute
1	18-24	10003	M	1 hour
2	24-36	10003	M	90
3	37-45	10007	**NULL**	50 min
4	0-18	10007	F	40 min
5	**NULL**	14260	F	1
6	0-18	14260	Other	70 min

Figure 1: Example input with missing values

Example 1 *Consider the example dataset* `survey` *in Figure 1, which illustrates responses to a survey of commuters, with one row per response. Several* **NULL** *values indicate omitted responses.* `Commute` *(duration) is a free-text response and follows multiple response formats. Several of these (in red) lack units, can not be parsed, and are consequently replaced with* **NULL**. *A data scientist explores this data with the query:*

```
SELECT Zip, AVG(Commute) FROM survey WHERE Gender = 'F' GROUP BY Zip
```

SQL's **NULL** *value semantics would silently exclude rows 2, 3 and 5 when computing results.*

Dropping records that contain missing data is *one* commonly used approach to managing uncertainty in data science, but by no means the only one. For example, when missing values from one attribute are correlated with other attributes, it may be better to impute missing values instead of removing them. Classical data management systems are designed around the assumption that such data uncertainty issues have already been addressed before data is ingested into the system. While this assumption makes data easier to work with, it is often violated in practice, because (i) users may not be in a position to identify all sources of uncertainty or (ii) insufficient information or resources may be available to repair the dataset with 100% certainty. Hence, **classical analytics requires users to make a *"best-guess"* and live with it**, as in the silent approximations of Example 1. Techniques for managing incomplete data (e.g., [20, 37, 51, 57, 103]) exist, but are generally too heavy-weight for real-world usage, and/or may hide relevant information from users. For example, *certain answers* [2, 57], which are the most widely applied query semantics for *incomplete databases* exclude from query results all data that is not 100% certain.

Example 2 *Evaluating the query from Example 1 under certain answer semantics returns an empty result, because the average commute time for all three zip codes from the input are uncertain: for both zip codes 10003 and 14260 the* `Commute` *attribute value of one row (respectively rows 2 and 5) is uncertain and for zip code 10007 the inclusion of row 3 into the group is uncertain since the gender for survey response 3 may or may not be* `'F'`.

Figure 2 shows a comparison of approaches for uncertain data management based on (i) what queries they support (positive relational algebra $\mathcal{RA}^+$, full relational algebra $\mathcal{RA}$, or full relational algebra plus aggregation $\mathcal{RA}^{agg}$); (ii) the data complexity of querying; (iii) whether they identify certain answers ($\subseteq$, indicates only a subset is identified); (iv) whether they return all possible answers ($\supseteq$ indicates that a superset is returned); (v) whether they filter or prioritize the typically large number of possible answers for easier human interpretation; and (iv) whether they interoperate with standard representations of incomplete/prob. data. Determining certain answers is computationally hard [2, 56, 57], even for quite limited incomplete data models. While efficient under-approximations for certain answers [72, 94] address the performance issue, the problem of excluded possible results is further exacerbated by the under-approximation. In contrast, probabilistic databases [32, 100] provide users with a complete description of uncertainty, but this comes at an even higher cost (`#P` data complexity). `PTIME` approximations, e.g., [41], reduce the cost paid per query result tuple, but still need to enumerate all possible answers, resulting in severe scalability issues [39]. Presenting all possible answers to a user may be overwhelming, but in contrast to incomplete databases, probabilistic databases can rank answers by probability to

Approach	Supp. Queries	Data Complexity	Certain Answers	Possible Answers	Filtered Possible	Interop.
Certain Answers [57]	$\mathcal{RA}^{agg}$	coNP-hard	✓	×	×	×
Under-approx. CA [72,94]	$\mathcal{RA}$	PTIME	✓$\subseteq$	×	×	×
Probabilistic DBs [31]	$\mathcal{RA}^{agg}$	#P-hard	✓	✓	✓	×
Approximate PDBs [41,58]	$\mathcal{RA}^{agg}$‡	PTIME	✓	✓	✓	✓
UA-DB [39]	$\mathcal{RA}^{+}$	PTIME	✓$\subseteq$ + $\supseteq$	×	✓	✓
AU-DB (this proposal)	$\mathcal{RA}^{agg}$	PTIME	✓$\subseteq$ + $\supseteq$	✓$\supseteq$	✓	✓

‡:*Existing approaches only handle a restricted subset of this query class.*

Figure 2: Comparison of approaches for uncertain data management

prioritize the user's attention. Most approaches for certain answer semantics and probabilistic query processing are tailored for specific data models such as *tuple-independent databases* (TIP-DB) [32,100] where the existence of each tuple is assumed to be independent of the others. However, such models may not be appropriate for some sources of uncertainty, e.g., a TIP-DB requires an infinite number of tuples to represent an unknown value from a continuous-valued attribute (one for each domain value).

Requirements for Practical Uncertainty Management. We argue that for an uncertain data management approach to be applied in practice, (i) its performance needs to be close to that of deterministic systems. At the same time we do want to (ii) retain as much of the rigor and guarantees of incomplete and probabilistic databases as possible. Furthermore, we need to (iii) avoid the pitfalls of certain answers which may exclude too much information by being overly zealous. Finally, the approach should (iv) support a large class of queries including features like aggregation and negation and (v) should be compatible with standard representations of uncertainty to support a wide range of data sources.

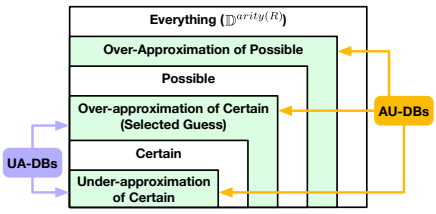

Figure 3: UA-DBs under- and over-approximate certain answers. The proposed AU-DBs additionally encode an over-approximation of possible answers.

Prior Work by the PIs. Our proposal is based on an efficient uncertain data model: **Uncertainty-Annotated Databases** [39] (**UA-DBs**), which bridge classical and incomplete data management. Rather than replacing classical data management, UA-DBs annotate existing data with *uncertainty labels* and define a lightweight semantics for propagating these labels through queries.

Example 3 *Continuing Example 2, the analyst now poses the query*

```
SELECT DISTINCT Zip FROM survey WHERE Commute > INTERVAL('60␣min')
```

	Zip
1	*10003*
2	*14260*

There are two possible answers: 14260 is a certain answer, as even though one surveyed individual from the zip code has a **NULL** *answer for commute time, at least one individual in the zip code satisfies the query's condition. Conversely 10003 is not certain. Its presence in the result depends on an unknown value (the duration corresponding to the input value '90'). However, a typical import heuristic: "Treat unspecified units as minutes" would still include 10003 in the query result. In a UA-DB, the outputs of such heuristics are included, but labeled as uncertain, and those labels are propagated through queries, e.g., the red-highlighted row in the result table.*

UA-DBs are a strict generalization of classical data management that clearly distinguishes between reliable (i.e., certain) and potentially unreliable (i.e., uncertain) data and results. As an added formal benefit, as shown in Figure 3, UA-DBs *sandwich certain answers between an under- and an over-approximation* that is preserved through queries. The PIs have already deployed these ideas into practice: UA-DBs serve as the foundation of error management in the Vizier open source data-centric notebook system [22,52], helping users to track data errors through analysis workflows. As shown in Figure 2, UA-DBs [39] fulfill requirements (i), (ii), (iii), and (v) for a practical approach towards uncertain data management [39]. However, they are only defined for positive relational algebra. Furthermore, they track uncertainty at the granularity of rows, which can lead to a loss of precision if, as is the case for Examples 1 and 2, only some attribute values of a row are uncertain.

1.1 Proposal Goals

The proposed work explores techniques that satisfy all of the uncertainty management requirements listed above. Based on this work we will build a framework called **U4U** (**Uncertainty for You**). Developing an approach that fulfills all requirements is very challenging and requires novel contributions to both the theory and systems aspects of uncertain data management. We subdivide the problem into three research thrusts focusing on: (I) Expressivity, (II) Performance vs Accuracy, and (III) Implementation Challenges, respectively.

I: AU-DBs: Beyond Positive Relational Algebra by Bounding Attribute Values. We will explore extensions of UA-DBs that allow us to cover a wider range of queries more precisely. *(a) Attribute-level Ranges*: We will extend UA-DBs with attribute-level annotations encoding bounds on the values of an attribute across all possible worlds, enabling more precise encoding of the attribute-level uncertainty inherent in many datasets. We refer to this new model as *attribute-annotated uncertain databases* (*AU-DBs*). *(b) Non-monotone Queries*: AU-DBs sandwich certain answers between an under- and an over-approximation, but under-approximations of certain answers for non-monotone queries (e.g., aggregation and set difference) require *over-approximations* of *possible* answers [53, 73]. These over-approximations will typically be very large. We propose to extend AU-DBs with an over-approximation of possible answers as shown in Figure 3. With attribute-level annotations, a single tuple can represent arbitrarily many incomplete tuples, admitting compact encodings of possible answers and flexible trade-offs between accuracy and performance. *(c) Certainty beyond Sets and Bags*: We will design AU-DBs to take advantage of the generality they inherit from the $\mathcal{K}$-relations [50] they extend.

II: Flexible Trading of Performance for Accuracy. Thrust I treats requirement (i) as a hard constraint, focusing on strategies that introduce only negligible overhead relative to classical deterministic query processing. In Thrust II, we relax this requirement, relying on larger annotations that provide increased accuracy and/or detail. Concretely, the goals of this thrust include: *(a) Sketching for Multiset DBs*: We draw a parallel to sketches [30], datastructures for approximately summarizing large datasets. Specifically, we will sketch the annotations of a tuple across all worlds and develop propagation rules for such sketches through queries. Preliminary estimates show better asymptotic behavior for sketches than sampling on selective queries. *(b) Initializing Sketches*: Typical applications of sketching assume that the input is worst-case (i.e., each datum must be visited). In contrast, the annotations of tuples across all possible worlds are usually highly redundant, suggesting the possibility of initializing the sketch in time poly-logarithmic in the number of possible worlds. To do so, we need to study k-wise independent hash functions h for which we want to be able to compute the size of any of its preimages (i.e. $|h^{-1}(y)|$ for any y in the range of h) in sub-linear time. To the best of our knowledge, this problem has not been studied so far and we propose to do so by using tools from algebraic geometry. *(c) Sketching for Set DBs*: We expect to rely on linearity of expectation for sketching multiset databases. This property does not hold for set databases, and we will need to explore alternative sketching techniques.

III: Implementing U4U. Thrusts I and II focus on a formal analysis of the complexity and accuracy of techniques for uncertainty management. Thrust III addresses the challenges that arise while translating these concepts into practice through a system called *U4U* (*Uncertainty For You*) that implements the results of the prior two thrusts. Concretely, the goals of this thrust include: *(a) Management of Uncertainty-Labeled Data*: We will develop adapters to translate from incomplete and probabilistic data models into AU-DBs. Furthermore, we propose to develop data curation and cleaning operations that directly generate AU-DBs. *(b) Query Rewriting for AU-DBs and Sketches*: Building on promising results with a prototype UA-DB [39], we will implement and evaluate AU-DBs and sketches. For AU-DBs we will also study queries with aggregation and other non-monotone operators as well as queries over other types of annotations (e.g., uncertain provenance). *(c) A Specialized Database Engine for U4U*: To further improve query performance, we will design and implement a specialized AU-DB query processing engine to exploit compact representations based on factorization and functional aggregate queries [61] (FAQ); as well as to efficiently join and group data with uncertain values. We will also investigate optimizing the trade off between more accurate (but slower) and more efficient (but less precise/accurate) execution strategies.

The Team. The PIs are uniquely qualified to conduct the proposed research. PI Glavic has extensive background in annotated databases and provenance [8, 49, 69, 90] (including a best-of-ICDE invitation to TKDE [89]). PI Kennedy has a background in probabilistic databases and uncertainty in general [59, 65, 79, 82, 105] (including a best-of-SIGMOD invitation to TODS in 2017 [79]). PIs Glavic and Kennedy are actively collaborating on topics relevant to this proposal [38, 42, 43, 82, 88, 98] and have long standing experience in developing systems in the

domain of data cleaning [12, 13], curation, uncertainty [38, 59, 105], and provenance [8, 49]. PI Rudra has a background in database theory and has studied formal aspects of annotated databases involving semirings [61, 62] (PODS 2016 [61] and PODS 2012 [84, 85] best paper awards) and compact representations of non-results in joins and #-SAT formulas [35, 60].

2 Background and Related Work

Before presenting an overview of our UA-DB framework, we introduce necessary background and related work on incomplete and probabilistic databases and the *incomplete* $\mathcal{K}$-databases that we introduced in [39].

2.1 Possible Worlds Semantics

Incomplete databases (IDBs) model uncertainty and its impact on query results. An *incomplete database* $\mathcal{D}$ is a set of deterministic database instances $D_1, \ldots, D_w$ called *possible worlds*. We write $t \in D$ to denote that a tuple t appears in a specific possible world D. Most approaches adopt the so called "possible worlds" semantics for querying incomplete databases: The result of evaluating a deterministic query Q over an incomplete database is the set of relation instances resulting from evaluating Q over each possible world individually using standard deterministic query semantics: $Q(\mathcal{D}) = \{\ Q(D) \mid D \in \mathcal{D}\ \}$. A probabilistic database (PDB) is an incomplete database paired with a probability distribution over the possible worlds. Decades of research [6, 21, 51, 57, 96, 100] have focused on algorithms for efficient query processing over IDBs and PDBs, introducing data models such as V-tables [3], C-tables [57], and tuple-independent probabilistic databases [100] that more compactly represent a set of possible worlds by factoring out commonalities.

2.2 Certain, Possible, and Selected-Guess Answers

The goal of query processing over incomplete databases is to differentiate query results that are certain from ones that are merely possible. Formally, a tuple is certain if it appears in every possible world and possible if it appears in at least one possible world [27, 57]:

$$certain(\mathcal{D}) = \{\ t \mid \forall D \in \mathcal{D} : t \in D\ \} \qquad possible(\mathcal{D}) = \{\ t \mid \exists D \in \mathcal{D} : t \in D\ \}$$

In contrast to [57], which studies certain answers to queries, we find it useful to define certainty at the instance level which is equivalent (the certain answers of a query Q over incomplete instance $\mathcal{D}$ are $certain(Q(\mathcal{D}))$). For simplicity we refer to the set of tuples that are certain in an instance as certain answers. Although computing certain answers is `coNP-hard` [3] in general, there exist `PTIME` under-approximations [53, 73, 94] that allow false negatives (some certain answers may be omitted from the query result) but guarantee that no false positives (tuples that are not certain answers) are returned. The rationale behind this choice is that it is considered less harmful to omit a certain answer than to claim that an uncertain answer is certain [54].

Selected-Guess Query Processing. As mentioned in the introduction, another approach commonly used in practice is to select one distinguished possible world as canon and ignore ambiguity in the data afterwards. Queries are evaluated solely in this world. We refer to this approach as *selected-guess query processing* (SGQP) [105]. Going forward, we refer to the selected world as the *selected-guess world* $D_{\mathcal{G}}$. For example, for PDBs one may select $D_{\mathcal{G}} = \text{argmax}_{D \in \mathcal{D}} P(D)$, i.e., the possible world with the highest probability, possibly using approximation techniques if determining this world is computationally infeasible.

2.3 Incomplete $\mathcal{K}$-relations

The UA-DB model that we extend in this proposal is based on incomplete $\mathcal{K}$-relations [39], a generalization of IDBs to $\mathcal{K}$**-relations** [50]. $\mathcal{K}$-relations annotate tuples with elements from a commutative semiring $\mathcal{K}$. A commutative semiring $(K, +_{\mathcal{K}}, \cdot_{\mathcal{K}}, \mathbb{0}_{\mathcal{K}}, \mathbb{1}_{\mathcal{K}})$ over a domain K has binary operations $+_{\mathcal{K}}$ and multiplication $\cdot_{\mathcal{K}}$ which obey certain set of equivalence laws, e.g., they are associative and commutative. Since $\mathcal{K}$-relations are functions from tuples to annotations, it is customary to denote the annotation of a tuple t in relation R as $R(t)$ (applying function R to input t).

Query Semantics. Positive relational algebra queries over $\mathcal{K}$-relations compute query result annotations by combining input tuple annotations using the semiring's addition ($+_{\mathcal{K}}$, for union or projection) and multiplication ($\cdot_{\mathcal{K}}$, for join) operations. For example, $(R \bowtie S)(t) \stackrel{\text{def}}{=} R(t) \cdot_{\mathcal{K}} S(t)$. Choosing the boolean semiring $\mathbb{B} = (\{T, F\}, \vee, \wedge, F, T)$ as our annotation domain, $\mathcal{K}$-relations model set semantics (i.e., a tuple annotated with T exists). A tuple exists in a union if it exists in either input ($\vee$), and in a join if it exists in both inputs

($\wedge$). Annotating data with elements from the semiring of natural numbers $(\mathbb{N}, +, \times, 0, 1)$), $\mathcal{K}$-relations model bag semantics. Other semirings can be used to model, e.g., provenance or access control level.

Incomplete $\mathcal{K}$-Relations. Like set-semantics incomplete databases, an **incomplete $\mathcal{K}$-databases** $\mathcal{D}$ is a set of possible worlds $\mathcal{D} = \{D_1, \ldots, D_n\}$. However, in incomplete $\mathcal{K}$-databases, the relations of each possible world D_i are $\mathcal{K}$-relations. Possible world query semantics remains unchanged, i.e., $Q(\mathcal{D}) = \{Q(D) \mid D \in \mathcal{D}\}$, and query results are sets of $\mathcal{K}$-relations. Incomplete $\mathcal{K}$-relations can be trivially extended to probabilistic $\mathcal{K}$-databases by defining a distribution $P : \mathcal{D} \mapsto [0, 1]$ over the possible worlds. Incomplete $\mathcal{K}$-relations enjoy the same generality properties as regular $\mathcal{K}$-relations (e.g., they generalize set, bag semantics, and more).

Certain and Possible $\mathcal{K}$-Annotations. While possible worlds semantics naturally extend to incomplete $\mathcal{K}$-databases, the concept of certain answers has to be re-thought. In [39] we demonstrated that for l-semirings [63], i.e., semirings that have well-defined *greatest lower bound* $\sqcap_\mathcal{K}$ and *least upper bound* $\sqcup_\mathcal{K}$ operations based on a partial order $\preceq_\mathcal{K}$, we can define the **certain annotation** of a tuple as the greatest lower bound (GLB) of the annotation of the tuple across all possible worlds. Analogously, the **possible annotation** is the least upper bound (LUB) across all worlds.

As we demonstrated in [39], for set semantics (with order $F < T$), the certain annotation of a tuple is exactly the classical notion of certainty: A tuple is *certain* if it appears in all possible worlds and *possible* if it appears in at least one possible world. For the bag semantics, certainty (possibility) is the minimum (maximum) multiplicity of the tuple across all worlds. This definition coincides with the bag semantics definition from Guagliardo et. al. [53]. Formally, the certain (possible) annotation $\text{CERT}_\mathcal{K}(\mathcal{D}, t)$ ($\text{POSS}_\mathcal{K}(\mathcal{D}, t)$) of a tuple t in incomplete $\mathcal{K}$-database $\mathcal{D}$ is defined as:

$$\text{CERT}_\mathcal{K}(\mathcal{D}, t) = \sqcap_\mathcal{K}(\{\ D(t) \mid D \in \mathcal{D}\ \}) \qquad \text{POSS}_\mathcal{K}(\mathcal{D}, t) = \sqcup_\mathcal{K}(\{\ D(t) \mid D \in \mathcal{D}\ \})$$

Example 4 *The survey table from Figure 1 admits a range of repairs. The* **NULL** *value in row 3 can be replaced by any value from the domain (F, M, O), while the unparseable values on rows 3 and 5 can be (reasonably) interpreted as seconds, minutes, or hours. There are 27 possible repairs for these two columns. However the query* `SELECT Gender FROM survey WHERE Zip <> 10003` *returns only three possible $\mathbb{N}$-databases:*

D_1:

Gender	$\mathbb{N}$
F	3
M	0
O	1

D_2:

Gender	$\mathbb{N}$
F	2
M	1
O	1

D_3:

Gender	$\mathbb{N}$
F	2
M	0
O	2

The certain multiplicity of the tuple (F) is $\text{CERT}_\mathbb{N}(\{3, 2, 2\}) = min(3, 2, 2) = 2$. Similarly, for the tuple (M) (only present in possible world D_2) we get $\text{CERT}_\mathbb{N}(\{0, 1, 0\}) = 0$, i.e., the tuple has certain multiplicity 0.

Just as in the set semantics case, we define incomplete $\mathcal{K}$-databases as sets of possible worlds to achieve a clean semantics. More compact representations are needed. For example, C-tables can be used to concisely encode any incomplete $\mathbb{B}$-database.

2.4 Uncertainty-Annotated Databases

In [39] we introduced UA-DBs as a generalization of selected-guess query processing for incomplete $\mathcal{K}$-databases. As we already alluded to in the introduction, and as shown in Figure 3, a UA-DB sandwiches the certain answers for a $\mathcal{K}$-database $\mathcal{D}$ between an under- and an over-approximation. This is achieved by annotating each tuple with a pair of annotations from $\mathcal{K}$ that bound the certain annotation of the tuple from above and below. The selected-guess world $D_\mathcal{G}$ serves as the over-approximation. We refer to the under-approximation $\mathcal{L}$ of certain answers as a *labeling*. Figure 4 shows how UA-DBs are used: an uncertain data source is translated into a UA-DB that (i) encodes one of its possible worlds and (ii) under-approximates its certain answers. The user can then run queries over this UA-DB to obtain a result encoded as a UA-DB, i.e., UA-DBs are closed under queries.

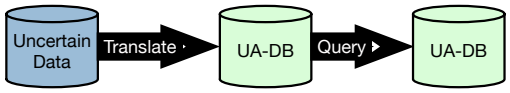

Figure 4: Creating and Querying UA-DBs

Translating Uncertain Data Into UA-DBs. In [39] we demonstrated how common incomplete and probabilistic data models can be translated into UA-DBs. We call this process *labeling* and refer to strategies that implement it as **labeling schemes**. Specifically, we presented labeling schemes for both the incomplete and probabilistic versions of TIP-DBs [100], C-tables [57], and x-DBs [103]. While this enables UA-DBs to be derived from

existing incomplete data models, it is often possible to derive a UA-DB directly from inconsistent or dirty data without going through an intermediate incomplete database representation first. For example, consistent query answering [10,16,25,34,47] computes certain answers to queries wrt. to all repairs of a database that violates a set of integrity constraints. However, in contrast to consistent query answering, UA-DBs are constructed from a database directly, and are not specific to any given query.

Example 5 *Continuing Example 4, we create a bag-semantics UA-DB (semiring $\mathbb{N}$) by first choosing one distinguished possible world $D_\mathcal{G}$. For this example, we adopt the following heuristics: commute lengths without units are considered to be hours and unknown genders are assumed to be female. We mark tuples that vary across possible worlds as uncertain. The UA-DB for Figure 1 follows. Tuples 1, 4, and 6 are known to be certain and, thus, annotated with $[1,1]$ (1 certain instance, 1 selected-guess instance). The remaining tuples are not guaranteed certain and annotated $[0,1]$ (0 certain, 1 selected-guess).*

	AgeRange	Zip	Gender	Commute	$\mathbb{N}^2$
1	18-24	10003	M	120 min	**[1,1]**
2	24-36	10003	M	5400 min	**[0,1]**
3	37-45	10007	F	50 min	**[0,1]**
4	0-18	10007	F	40 min	**[1,1]**
5	**NULL**	14260	F	60 min	**[0,1]**
6	0-18	14260	Other	70 min	**[1,1]**

Efficient Bound-Preserving Querying of UA-DBs. An important feature of UA-DBs that we have demonstrated in [39] is that they are closed under a simple and efficient query semantics for positive relational algebra, and that this query semantics preserves the bounds on certain answers encoded by the UA-DB. That is, if we evaluate a $\mathcal{RA}^+$ query Q over a UA-DB D that bounds the certain tuples for an incomplete database $\mathcal{D}$, then the result bounds the certain answers for Q over $\mathcal{D}$. The query semantics we are referring to evaluates the query independently over the two annotations of each tuple in the UA-DB using standard $\mathcal{K}$-relational query semantics, which are known to be efficient. Recall that $\mathsf{CERT}_\mathcal{K}(\mathcal{D},t)$ denotes the certain annotation of a tuple in an incomplete $\mathcal{K}$-database $\mathcal{D}$. Given a UA-DB D for $\mathcal{D}$ and query Q, let the annotation of a tuple t in $Q(D)$ be $[l,u]$, i.e., $Q(D)(t) = [l,u]$. Then,

$$l \preceq_\mathcal{K} \mathsf{CERT}_\mathcal{K}(Q(D),t) \preceq_\mathcal{K} u$$

Example 6 *Consider the query* `SELECT Gender FROM survey WHERE Commute >= INTERVAL('60␣min')` *over the UA-DB from Example 5. $\mathcal{K}$-query semantics annotates each tuple in the result of a projection with the sum of the annotations of all inputs projected onto it. Applying this semantics to each dimension of the UA-DB annotations individually, we get the result shown to the right. Gender female in the selected-guess world, but may may not be in the result over all other possible worlds. Thus, its certain multiplicity is 0. Gender male appears twice in the selected-guess world. Here, the (approximated) result only guarantees one certain copy, even though both copies appear in all worlds. However, the result still "sandwiches" the row's certain multiplicity.*

Gender	$\mathbb{N}^2$
M	**[1,2]**
F	**[0,1]**

3 Thrust I - Extending UA-DBs

As explained in Section 2.4, UA-DBs [22, 39] combine a selected-guess world, i.e., a single distinguished possible world, with an *under-approximation* of certain answers to bound certain answers from above and below. Furthermore, we did extend certain and possible answers for incomplete databases to support many new semantics expressible as semiring-annotated relations, including bag semantics, various types of provenance, access control, time [33], as well as other extensions of the relational model.

In the proposed work, we will significantly enhance UA-DBs by extending them with attribute-level uncertainty, where attribute values are annotated with intervals to bound the values the attribute can take across all possible worlds. This will enable more precise bounds on certain answers in many practical scenarios. For example, without attribute-level uncertainty, missing value imputation must mark the entire tuple as uncertain. We will then further extend this model to deal with non-monotone queries which necessitates capturing an over-approximation of all possible answers. This over-approximation additionally provides the user with an overview of what possible data could be missing from the selected-guess world.

3.1 I-a: Bounding Values for Attribute-Level Uncertainty

To enable more precise encoding of uncertainty, we will extend the UA-DB model with attribute-level annotations, and study how these annotations propagate through queries.

Like many existing models for incomplete and probabilistic databases, our preliminary work on UA-DBs tracks uncertainty at the row-level. However, as repeated efforts have shown [7, 58, 59, 82, 96, 103], tracking uncertainty at the attribute-level can lead to more concise and precise representations of uncertainty.

We propose to enhance the UA-DB model with attribute-level annotations that encode lower and upper bounds on the values of attributes across all possible worlds. For attribute domains without a meaningful order, we can use boolean markers (uncertain or not) instead. We call the resulting model **attribute-annotated uncertain databases** (**AU-DBs**). Note that in contrast to models that allow variables as values, such as Codd-tables or V-tables in which a variable is a universal statement about the existence of worlds (the attribute takes every possible value in at least one possible world), AU-DB annotations bound the value of the attribute across all worlds (in every world the attribute's value is within the provided bounds). In other words, AU-DBs are approximations, enabling efficient query evaluation over this model, while preserving the bounds' correctness. The concept of over- and under-approximations of certain answers extends naturally to AU-DBs, but requires us to now reason about the certainty of tuples that do not have the same values in every possible world. We use $c^{[l,u]}$ to denote a value that is c in the SGW, annotated with a bound $[l, u]$ and c^* to denote a value that is marked as uncertain using a boolean marker.

Example 7 *Revisiting our running example from Figure 1, observe that the existence of all rows and most of their attribute values are certain. In contrast to UA-DBs where only half of the rows are identified certain, in AU-DBs, all tuples are marked certain and, four attribute values are marked uncertain: the missing* Gender *and* $\mathit{AgeRange}$ *values and the two unit-less* $\mathit{Commute}$ *values. With simple heuristic assumptions (e.g.,* $1 \leq \mathit{Commute} \leq 240$ *or* $0 \leq \mathit{Age} \leq 123$*), we can bound the uncertainty of missing values. Furthermore, age is only provided as coarse-grained ranges, but can be recast to a granularity of years by applying a heuristic to extract ages from rows, e.g., by correlating this table with a distribution of population per gender and zip code. While these resulting age values are uncertain we can bound their value across all possible worlds based on the original age ranges as shown below.*

	AgeRange	Zip	Gender	Commute	$\mathbb{N}^2$
1	$21^{[18,24]}$	10003	M	120 min	**[1,1]**
2	$30^{[24,36]}$	10003	M	90 min$^{[1\text{ min},240\text{ min}]}$	**[1,1]**
3	$38^{[37,45]}$	10007	F^*	50 min	**[1,1]**
4	$16^{[0,18]}$	10007	F	40 min	**[1,1]**
5	$21^{[0,123]}$	14260	F	60 min$^{[1\text{ min},60\text{ min}]}$	**[1,1]**
4	$16^{[0,18]}$	14260	O	70 min	**[1,1]**

Note that above we could have chosen coarser ranges for `Commute` and `Age` without affecting correctness. Intuitively, tighter bounds are more precise and thus preferable. We plan to formalize this concept into a measure for how precisely an AU-DB represents an incomplete database. For instance, this measure may take the form of a partial order that models whether a tuple with attribute-level annotations "dominates" another tuple with attribute-level annotations.

3.2 I-b: Non-Monotone Queries with Compact Encodings of Possible Answers

Bounding certain answers for non-monotone queries, e.g., aggregation or negation, is challenging since non-monotonicity requires tracking of an over-approximation of all possible answers in order to compute an under-approximation of certain answers. We propose to extend AU-DB tuple-level annotations to record such an over-approximation compactly.

For an uncertainty management approach to be practically relevant, it needs to support common query features such as aggregation and negation (difference). However, queries using these features may be non-monotone. As observed in [54], to compute an under-approximation of certain answers for such queries, it is necessary to maintain an over-approximation of possible answers. Simply enumerating all possible answers is not an option since the number of possible answers can be enormous or even infinite, e.g., if the value of an attribute with a continuous domain is unknown. As shown in Figure 3, we propose to further extend AU-DBs with yet another tuple-level annotation that records an upper bound on the annotations of tuples represented by the tuple across all worlds. Similar to m-tables [101], this enables us to use a single tuple in the representation

to encode any number of possible tuples. We observe that for m-tables, which generalize C-tables (to represent incomplete databases of unbounded size), determining certainty and possibility is hard. In contrast, we will emphasize performance and usability at the cost of providing only *bounds* on certain and possible answers.

We will investigate query semantics that preserve the bounds on certain and possible answers encoded by AU-DBs. Compared to UA-DBs, AU-DBs enable more flexible trade-offs between precision and performance, by allowing more compact, less precise representations that group multiple (similar) uncertain tuples into one; More aggressive grouping leads to smaller representations and better performance, at the cost of less precise results. We will explore multiple query semantics within this solution space and study their bounding properties. Aggregation over probabilistic and incomplete databases is notoriously difficult [4, 40, 68, 81, 93], e.g., just determining whether a specific aggregation result is possible is NP-hard. Computing bounds on aggregated values is simpler, e.g., the sum of the lower bounds is a lower bound on the sum. However, group-by is challenging, since tuples with uncertain attribute values may belong to different groups in different possible worlds.

Example 8 *The following query returns the highest age of short-range commuters by gender.*

```
SELECT MAX(Age), Gender FROM survey WHERE Commute > INTERVAL('45 min') GROUP BY Gender
```

On the AU-DB from Example 7 the resulting AU-DB shown below. Gender (resp., Commute) on the third (resp., fifth) input row is uncertain, so (i) the group may not be in the result, and (ii) the age bounds (e.g., the upper bound of 45) for this row affect the male group, as it could possibly be grouped there as well.

max(age)	Gender	$\mathbb{N}^3$
$30^{[18,45]}$	M	[1,1,1]
$38^{[0,123]}$	F^*	[0,1,1]

In addition to bound-preserving query semantics, we will also investigate how to translate existing representations of uncertain data into the AU-DB model to preserve one of the major advantages of the original UA-DB model — its compatibility with many other models of uncertainty. For many models, it will be possible to trade between size of the representation and accuracy.

3.3 I-c: Incomplete Databases and Certain Answers Beyond Sets and Bags

AU-DBs admit a natural extension of incomplete databases and certain answers beyond sets and bags. We will study these these non-traditional cases.

Incomplete $\mathcal{K}$-relations [39] generalize incomplete data and certain answers beyond sets and bags. This opens up new use cases such as uncertain provenance, where we keep track of which parts of the provenance of a data item are certain; or uncertain fine-grained access-control where a query result can be exposed to a user if its *certain* confidentiality level is one that the user is allowed to see. We will study how the choice of semiring affects the precision of our approximation of certain answers. Furthermore, we will investigate novel applications. For instance, the aforementioned incomplete databases with access control annotations would enable a rigorous treatment of access control over the inherently uncertain result of information extraction, data cleaning, or data wrangling. For many semirings, the result of aggregate queries necessitates attribute values that are symbolic expressions. These symbolic values can often not be resolved into concrete domain values [39, 40]. To encode bounds on such relations as AU-DBs we need to allow for attribute bounds that are symbolic expressions. For instance, an example of an aggregation result value in the provenance polynomial semiring is ${x_1}^2 \otimes 3 \oplus_{sum} x_2 \cdot_{\mathbb{N}[X]} x_3 \otimes 5$ which intuitively means that the symbolic expression is the sum of 3 and 5 subject to semiring polynomial expressions x_1^2 and $x_2 \cdot_{\mathbb{N}[X]} x_3$. This expression can be bound from below using, e.g., $x_1 \otimes 3$.

4 Thrust II - Sketching Incomplete and Probabilistic Databases

From an abstract view point, UA-DBs can be described as follows: we annotate tuples with measurements taken over the space of possible worlds — for example the upper and lower bounds of a tuple's annotations across all worlds — and ensure that these annotations can be propagated through extensional query evaluation. Through Thrust I, we focus on ensuring query performance competitive with deterministic databases by limiting the number of annotations. In Thrust II, we relax this constraint by allowing larger annotations that can produce more detailed, precise results. For example, a use case may require characterizing not only the range of possible attribute values or row counts, but the distribution of values (e.g., via one or more statistical moments).

Concretely, we ask how adding more measurements to the tuple annotation can help us to obtain a more precise view of a query's output. One strategy would be to annotate tuples with more than one selected-guess

world. This mirrors a classical approach to computing statistical moments: sampling (e.g., as in MCDB [14, 58], Pip [59], or MayBMS [6]). However, aside from specific exceptions (e.g., [14, 59]), sampling accuracy drops rapidly as data becomes sparse (e.g., as a result of selective predicates).

In Thrust II we will investigate sketching [30] as an unexplored alternative to sampling from probabilistic databases. Sketches are a family of data structures encoding approximate statistics about a dataset. Details vary, but each sketch collects measures of a dataset that can be used to estimate properties of interest (e.g., top-k or group-by counts). Generally, for one or more functions of interest (f), a sketch defines efficiently computable functions $\texttt{sketch}$, $\texttt{estimate}_f$ and a fold operation $\oplus$ such that for records $\vec{v}_1 \ldots \vec{v}_N$:

$$\texttt{estimate}_f\left(\texttt{sketch}(\vec{v}_1) \oplus \ldots \oplus \texttt{sketch}(\vec{v}_N)\right) \approx f(\vec{v}_1, \ldots, \vec{v}_N)$$

Sketching has previously been explored in the context of probabilistic data [29], but with one important difference. Our aim is not to sketch *the data* (which is often worst-case), as in classical applications of sketches, but rather to annotate each tuple with a sketch of its *possible annotations* (which is structured) across all possible worlds. For this to be practically relevant, we need to develop a query evaluation semantics for data annotated with sketches (Goals II-a, II-c). We focus on bag-relational databases first (Goal I-a), developing sketches with additive and multiplicative combiners that act, in expectation, as a semiring. We next explore how such sketches can be efficiently constructed, taking advantage of structured probabilistic data encodings like TIP- or BI-DBs[1] (Goal II-b). Finally, we revisit evaluation semantics in the context of set-semantics databases (Goal II-c).

4.1 II-a: Sketches for $\mathbb{N}$-annotated Databases

We will develop sketching techniques suitable for compactly encoding bag-annotations (i.e., $\mathbb{N}$-annotations). The resulting sketches will be suitable for approximating expectations and other statistical moments, and for generating approximate samples.

Assume we have W possible worlds indexed as $w \in [1, W]$. Let t be a tuple and $\vec{v}_t$ denote the vector of its $\mathbb{N}$-annotations across all possible worlds: $\vec{v}_t[w]$ is the annotation of t in world w. For example, for a TIP-DB with N rows there are $W = 2^N$ possible worlds, the possible worlds can be indexed by N bits $[1, W] = \{0, 1\}^N$, where each of the N bits corresponds to an tuple in the input. Further, in $\vec{v}_t$ the bits are set in the 2^{N-1} worlds where the tuple t is present. Our approach will be based on the Count sketch [26], which, in our setting is defined by two hash functions: a binning function $h : [1, W] \to [1, B]$ for B buckets, and a sign function $s : [1, W] \to \{\ 1, -1\ \}$.

In its simplest form, for our use-case, a sketch is a B-length vector $\mathbb{S}$ computed as $\mathbb{S}[j] = \sum_{w:h(w)=j} \vec{v}_t[w]s(w)$. We emphasize a key difference from classical sketching in databases: We are binning by the hash of the *position* of each vector element and not the element's value. The insight behind the Count sketch is that, assuming $s(w)$ is pairwise independent, $\mathbb{S}[h(w)] \cdot s(w)$ is an unbiased estimator for $\vec{v}_t[w]$ (that is, the tuple annotation in world w). This estimator has (extremely) high variance, which is reduced by taking the median of repeated independent trials. The sketch is then an $B \times N$ matrix of B bins for N trials.

Our goal is to develop a variant of the Count sketch that behaves (at least in expectation) as a semiring, allowing us to rely on $\mathcal{K}$-relational query evaluation semantics. To accomplish this, we first need additive and multiplicative combinators. This sketch naturally supports an additive combinator: pointwise addition. Adding $\mathbb{S}_1 = \texttt{sketch}(\vec{v}_1)$ and $\mathbb{S}_2 = \texttt{sketch}(\vec{v}_2)$, we get $\mathbb{S}_1 \oplus S_2 = \texttt{sketch}(\vec{v}_1 + \vec{v}_2)$. Multiplicative combinators pose a greater challenge. Naively applying pointwise product :

$$(\mathbb{S}_1 \circ \mathbb{S}_2)[j] = \left(\sum_{w:h(w)=j} \vec{v}_1(w)s(w)\vec{v}_2(w)s(w)\right) + \left(\sum_{w \neq w':h(w)=h(w')=j} \vec{v}_1(w)s(w)\vec{v}_2(w')s(w')\right)$$

The right-hand side term contains $s(w) \cdot s(w')$, which is zero in expectation if $w \neq w'$, and thus does not bias the estimate. However, on the left-hand side, the sign vector terms cancel out (i.e., $s(w) \cdot s(w) = 1$), and the result is no longer an unbiased estimator of possible annotations $\vec{v}_1 \circ \vec{v}_2(w)$. The resulting matrix can be used to estimate the sum (and thus average) of the annotation vector, further products bias even this.

[1] Block-independent databases [32, 100]

We observe that both problems can be avoided by ensuring that sketches multiplied (resp., added) together were constructed using distinct (resp., identical) sign vectors, thus, guaranteeing sketches to be unbiased estimators[2]. A central challenge will be to frame the resulting structure as a semiring.

Our preliminary exploration of this approach is promising, suggesting that sketches can achieve a fixed *absolute* ϵ-error with memory scaling as $O(\sqrt{W})$ (as opposed to $O(W)$ for sampling). This makes sketches compelling for queries with highly selective predicates or data where each tuple has many alternatives, because for such sparse data many samples are required to achieve appropriate accuracy.

4.2 II-b: Initializing the sketches

We will develop way of initializing the sketches in sub-linear time while exploiting the structure in the annotation vector. This leads us to a fundamental and natural problem about hash functions that surprisingly seems to not have been studied before. This problem has some deep connections to algebraic geometry.

Using sketches to summarize vectors (e.g. computing their norms or computing the inner product of two vectors) has been studied for a long time now [30], and is somewhat similar to our situation. However, there is a big difference between existing streaming work and our proposed work. As we have already mentioned, in most existing work the vector $\vec{v}$ is an *arbitrary vector* while in our case it is *very structured*. In the case of an arbitrary vector $\vec{v}$, an algorithm has to scan the vector *at least once* (and indeed in streaming algorithms exactly once). However, an algorithm that runs in time linear in the length of the vector will be linear in the *number of possible worlds*, which is prohibitively expensive for us. Ideally, we would like algorithms that run in time *poly-logarithmic* in the number of possible worlds. It turns out that the sketches are already of this size and indeed if we are able to initialize these sketches in poly-logarithmic time, then the rest of the operations (which involve combining various sketches) will also have a similar runtime bounds. This leads to the main theoretical question of this section: How to exploit the inherent structure of the annotation vectors to initialize their sketches in poly-logarithmic (in the number of possible worlds) time. We note that for streaming use cases where vectors have a succinct description (instead of being worst-case), e.g. work on compressed sensing deals with sparse (and other structured) signals [36, 48], initializing the sketch has been shown to still take at least linear time.

To illustrate the technical richness of this question let us consider the following simple version of the initialization problem where the vector of annotations is binary (i.e., has values 0 or 1). In this case we have hash functions $h : [1, W] \to [1, B]$ and $s : [1, W] \to \{-1, 1\}$ and for every $1 \leq j \leq B$, we want to efficiently compute the *bias* of the Hadamard product $\vec{v} \circ \vec{s}$: $\sum_{w \in [1,W]: h(w)=j} \vec{v}(w) \cdot s(w)$

$$= |\{w : h(w) = j, s(w) = 1\} \cap \{w : \vec{v}(w) = 1\}| \; - \; |\{w : h(w) = j, s(w) = -1\} \cap \{w : \vec{v}(w) = 1\}| \qquad (1)$$

Even in the simplified case where the hash function h maps all worlds to j, we still have to compute the *bias* of the hash function s. Furthermore, it turns out that to make the error analysis go through, we need s to be k-wise independent (for $k \geq 2$). Our application domain is frequently structured. Thus, we have to solve the following problem: Given a k-wise hash function family that maps w to $\{-1, 1\}$ can we compute the bias of any given specific hash function from this family in time poly-logarithmic in $W = 2^N$? Note that usually, we would like the evaluation of a k-wise independent hash function to run in a similar time. We are now asking if we can also "invert" such hash functions quickly. To the best of our knowledge, this natural inversion question about limited independence hash function families has not been studied before.

We have some preliminary observations on the above problem. First, we can pick the family of hash functions: so for k-wise independent hash functions, we studied the well-known family of hash functions defined by the so called *binary dual BCH* codes [75]. It turns out that via Hilbert's Theorem 90, this reduces to a question about counting the number of solutions to certain algebraic curves over finite fields with characteristic 2. Somewhat amazingly, this latter problem is already solved [67]. However, there are still many questions that need to be answered. The algorithm mentioned above uses very highly sophisticated algebraic geometry tools, which we not quite understand that well. One of our goals is to instead design simpler algorithms that can be implemented in our framework. It turns out we can solve this problem with a simple linear-time dynamic program for $k \leq 7$. However, designing simple implementable algorithm for larger (but still constant) values of k is a challenging task.

[2]The sign vector used to "construct" the product of two sketches is the pointwise (Hadamard) product of their sign vectors.

A further generalization of the problem above is to consider a “weighted" version of the problem. In particular, one can think of computation in (1) as computing $E_w[v(w)s(w)]$, where $w \in [1, W]$ is chosen uniformly at random. In probabilistic databases where a distribution is given over w, we would like to compute instead $\sum_w v(w)s(w)p_w$ with world w being assigned probability p_w. Note that (1) corresponds to the case of $p_w = 2^{-N}$ for every w. We do not know if the algebraic geometry result can be adapted to work for this general case. We propose to tackle this challenge as part of our project.

4.3 II-c: Sketching Non-Linear Domains

We will develop sketching techniques suitable for compactly encoding set- (i.e., $\mathbb{B}$-) and other $\mathcal{K}$-annotations. The resulting sketches will then be used to approximate expectations and other statistical moments, and for generating approximate samples.

The additive combinator for the count sketch relies on linearity of expectation. The Boolean semiring used to model set semantics is non-linear, and thus requires a different approach. As before, we plan to design a sketch with additive and multiplicative combinators which can handle idempotent addition and multiplication. A natural adaptation of our earlier approach is to base set-relational sketches on the Count Min sketch [28], which omits the sign vector from sketch construction. The resulting sketches are no longer unbiased estimates of individual values, but can be used as upper bounds on those values, for example as a way to efficiently approximate top-k-by-count queries. The Count Min sketch also naturally generalizes the bounding techniques we explore in Thrust I, albeit providing a more detailed description of the space of possible worlds. Additive and multiplicative combinators are already well defined for Count Min, so the primary challenge will be to estimate the variance of the resulting estimates, and to select optimal parameters (i.e., B and N). We will also explore the use of (sub-linear time) group testing [86] that is specifically developed to recover sparse vectors from measurements over the Boolean semi-ring. We have done work on computing functions over (non-necessarily sparse) vectors in this setup that would be a useful starting point in extending our results from Thrusts II-a and II-b to the Boolean semiring case [15]. We note that this sketch will work best on sparse inputs — tuples that are present in few worlds; We will also explore how to account for tuples that are present in more worlds.

5 Thrust III - U4U Design and Implementation

Thrust III addresses the challenge of translating the principles developed in Thrusts I and II into practice through a prototype AU-DB called *U4U* (*Uncertainty for You*). Figure 5 shows an overview of the proposed system. Data from uncertain sources is loaded using an import module based on the source's format (Goal III-a). During import the data is translated into an AU-DB and/or sketch-annotated database and encoded as relational data in a database backend. U4U also will maintain metadata like the fraction of certain tuples per table, e.g., to facilitate query optimization. Queries are executed on the database backend (Goal III-b), eventually using specific extensions like specialized join or grouping operators (Goal III-c) integrated into the backend. The Vizier notebook system [22,52], being developed in a separate project, already relies on UA-DB techniques for uncertainty tracking in data curation and analysis workflows, and stands to directly benefit from U4U.

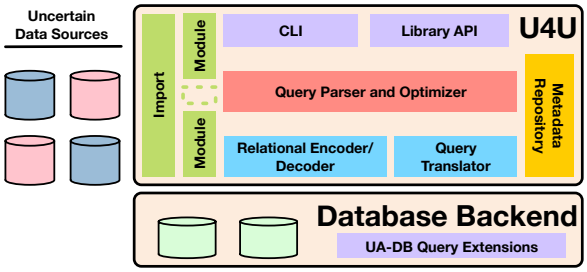

Figure 5: U4U — Overview

5.1 III-a: Management of Uncertainty Data

We will develop import modules for creating AU-DBs and sketches.

Our first goal explores techniques for creating uncertainty-annoted data.

Import Modules for Incomplete and Probabilistic Data. Numerous compact representations exist for incomplete [57], probabilistic [51,100], and fuzzy [20,106] data. We will implement import modules for AU-DBs, extending our prior work on UA-DBs [39] and from Goal II-b, to translate TIP-DBs, x-DBs, and C-tables into AU-DBs or

sketch-annotated databases. For instance, in [39] we translated a TIP-DB into a UA-DB by deriving a selected-guess world from all tuples with $p > 0.5$, and marking all tuples with $p = 1$ as certain. AU-DBs specifically pose another challenge: encoding possible tuples (e.g., for over-approximating possible answers). Creating import modules will require trading off between space/performance and precision. For maximal precision, every distinct possible tuple maps to separate rows in the encoding (as MayBMS [6]). For maximal performance, attribute-range annotations encode all possible tuples as a single row in the encoding (similarly to how in m-Tables [101] one tuple can encode multiple tuples). The "best" encoding likely lies between these extremes.

Import Modules for Curation and Cleaning. The result of data curation and cleaning operations [22, 105] is often uncertain because of the difficulty of distinguishing errors from legitimate data [1, 55, 76, 92, 102]. Even when errors are known with certainty, typically we lack sufficient information to decide how to correctly repair them [46, 95]. Most cleaning and curation systems, with notable exceptions in Lenses [22, 105] and Beskales et al. [17–19], do not directly expose this uncertainty. We will extend Lenses with support for AU-DBs and sketch annotations. For constraint-based data cleaning, a promising starting point is to extend (approximate) consistent query answering techniques [16,25,34,47] (i.e., computing answers consistent with all possible repairs of constraint violations) to produce AU-DBs. As above, a key challenge will be constructing over-approximations.

5.2 III-b: Rewrite-based Query Processing Over UA-DBs and Sketches

We will develop rewrite-based techniques for processing queries over sketches and queries with aggregation and other non-monotone operators over AU-DBs, ensuring that these queries run efficiently on existing relational databases.

In [39] (drawing on prior work [82,105]), we developed query rewriting middleware implementing a bag semantics UA-DB with tuple-level uncertainty over a classical relational database. As the first stage of realizing U4U, we will extend this middleware with support for: (1) attribute-level uncertainty and over-estimations of possible answers, (2) aggregation and other non-monotone operations, and (3) sketches.

Efficiently Supporting Attribute-Level Uncertainty. We will translate the theory developed in goal I-c into practice by realizing it within U4U. For example, a naive implementation of AU-DB attribute bounds would be to extend an existing uncertain relation $R(a_1, \ldots, a_N)$ with $2N + 3$ attributes a_i^+ and a_i^- (attribute bounds), and $n^\downarrow$, n^d, $n^\uparrow$ (tuple multiplicity and bounds). We will identify, implement, and evaluate more efficient bounds encodings. For example, attributes that are always certain do not need bounds. Alternatively, bounds (e.g., $150^{[140,160]}$) might be encoded in fewer bytes as deltas (e.g., $[150, -10, 10]$).

To control the size of the possible answer over-approximations encoded by an AU-DB, we may have to compact intermediate results, replacing multiple tuples with a single tuple whose attribute-level bounds contain all replaced tuples; This reduces size and improves performance, at the cost of reduced precision.

Supporting Aggregation. Aggregate and other non-monotone queries introduce an additional layer of complexity. For example, consider the query: `SELECT SUM(Commute) AS total FROM survey`. If `Commute` is non-negative, then the sum can be bound from below by summing up the products between the lower bound of `Commute` values ($\texttt{Commute}^-$) with the under-approximation of their multiplicity ($n^\downarrow$):

```
SELECT SUM(Commute * n^d) AS total, SUM(Commute^- * n^↓) AS total^-,
       SUM(Commute^+ * n^↑) AS total^+, 1 AS n^↓, 1 AS n^d, 1 AS n^↑ FROM survey
```

For summation over an attribute with negative values, the logic is slightly more complex, but tractable. The same approach generalizes to other decomposable functions, e.g., `min`, `max`, and `count`. Propagating bounds is harder for group-by queries, where result tuples may no longer be certain (e.g., if all tuples in a group are uncertain), and group membership now plays a role in the certainty and bounds for aggregate values (e.g., if grouping attributes are uncertain). We will initially explore two solutions. First, we will consider tracking simple schema-level properties like whether a best-guess relation is "complete" (i.e., whether it consists of exactly the possible tuples of the relation), or complete for specific groups. Second, we will consider evaluating aggregates in multiple passes — If most input data is certain, then it may be more efficient to partition the input into uncertain and certain fragments and independently process each.

5.3 III-c: Database Engine Specializations for AU-DBs and sketches

We will identify, realize, and evaluate new data-structures, algorithms, and optimization techniques to improve the performance of query processing over AU-DBs and sketches.

The third goal of this thrust focuses on architectural changes for the backend database: new algorithms, data structures, and optimization techniques that improve performance and utility on uncertain queries.

Specialized Join and Group-By Operators. To preserve an over-approximation of possible answers when evaluating join operations over AU-DBs, we need to join tuples if their attribute bounds overlap. This is essentially a spatial join where each tuple represents a box in a multidimensional space. Spatial [11, 23, 24, 77, 78] and temporal join techniques [45, 91] could prove to be effective. In contrast to spatial data, for AU-DBs, often many values are certain (points instead of boxes), which can be exploited to design join algorithms that are hybrids between regular and spatial joins. Group-by operators pose similar challenges [74].

TETRIS for Over-Approximating Possible Answers. One approach to managing possible, but not selected-guess answers is to use dual queries [53] to identify tuples that are certainly not in the result. This approach produces large intermediate results; often most of the active domain [71]. The TETRIS join algorithm [35, 60] avoids large intermediate states by incrementally ruling out progressively larger regions of the space of output tuples, and may serve as a basis for efficient "any-size" bounding of dual query results.

Incremental Refinement of Uncertainty Representations. The larger annotations of Thrust II result in larger overhead for query processing. This overhead is unnecessary when tuples/attributes are mostly certain. We will explore the use of UA-DBs and AU-DBs as a pre-filtering mechanism for more heavyweight, precise techniques like sketch annotations. Only results labeled as uncertain and or possible require heavyweight probabilistic query processing for e.g., exact certainty, probabilities, expectations, and other measures.

6 Intellectual Merit

The proposed research generalizes incomplete databases, probabilistic databases, and certain answers beyond set semantics. Importantly, we develop methods for efficient approximation of uncertainty with strong theoretical guarantees. In addition to practical applications like bag semantics (SQL), access control under incompleteness, and incomplete provenance, this also will open up new research opportunities for studying the behavior of new probabilistic data models including UA-DBs, AU-DBs, and sketch-annotated databases. The proposed work will lay down a solid formal foundation for future research by investigating approximation of certain answers for large classes of queries, diverse annotation domains, and attribute-level uncertainty. All of the expected results are also interesting research contributions in their own right. The inherent low complexity of query evaluation for AU-DBs and their backward compatibility with selected-guess query processing makes it easy to employ them in a wide range of use cases that were, until now, limited to selected-guess query processing only. Applications currently relying on selected-guess query processing can now benefit from the additional trust provided by (an under-approximation of) certain answers, without the performance penalty or the risk that useful answers might be omitted. Furthermore, applications that already rely on existing incomplete or probabilistic data models can benefit directly from AU-DBs thanks to our proposed adapters. The algorithms and methods developed in the second research thrust and their implementation in U4U will provide a pathway to making incomplete databases more practical, while laying the groundwork for future research in *pragmatic* uncertainty management.

7 Broader Impacts

The state of the art in uncertainty management is what we call selected-guess query processing (SGQP): A human or ingest process selects one possible world and proceeds with analyses as if that world was fact. In other words, uncertainty is resolved up-front in an ad hoc manner and then ignored afterwards. As we showed in [105] this model is applied in ETL (Extract-Transform-Load) pipelines. For instance, when imputing missing values as part of an ETL workflow, **NULL** values in a column are often arbitrarily replaced (with a classifier), or simply removed (Figure 1). Most data curation and cleaning operations admit multiple alternative repairs, and although repairs are chosen heuristically, they are still inherently uncertain — even if standard practice is presently to discard this uncertainty. AU-DBs have the potential for great practical impact since they combine the practicality and performance of SGQP with the rigor of certain answers. In fact the version of UA-DBs as described in [39] has been successfully implemented in the Vizier [22, 52] data-centric notebook system to track uncertainty in data curation and exploration workflows. Our proposed techniques can significantly improve many real world use cases, where decisions based on uncertain data can result in severe negative impacts. In addition to the potential of the proposed research itself, this grant will support three Ph.D. students and one postdoctoral researcher.

Integration of Research and Education. All PIs are dedicated to integrate research into education. Beyond the involvement of undergraduate and master students though summer internships, research projects, and master theses, we will incorporate uncertainty and AU-DBs into related classes at SUNY Buffalo and IIT. For instance, PI Glavic is teaching a graduate-level course on data integration, cleaning, and provenance. A module covering incomplete databases will be integrated into this course. Similarly, PI Kennedy teaches a graduate-level course that uses research-inspired projects to teach databases and programming languages concepts to students. To date, four projects have involved probabilistic or incomplete databases. Because of the severe real world impact that uncertainty can have when not handled properly, we believe that awareness of how uncertainty can affect analysis results should be a core part of early data science curricula. The PIs will incorporate this topic into graduate and undergraduate database and data science courses, demonstrating by example the impact of uncertainty on the trustworthiness of query results.

Dissemination of Research Results. Results will be published in top-tier database conferences such as SIGMOD, VLDB, PODS, ICDE, ICDT, and EDBT. The PIs will maintain a webpage for the project to keep track of U4U users and inform them about new releases and features.

Minority and Undergraduate Involvement. We are committed to recruiting and mentoring minorities. The PIs have been involved in efforts to foster minority involvement and we will use this proposal to further these efforts. A detailed description can be found in our *Broadening participation in computation plan*.

Technology Transfer and Software Tools. Our team has an strong record of translating research results into practice. The software we will develop (i.e., the U4U system) will be thoroughly documented and released as open source. Furthermore, AU-DB techniques and the sketching methods developed in this proposal will additionally be integrated into our Vizier system [22, 52]. As such they will aid a wide range of users in a fully fledged and easy-to-use environment for data curation, exploration, and analysis. Value-added open data derived by our methods will also be made available under creative commons licenses where permitted.

8 Evaluation Plan

We define the following success metrics for this work: (i) query processing over AU-DBs has a small overhead compared to deterministic (selected-guess) query processing; (ii) the approximation accuracy of certain answers provided by AU-DBs is high for many real world use cases; (iii) attribute-level uncertainty encoded as bounds on attribute values increases the accuracy of approximations encoded by AU-DBs compared to UA-DBs at a low additional performance penalty; (iv) at comparable space usage, sketch-annotated databases achieve higher accuracy than sampling-based approaches when computing common statistical moments; (v) the concise representations of possible answers we use to deal with non-monotone queries such as negation and aggregation allows the efficient and correct (the output is an approximations of certain answers) evaluation of such queries over AU-DBs.

We will evaluate the performance and accuracy of our implementation of U4U over a wide range of real world datasets and synthetic benchmarks from the literature (e.g., PDBench [5]). Furthermore, we will integrate U4U as the new uncertainty tracking backend for our Vizier [22, 52] data-centric notebook platform. Vizier features automated error detection and repair methods called Lenses [105] which already expose the uncertainty in input data (by detecting errors) or output of a curation or cleaning step (where it is unclear which repair is correct) as UA-DBs. We will extend Lenses for AU-DBs and sketch annotations. This is similar in spirit to [18], but we use lighter-weight annotations instead of a full probabilistic database. Integration with Vizier, permits us to evaluate our techniques over a plethora of publicly available (messy) open government datasets and real-world data curation and analysis workflows that were created using Vizier. We will experimentally compare the performance and approximation accuracy of each uncertainty-annotation strategy against (i) other methods for approximating certain answers, (ii) exact methods for computing certain answers, and (iii) selected-guess query processing. Furthermore, we will compare UA-DBs with only tuple-level uncertainty with the novel attribute-bound AU-DBs encoding we propose in this work. Finally, we will compare certain answer techniques for non-monotone queries with our approach for maintaining a compact representation of possible answers to deal with non-monotonicity. In addition, the integration with the user-friendly Vizier environment enables us to evaluate the usability of user-facing representations of AU-DBs through interactions with domain experts.

9 Project Management and Timeline

The PIs will coordinate work on the project through joint bi-weekly online meetings with the students from both sites. A yearly in-person meeting will be held alternating between University at Buffalo and IIT (travel funds are requested as part of the budget). These day-long meetings will include progress reports, presentations by students, and a strategic planning session by the PIs. In addition to the funded Ph.D. students and PostDocs, the PIs will also involve students at all levels in projects related to the proposed research. The yearly meetings will also serve as a platform for these students to meet in person, present their research results, and to identify new opportunities for deeper collaboration. The PIs have successfully collaborated in the past, and this ongoing collaboration led to the development of some of the preliminary work for this proposal. Further project management details can be found in the collaboration plan. The time table below shows the 4 year plan for the proposed research.

Year	Trust I - AU-DBs	Thrust II - Sketches	Thrust III - U4U System
1	**•I-a** - extend UA-DB model for attribute-level uncertainty **•I-b** - extend UA-DBs to over-approximate possible answers	**•II-a** - investigate sketches for $\mathbb{N}$-databases **•II-b** - investigate efficiently invertible hash-functions	**•III-a** - AU-DB import modules for incomplete and probabilistic data **•III-b** - develop AU-DBs query rewrites for sets and bags
2	**•I-a** - develop metric for approximation tightness **•I-b** - study aggregation queries over AU-DBs **•I-c** - investigate applications beyond sets and bags	**•II-a** - study $\mathbb{N}$ sketch approximation guarantees **•II-b** - study sub-linear sketching algorithms for TIP-DBs	**•III-a** - import modules for basic cleaning and error detection methods **•III-b** - extend UA-DB rewrites for aggregation **•III-c** - develop specialized join operators
3	**•I-a** - investigate AU-DB approximation guarantees **•I-b** - study queries with set difference over AU-DBs **•I-c** - study UA-DB approximation guarantees for other semirings	**•II-a** - investigate probability estimation based on sketches **•II-b** - study sketch generation beyond TIP-DBs **•II-c** - study sketches for $\mathbb{B}$ (sets)	**•III-a** - extend import to support additional cleaning methods **•III-b** - extend rewrites for set difference **•III-b** - develop query rewrites for sketches **•III-c** - adapt TETRIS [60] to over-approximate possible answers
4	**•I-a/b** - study efficiency vs. accuracy trade-off **•I-c** - extend attribute-level bounds to semi-module expressions	**•II-a/b** - study interplay between sketch initialization and querying **•II-c** - study sketches for other $\mathcal{K}$ annotations	**•III-b** - study rewrites for compact representations of attribute bounds **•III-c** - online refinement for probabilistic/incomplete DBs

10 Prior NSF Results

PI Glavic. Dr. Glavic is funded through NSF grant ACI-1640864 as a subrecipient. Project Title: CIF21 DIBBs: EI: Vizier, Streamlined Data curation; Award Amount: $2.73 million; and Period of Performance: 10/03/2016 - 12/31/2019. **Intellectual Merit**: The goal of this work is to built Vizier, a next generation data curation system that will make it easier and faster to explore and analyze raw data. The work significantly improves the way how data science is conducted, making data curation easier, simpler, more transparent, more reliable, more exploratory, and more efficient. This work has lead to several publications in conferences, journals, and workshops [8, 9, 22, 39, 70, 71, 80, 89, 98]. **Broader Impact**: The Vizier system will be deployed by stakeholders from industry, government, and academia who will ensure sustainability. Thus, the research will have a long term transformative effect on how data science is conducted by a wide range of users. This grant is funding two Ph.D. students and one undergraduate student through a REU. Dr. Glavic is supervising 8 Ph.D. students (one minority female) and 1 undergraduate student.

PI Kennedy. PI Kennedy is funded through NSF grant ACI-1640864. Project Title: CIF21 DIBBs: EI: Vizier, Streamlined Data curation; Award Amount: $2.73 million; and Period of Performance: 10/03/2016 - 12/31/2019. **Intellectual Merit**: PI Kennedy's work for this award has explored techniques for managing messy and incomplete data [22, 38, 79, 97, 99, 104], inter-modal provenance [44, 87], and interfaces for the above [64–66], through a practical system called Vizier. **Broader Impact**: The Vizier system [52] facilitates collaborative reproducible research between participants of varying skill levels. This grant is funding one Ph.D. student, one developer, and has funded one undergraduate student through an REU. Dr. Kennedy is presently supervising 5 Ph.D. students (2 female).

PI Rudra. Rudra was funded through NSF grant CCF-1319402. Project Title: AF:III:Small:Collaborative Research: New Frontiers in Join Algorithms: Optimality, Noise, and Richer Languages; Amount: $326,101; and Period of Performance: 09/01/2013 - 08/31/2017. The main results of this grant in **Intellectual Merit** are as follows: (i) the first beyond-worst-case results for the general join query in [83], (ii) a resolution-based framework for designing join algorithms that recovers most of known (worst-case and beyond worst-case) results as well as new results on computing joins [60]; (iii) extending the join results to a more general framework [61], which won the **PODS 2016 best paper award**. With respect to **Broader Impacts**, this grant partially supported two Ph.D. students at Buffalo, Mahmoud Abo Khamis and Jimmy Dobler. The grant also supported a general database audience survey on the developments in worst-case optimal join algorithms [85]. PI Rudra is currently funded through NSF CCF-1763481 and CCF-1717134 but they are not as closely related to this proposal as the above award.

10 References

[1] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. Detecting data errors: Where are we and what needs to be done? *PVLDB*, 9(12):993–1004, 2016.

[2] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.*, 78(1):158–187, 1991.

[3] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. In *SIGMOD Conference*, pages 34–48, 1987.

[4] Foto N. Afrati and Phokion G. Kolaitis. Answering aggregate queries in data exchange. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 129–138, 2008.

[5] Lyublena Antova, Thomas Jansen, Christoph Koch, and Dan Olteanu. Fast and simple relational processing of uncertain data. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*, pages 983–992, 2008.

[6] Lyublena Antova, Christoph Koch, and Dan Olteanu. Maybms: Managing incomplete information with probabilistic world-set decompositions. In *ICDE*, pages 1479–1480, 2007.

[7] Lyublena Antova, Christoph Koch, and Dan Olteanu. $10^{(10^6)}$ worlds and beyond: Efficient representation and processing of incomplete information. *VLDB*, 18(5):1021–1040, 2009.

[8] Bahareh Sadat Arab, Su Feng, **Boris Glavic**, Seokki Lee, Xing Niu, and Qitian Zeng. Gprom - A swiss army knife for your provenance needs. *IEEE Data Eng. Bull.*, 41(1):51–62, 2018.

[9] Bahareh Sadat Arab, Dieter Gawlick, Vasudha Krishnaswamy, Venkatesh Radhakrishnan, and **Boris Glavic**. Using reenactment to retroactively capture provenance for transactions. *IEEE Trans. Knowl. Data Eng.*, 30(3):599–612, 2018.

[10] Marcelo Arenas, Leopoldo E. Bertossi, Jan Chomicki, Xin He, Vijay Raghavan, and Jeremy P. Spinrad. Scalar aggregation in inconsistent databases. *Theor. Comput. Sci.*, 296(3):405–434, 2003.

[11] Lars Arge, Octavian Procopiuc, Sridhar Ramaswamy, Torsten Suel, and Jeffrey Scott Vitter. Scalable sweeping-based spatial join. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 570–581, 1998.

[12] Patricia C. Arocena, **Boris Glavic**, Radu Ciucanu, and Renée J. Miller. The ibench integration metadata generator. *PVLDB*, 9(3):108–119, 2015.

[13] Patricia C. Arocena, **Boris Glavic**, Giansalvatore Mecca, Renée J. Miller, Paolo Papotti, and Donatello Santoro. Messing up with BART: error generation for evaluating data-cleaning algorithms. *PVLDB*, 9(2):36–47, 2015.

[14] Subi Arumugam, Fei Xu, Ravi Jampani, Christopher Jermaine, Luis L. Perez, and Peter J. Haas. Mcdb-r: Risk analysis in the database. *PVLDB*, 3(1-2):782–793, 2010.

[15] James Aspnes, Eric Blais, Murat Demirbas, Ryan O'Donnell, **Atri Rudra**, and Steve Uurtamo. k $^+$ decision trees - (extended abstract). In *Algorithms for Sensor Systems - 6th International Workshop on Algorithms for Sensor Systems, Wireless Ad Hoc Networks, and Autonomous Mobile Entities, ALGOSENSORS 2010, Bordeaux, France, July 5, 2010, Revised Selected Papers*, pages 74–88, 2010.

[16] Leopoldo E. Bertossi. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[17] George Beskales. *Modeling and Querying Uncertainty in Data Cleaning*. PhD thesis, University of Waterloo, Ontario, Canada, 2012.

[18] George Beskales, Ihab F. Ilyas, Lukasz Golab, and Artur Galiullin. Sampling from repairs of conditional functional dependency violations. *VLDBJ*, 23(1):103–128, 2014.

[19] George Beskales, Mohamed A. Soliman, Ihab F. Ilyas, Shai Ben-David, and Yubin Kim. Probclean: A probabilistic duplicate detection system. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, pages 1193–1196, 2010.

[20] Patrick Bosc, BillB. Buckles, FrederickE. Petry, and Olivier Pivert. Fuzzy databases. In JamesC. Bezdek, Didier Dubois, and Henri Prade, editors, *Fuzzy Sets in Approximate Reasoning and Information Systems*, volume 5 of *The Handbooks of Fuzzy Sets Series*, pages 403–468. 1999.

[21] Jihad Boulos, Nilesh Dalvi, Bhushan Mandhani, Shobhit Mathur, Chris Re, and Dan Suciu. Mystiq: A system for finding more answers by using probabilities. In *SIGMOD*, pages 891–893, 2005.

[22] Mike Brachmann, Carlos Bautista, Sonia Castelo, Su Feng, Juliana Freire, **Boris Glavic**, **Oliver Kennedy**, Heiko Müller, Rémi Rampin, William Spoth, and Ying Yang. Data debugging and exploration with vizier. In *Proceedings of the 44th International Conference on Management of Data (Demonstration Track)*, pages 1877–1880, 2019.

[23] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient processing of spatial joins using r-trees. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993.*, pages 237–246, 1993.

[24] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Parallel processing of spatial joins using r-trees. In *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana, USA*, pages 258–265, 1996.

[25] Marco Calautti, Leonid Libkin, and Andreas Pieris. An operational approach to consistent query answering. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 239–251, 2018.

[26] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *ICALP*, volume 2380 of *Lecture Notes in Computer Science*, pages 693–703, 2002.

[27] E. F. Codd. Extending the database relational model to capture more meaning. *TODS*, 4(4):397–434, 1979.

[28] Graham Cormode. *Count-Min Sketch*, pages 511–516. 2009.

[29] Graham Cormode and Minos N. Garofalakis. Sketching probabilistic data streams. In *SIGMOD Conference*, pages 281–292, 2007.

[30] Graham Cormode, Minos N. Garofalakis, Peter J. Haas, and Chris Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1-3):1–294, 2012.

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

[32] Guy Van den Broeck and Dan Suciu. Query processing on probabilistic data: A survey. *Foundations and Trends in Databases*, 7(3-4):197–341, 2017.

[33] Anton Dignös, **Boris Glavic**, Xing Niu, Johann Gamper, and Michael H. Böhlen. Snapshot semantics for temporal multiset relations. *PVLDB*, 12(6):639–652, 2019.

[34] Akhil A. Dixit and Phokion G. Kolaitis. A sat-based system for consistent query answering. *CoRR*, abs/1905.02828, 2019.

[35] Jimmy Dobler and **Atri Rudra**. Implementation of tetris as a model counter. *CoRR*, abs/1701.07473, 2017.

[36] M. F. Duarte and Y. C. Eldar. Structured compressed sensing: From theory to applications. *Trans. Sig. Proc.*, 59(9):4053–4085, September 2011.

[37] Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. Probabilistic data exchange. *J. ACM*, 58(4):15:1–15:55, 2011.

[38] Su Feng, Aaron Huber, **Boris Glavic**, and **Oliver Kennedy**. Uncertainty annotated databases - a lightweight approach for dealing with uncertainty. Technical Report IIT/CS-DB-2018-01, Illinois Institute of Technology, 2018.

[39] Su Feng, Aaron Huber, **Boris Glavic**, and **Oliver Kennedy**. Uncertainty annotated databases - A lightweight approach for approximating certain answers. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.*, pages 1313–1330, 2019.

[40] Robert Fink, Larisa Han, and Dan Olteanu. Aggregation in probabilistic databases via knowledge compilation. *PVLDB*, 5(5):490–501, 2012.

[41] Robert Fink, Jiewen Huang, and Dan Olteanu. Anytime approximation in probabilistic databases. *VLDBJ*, 22(6):823–848, 2013.

[42] Juliana Freire, **Boris Glavic**, **Oliver Kennedy**, and Heiko Mueller. The exception that improves the rule. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics, HILDA@SIGMOD 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, page 7, 2016.

[43] Juliana Freire, **Boris Glavic**, **Oliver Kennedy**, and Heiko Mueller. The exception that improves the rule. *CoRR*, abs/1606.00046, 2016.
[44] Juliana Freire, **Boris Glavic**, **Oliver Kennedy**, and Heiko Mueller. The exception that improves the rule. In *HILDA*, 2016.
[45] Dengfeng Gao, Christian S. Jensen, Richard T. Snodgrass, and Michael D. Soo. Join operations in temporal databases. *VLDB J.*, 14(1):2–29, 2005.
[46] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. The LLUNATIC data-cleaning framework. *PVLDB*, 6(9):625–636, 2013.
[47] Floris Geerts, Fabian Pijcke, and Jef Wijsen. First-order under-approximations of consistent query answers. *Int. J. Approx. Reasoning*, 83:337–355, 2017.
[48] Anna C. Gilbert and Piotr Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.
[49] **Boris Glavic** and Gustavo Alonso. Perm: Processing provenance and data on the same data model through query rewriting. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pages 174–185, 2009.
[50] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.
[51] ToddJ. Green and Val Tannen. Models for incomplete and probabilistic information. In Torsten Grust, Hagen Höpfner, Arantza Illarramendi, Stefan Jablonski, Marco Mesiti, Sascha Müller, Paula-Lavinia Patranjan, Kai-Uwe Sattler, Myra Spiliopoulou, and Jef Wijsen, editors, *EDBT*, volume 4254 of *Lecture Notes in Computer Science*, pages 278–296. 2006.
[52] The VizierDB Group. Vizier: Built for data exploration. http://vizierdb.info.
[53] Paolo Guagliardo and Leonid Libkin. Making sql queries correct on incomplete databases: A feasibility study. In *PODS*, pages 211–223, 2016.
[54] Paolo Guagliardo and Leonid Libkin. Correctness of SQL queries on databases with nulls. *SIGMOD Record*, 46(3):5–16, 2017.
[55] Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. Holodetect: Few-shot learning for error detection. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.*, pages 829–846, 2019.
[56] Tomasz Imielinski, Ron van der Meyden, and Kumar V. Vadaparty. Complexity tailored design: A new design methodology for databases with incomplete information. *J. Comput. Syst. Sci.*, 51(3):405–432, 1995.
[57] Tomasz Imieliński and Witold Lipski, Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.
[58] Ravi Jampani, Fei Xu, Mingxi Wu, Luis Leopoldo Perez, Christopher Jermaine, and Peter J. Haas. Mcdb: A monte carlo approach to managing uncertain data. In *SIGMOD*, pages 687–700, 2008.
[59] **Oliver Kennedy** and Christoph Koch. Pip: A database system for great and small expectations. In *ICDE*, pages 157–168, 2010.
[60] Mahmoud Abo Khamis, Hung Q. Ngo, Christopher Ré, and **Atri Rudra**. Joins via geometric resolutions: Worst case and beyond. *ACM Trans. Database Syst.*, 41(4):22:1–22:45, 2016.
[61] Mahmoud Abo Khamis, Hung Q. Ngo, and **Atri Rudra**. FAQ: questions asked frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 13–28, 2016.
[62] Mahmoud Abo Khamis, Hung Q. Ngo, and **Atri Rudra**. Juggling functions inside a database. *SIGMOD Record*, 46(1):6–13, 2017.
[63] Egor V. Kostylev and Peter Buneman. Combining dependent annotations for relational algebra. In *ICDT*, pages 196–207, 2012.
[64] Poonam Kumari. Make informed decisions: Understanding query results from incomplete databases. In *VLDB-PhD Workshop*, 2019.
[65] Poonam Kumari, Said Achmiz, and **Oliver Kennedy**. Communicating data quality in on-demand curation. In *QDB*, 2016.

[66] Poonam Kumari and **Oliver Kennedy**. The good and bad data. In *NEDB*, 2018.
[67] Alan G. B. Lauder and Daqing Wan. Counting points on varieties over finite fields of small characteristic. *arXiv Mathematics e-prints*, page math/0612147, Dec 2006.
[68] Jens Lechtenbörger, Hua Shu, and Gottfried Vossen. Aggregate queries over conditional tables. *J. Intell. Inf. Syst.*, 19(3):343–362, 2002.
[69] Seokki Lee, Sven Köhler, Bertram Ludäscher, and **Boris Glavic**. A sql-middleware unifying why and why-not provenance for first-order queries. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 485–496, 2017.
[70] Seokki Lee, Bertram Ludäscher, and **Boris Glavic**. Provenance summaries for answers and non-answers. *PVLDB*, 11(12):1954–1957, 2018.
[71] Seokki Lee, Bertram Ludäscher, and **Boris Glavic**. Pug: a framework and practical implementation for why and why-not provenance. *The VLDB Journal*, August 2018.
[72] Leonid Libkin. SQL's three-valued logic and certain answers. *TODS*, 41(1):1:1–1:28, 2016.
[73] Leonid Libkin. Sql's three-valued logic and certain answers. *TODS*, 41(1):1:1–1:28, 2016.
[74] Inés Fernando Vega López, Richard T. Snodgrass, and Bongki Moon. Spatiotemporal aggregate computation: a survey. *IEEE Trans. Knowl. Data Eng.*, 17(2):271–286, 2005.
[75] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.
[76] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. Raha: A configuration-free error detection system. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.*, pages 865–882, 2019.
[77] Nikos Mamoulis, Yannis Theodoridis, and Dimitris Papadias. Spatial joins: Algorithms, cost models and optimization techniques. In *Spatial Databases: Technologies, Techniques and Trends*, pages 155–184. 2005.
[78] Yannis Manolopoulos, Apostolos Papadopoulos, and Michael Vassilakopoulos, editors. *Spatial Databases: Technologies, Techniques and Trends*. Idea Group, 2005.
[79] Niccolò Meneghetti, **Oliver Kennedy**, and Wolfgang Gatterbauer. Beta probabilistic databases: A scalable approach to belief updating and parameter learning. In *SIGMOD*, 2017.
[80] Niccolò Meneghetti, **Oliver Kennedy**, and Wolfgang Gatterbauer. Learning from query-answers: A scalable approach to belief updating and parameter learning. *TODS*, 2018.
[81] Raghotham Murthy, Robert Ikeda, and Jennifer Widom. Making aggregation work in uncertain and probabilistic databases. *IEEE Trans. Knowl. Data Eng.*, 23(8):1261–1273, 2011.
[82] Arindam Nandi, Ying Yang, **Oliver Kennedy**, **Boris Glavic**, Ronny Fehling, Zhen Hua Liu, and Dieter Gawlick. Mimir: Bringing ctables into practice. *CoRR*, abs/1601.00073, 2016.
[83] Hung Q. Ngo, Dung T. Nguyen, Christopher Ré, and **Atri Rudra**. Beyond worst-case analysis for joins with minesweeper. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 234–245, 2014.
[84] Hung Q. Ngo, Ely Porat, Christopher Ré, and **Atri Rudra**. Worst-case optimal join algorithms: [extended abstract]. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 37–48, 2012.
[85] Hung Q. Ngo, Christopher Ré, and **Atri Rudra**. Skew strikes back: new developments in the theory of join algorithms. *SIGMOD Record*, 42(4):5–16, 2013.
[86] Hung Q. Ngo and **Atri Rudra**. Efficient decodable group testing. In *Encyclopedia of Algorithms*, pages 614–618. 2016.
[87] Xing Niu, Bahareh Arab, Dieter Gawlick, Zhen Hua Liu, Vasudha Krishnaswamy, **Oliver Kennedy**, and **Boris Glavic**. Provenance-aware versioned dataworkspaces. In *TaPP*, 2016.
[88] Xing Niu, Bahareh Sadat Arab, Dieter Gawlick, Zhen Hua Liu, Vasudha Krishnaswamy, **Oliver Kennedy**, and **Boris Glavic**. Provenance-aware versioned dataworkspaces. In *8th USENIX Workshop on the Theory and Practice of Provenance, TaPP 2016, Washington, D.C., USA, June 8-9, 2016.*, 2016.

[89] Xing Niu, Raghav Kapoor, **Boris Glavic**, Dieter Gawlick, Zhen Hua Liu, Vasudha Krishnaswamy, and Venkatesh Radhakrishnan. Heuristic and cost-based optimization for diverse provenance tasks. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[90] Xing Niu, Raghav Kapoor, **Boris Glavic**, Dieter Gawlick, Zhen Hua Liu, and Venkatesh Radhakrishnan. Provenance-aware query optimization. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 473–484, 2017.

[91] Danila Piatov, Sven Helmer, and Anton Dignös. An interval join optimized for modern hardware. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 1098–1109, 2016.

[92] Abdulhakim Ali Qahtan, Nan Tang, Mourad Ouzzani, Yang Cao, and Michael Stonebraker. ANMAT: automatic knowledge discovery and error detection through pattern functional dependencies. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.*, pages 1977–1980, 2019.

[93] Christopher Ré and Dan Suciu. The trichotomy of HAVING queries on a probabilistic database. *VLDB J.*, 18(5):1091–1116, 2009.

[94] Raymond Reiter. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *J. ACM*, 33(2):349–370, 1986.

[95] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.

[96] Sarvjeet Singh, Chris Mayfield, Sagar Mittal, Sunil Prabhakar, Susanne Hambrusch, and Rahul Shah. Orion 2.0: Native support for uncertain data. In *SIGMOD*, pages 1239–1242, 2008.

[97] William Spoth, Bahareh Sadat Arab, Eric S. Chan, Dieter Gawlick, Adel Ghoneimy, **Boris Glavic**, Beda Hammerschmidt, **Oliver Kennedy**, Seokki Lee, Zhen Hua Liu, Xing Niu, and Ying Yang. Adaptive schema databases. In *CIDR*, 2017.

[98] William Spoth, Bahareh Sadat Arab, Eric S. Chan, Dieter Gawlick, Adel Ghoneimy, **Boris Glavic**, Beda Christoph Hammerschmidt, **Oliver Kennedy**, Seokki Lee, Zhen Hua Liu, Xing Niu, and Ying Yang. Adaptive schema databases. In *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*, 2017.

[99] William Spoth, Ting Xie, **Oliver Kennedy**, Ying Yang, Beda Hammerschmidt, Zhen Hua Liu, and Dieter Gawlick. Schemadrill: Interactive semi-structured schema design. In *HILDA*, 2018.

[100] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[101] Bruhathi Sundarmurthy, Paraschos Koutris, Willis Lang, Jeffrey Naughton, and Val Tannen. m-tables: Representing missing data. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 68. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[102] Pei Wang and Yeye He. Uni-detect: A unified approach to automated error detection in tables. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.*, pages 811–828, 2019.

[103] Jennifer Widom. Trio: A system for integrated management of data, accuracy, and lineage. *Technical Report*, 2004.

[104] Ying Yang and **Oliver Kennedy**. Convergent inference with leaky joins. In *EDBT*, 2017.

[105] Ying Yang, Niccolò Meneghetti, Ronny Fehling, Zhen Hua Liu, and **Oliver Kennedy**. Lenses: An on-demand approach to ETL. *PVLDB*, 8(12):1578–1589, 2015.

[106] Maria Zemankova and Abraham Kandel. Implementing imprecision in information systems. *Information Sciences*, 37(1):107–141, 1985.