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 types of uncertainty include missing values, sensor errors, 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 such as unfounded scientific discoveries, financial damages, or even effects on people's physical well-being (e.g., medical decisions based on incorrect data).

Classical data management systems are designed around the assumption that data quality issues and uncertainties have been resolved 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. Techniques for managing incomplete data (e.g., [15,22,31,36,68]) exist, but are generally too heavy-weight for real-world usage, and also may hide relevant information from users. We propose to explore a recently introduced abstraction called **Uncertainty-Annotated Databases** (**UA-DBs**) that bridges the gap between classical and incomplete data management. Rather than trying to outright replace classical data management semantics, UA-DBs annotate existing data with *uncertainty labels*, and provide lightweight semantics for propagating these labels through queries. The result is a strict generalization of classical data management that also clearly distinguishes between reliable (i.e., certain) and potentially unreliable (i.e., uncertain) data and results.

`customers`

SSN	name	income	ownsProperty
777-777-7777	Alice	$60,000	**NULL**(no)
333-333-3333	Bob	$102,000	no
111-111-1111	Peter	**NULL**($0)	yes
555-555-5555	Arno	$95,000	yes

Figure 1: Example of uncertainty in data. Loan decisions based on this data, even after imputing missing values (shown in red), may lead to loans erroneously being denied.

Example 1 *Consider the* `customers` *table shown in Figure 1, which tracks customer features that a bank uses to determine a customer's eligibility for loans. This data is missing values, including Peter's income and whether Alice owns property, and is thus uncertain. Ideally, the bank would curate the data manually (e.g., a service representative calls Alice to request the missing information) before acting on it. However if the bank is sufficiently large, it may be more productive to mine existing data (e.g., property tax records) to "fill in" missing values. Such mining often also involves uncertainty, for example as a result of entity matching. For this example, assume that the best match for Alice in her town's property tax database has a 40% likelihood. Because no match exists with greater than 50% likelihood, the* **NULL** *value would typically be replaced replaced with a "no". From the bank's perspective, the data quality issue is resolved. However, there is now at least a 40% chance that any loan decision the bank makes will be based on faulty evidence.*

This problem arises because classical data management systems are not equipped to track uncertainty. Consider an alternative where the mined `ownsProperty` value was labeled as uncertain, due to the not-insignificant possibility of a match. By propagating this label through the query that determines Alice's eligibility, a loan officer can see that the outcome (i.e., denied) is not certain and investigate further.

The challenge of tracking uncertainty has been explored through *incomplete* [3, 36], *probabilistic* [20, 67], and *fuzzy* [71] databases. An incomplete database encodes a set of database instances called *possible worlds*, while a probabilistic (resp., fuzzy) database also encodes a probability (resp., score) for each world. Unfortunately, probabilistic and fuzzy databases are slow (e.g., [53] finds probabilistic databases to be up to an order of magnitude slower than classical databases), and can be unintuitive. Incomplete databases track more intuitive, qualitative descriptors like "certain" and "possible." However the primary query semantics for incomplete databases: computing *certain answers* (a) is also slow (coNP-hard [3, 36] in general), and (b) discards many results that are within the realm of possibility and might thus still be useful to the user.

Hence, most analysts simply make a best-effort attempt to remove potential sources of error in the data, and then work with the data as if it were perfect. Unfortunately, in this sort of *selected-guess analytics*, information about ambiguous interpretations of the source data and irreparable data errors is lost, or at best only retained out-of-band (e.g., in a README file or comments field).

An abstraction recently proposed by the PIs [23] called **Uncertainty-Annotated Databases** (**UA-DBs**) bridges the gap between certain query answering and selected-guess analytics by fusing the latter with a lightweight approximation of the former. Specifically, queries in a UA-DB behave exactly as in selected-guess analytics, but results also include sufficient information to distinguish result tuples and values that are certain from those that might not be. As illustrated in Figure 2 (see [23] for more details), UA-DBs only introduce a small performance penalty compared to selected-guess analytics, never falsely flag an answer as certain, and (depending on dataset quality and query specifics) only rarely incorrectly flag answers as being uncertain. UA-DBs (and our preliminary implementation of them) extend the state-of-the-art in the following ways:

	$Q1$	$Q2$	$Q3$	$Q4$	$Q5$
Runtime Overhead	2.28%	1.81%	1.32%	2.88%	3.51%
Misclassified Answers	0.55%	0.37%	0%	0.92%	0.29%

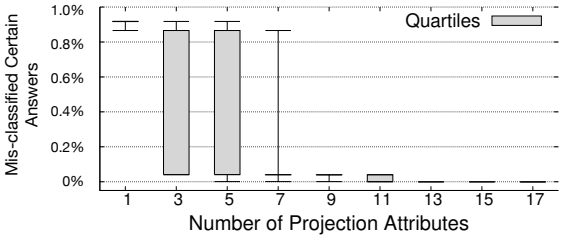

Figure 2: Experimental results using crime data [1]: (top) 5 real-world queries; (bottom) a large number of randomly generated projection queries. UA-DBs have low overhead, never incorrectly flag an answer as certain, and rarely mis-classify certain results as uncertain.

Generality. Most past approaches for dealing with uncertainty have been limited to set semantics. Our framework also handles bags (and, thus, SQL databases). Furthermore, we support a wide range of extensions of the relational model such as access control annotations, various types of provenance, and many more. In fact, any extension of the relational model that can be expressed through the (quite general) $\mathcal{K}$-relations framework [29] is supported by UA-DBs.

Efficiently Bounding Certain Answers. Previous work has focused exclusively on under-approximating certain answers. The rationale is that under-reporting certain answers is safer than mis-classifying an uncertain answer as certain. By constrast, our approach combines over- and under-approximations: A superset of the certain answers is annotated such that the subset labeled as certain is an under-approximation.

Backward Compatibility. A UA-DB can be efficiently constructed from any existing model of incomplete, probabilistic, or fuzzy data, as long as we can compute (1) a distinguished possible world (the over-approximation) and (2) a labeling of the tuples from this world as (un-)certain (the under-approximation). In preliminary work [23], we have already demonstrated compatibility with several common incomplete database models such as V-tables [3], c-tables [36], and tuple- and block-independent probabilistic databases [67].

Combing Selected-Guess Query Processing with Certain Answers. UA-DBs combine the benefits of selected-guess analytics (a practically proven solution that is efficient and works well in a highly uncertain world) with an approximation of certain answers (a proven formal approach with strong guarantees of correctness). Furthermore, unlike certain answers, UA-DBs are closed under queries.

In the following, we discuss background and related work, including preliminary work on UA-DBs. Next, we present the primary goals of the proposed work and our evaluation plan, and finally discuss the broader impacts and intellectual merit of the proposal. The proposed work follows two major research thrusts:

Thrust I: Formal foundations of UA-DBs. Building on the foundation of our preliminary work [23], we will extend the UA-DB model and study its properties. *(a) Approximation Guarantees*: We will investigate how properties of queries and data affect the accuracy of the approximation of certain answers provided by UA-DBs. *(b) Certainty in Non-Set Databases*: UA-DBs, in principle, support any data model expressible in Green et al.'s $\mathcal{K}$-relations framework [29] (e.g., provenance annotations or natural numbers encoding bag semantics). We will develop the theory of UA-DBs to take advantage of this generality and investigate how

the choice of annotation domain affects the accuracy of our approximations of certain answers. *(c) Attribute-level Annotations*: We will extend UA-DBs with attribute-level annotations that encode bounds on the values of an attribute across all possible worlds, enabling a more precise encoding of the uncertainty inherent in a dataset. We conjecture that this extension will not result in a significant increase in computational complexity. *(d) Non-monotone Queries*: Under-approximating certain answers is easy only for monotone queries. As observed elsewhere [33, 51], non-monotone queries can be dealt with if an *over-approximation* of *possible* answers is available. However, such over-approximations can be large. We will combine this idea with recent results from instance optimal join query processing [21, 42] to support efficient non-monotone queries in UA-DBs.

Thrust II: Efficient algorithms for UA-DBs and their implementation in U4U. We propose to build a system called *U4U* (*Uncertainty For You*) that implements algorithms for transforming existing incomplete and probabilistic data models into UA-DBs and for efficiently evaluating queries over UA-DBs. *(a) Management of Uncertainty-Labeled Data*: We will develop adapters expressed as relational queries that translate data from incomplete and probabilistic data models into UA-DBs. Furthermore, we propose to extend SQL's DML and DDL constructs to encode operations (e.g., key-repair [6]) that introduce and manage uncertainty. Importantly, this will include the study of how update operations can be executed over UA-DBs. *(b) Query Rewriting for UA-DBs*: Building on promising results with a prototype query-rewriting middleware implementing tuple-level UA-DBs, we will implement and evaluate support for attribute-level annotations, aggregation (and non-monotone queries), and additional types of annotations (e.g., uncertain provenance annotations). *(c) A Specialized Database Engine for UA-DBs*: To further improve performance of query evaluation, we will design and implement a specialized UA-DB query processing engine that will exploit compact representations based on factorization, functional aggregate queries [43] (FAQ), and geometrical interpretations of relations [42]. *(d) Online Refinement of UA-DBs*: Since UA-DBs encode both super- and subsets of certain answers, they can be used as a pruning step to speed up exact algorithms for computing certain answers (and/or probabilities).

The Team. The PIs are uniquely qualified to conduct the proposed research. PI Glavic has extensive background in annotated databases and provenance [9, 28, 48, 60] (including a best-of-ICDE invitation to TKDE [59]). PI Kennedy has a background in probabilistic databases and uncertainty in general [41, 46, 52, 53, 70] (including a best-of-SIGMOD invitation to TODS in 2017 [52]). PIs Glavic and Kennedy are actively collaborating on topics relevant to this proposal [23, 25, 26, 53, 58, 65] and have long standing experience in developing systems in the domain of data cleaning [12,13], curation, uncertainty [23,41,70], and provenance [9,28]. PI Rudra has background in database theory and has studied formal aspects of annotated databases involving semirings [43,44] (a line of research that won the PODS 2016 [43] and PODS 2012 [55,56] best paper awards) and compact representations of non-results in joins and #-SAT formulas [21, 42].

2 Background and Related Work

Before presenting an overview of our UA-DB framework, we first introduce necessary background and related work on incomplete databases, probabilistic databases, and $\mathcal{K}$-relations. A database schema $\mathbf{D} = \{\mathbf{R}_1, \ldots, \mathbf{R}_n\}$ is a set of relation schemas. A relational schema $\mathbf{R}(A_1, \ldots, A_n)$ consists of a relation name and a set of attribute names $A_1, \ldots, A_n$. The arity $arity(\mathbf{R})$ of a relation schema $\mathbf{R}$ is the number of attributes in $\mathbf{R}$. A database instance D for database schema $\mathbf{D}$ is a set of relation instances with one relation for each relation schema in $\mathbf{D}$: $D = \{R_1, \ldots, R_n\}$. Assume a universal domain of attribute values $\mathbb{D}$. A tuple with schema $\mathbf{R}$ is an element from $\mathbb{D}^{arity(\mathbf{R})}$. A set relation R with schema $\mathbf{R}$ is a set of tuples with schema $\mathbf{R}$, i.e., $R \subseteq \mathbb{D}^{arity(\mathbf{R})}$. A bag relation R with schema $\mathbf{R}$ is a bag (multiset) of tuples with schema $\mathbf{R}$.

2.1 Possible Worlds Semantics

Incomplete databases 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

id	address	ℓ	$\mathbb{N}$	$\mathbb{B}$
1	51 Co...	L_1	1	T
2	Grant ...	L_2	1	T
3	499 W...	L_4	1	T

(a) Address

ℓ	locale	state	$\mathbb{N}$	$\mathbb{B}$
L_1	L...	NY	1	T
L_2	T...	AZ	1	T
L_3	G...	NY	1	T
L_4	K...	NY	1	T
L_5	W...	IL	1	T

(b) Neighborhood

state	$\mathbb{N}$	$\mathbb{B}$
NY	$2 = (1 \cdot 1) + (1 \cdot 1)$	$T = (T \wedge T) \vee (T \wedge T)$
AZ	$1 = (1 \cdot 1)$	$T = (T \wedge T)$
IL	$0 = (0 \cdot 1)$	$F = (F \wedge T)$

(c) Result of Q_a

Figure 3: $\mathbb{N}$- and $\mathbb{B}$-relation examples

standard deterministic query semantics: $Q(\mathcal{D}) = \{\ Q(D) \mid D \in \mathcal{D}\ \}$. Decades of research [6, 16, 31, 36, 63, 67] has focused on algorithms for efficient query processing over incomplete databases, introducing data models such as V-tables [3], c-tables [36], and tuple-independent probabilistic databases [67] that more compactly represent a set of possible worlds by factoring out commonalities.

2.2 Certain, Possible, and Best-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 [18, 36]:

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

In contrast to [36], which studies certain answers to queries, we find it useful to define certainty at the instance level. The two approaches are equivalent since we can compute the certain answers of a query Q over incomplete instance $\mathcal{D}$ as $certain(Q(\mathcal{D}))$. For ease of presentation we will abuse terminology and 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 [33, 51, 61] 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.

Selected-Guess Query Processing. As mentioned in the introduction, another approach commonly used in practice is to ignore ambiguity in the data and select one distinguished possible world as canon. Queries are evaluated solely in this world, and ambiguity is ignored or documented outside of the database. We refer to this approach as *selected-guess query processing* (SGQP) [70] since typically one would like to select the possible world that is deemed most likely (e.g., the world with the highest probability). Going forward, we will assume that it is possible to select one distinguished possible world that we will call the *selected-guess world* $D_{\mathcal{G}}$. For example, a probabilistic database [67] pairs the set of possible worlds with a probability measure $P : \mathcal{D} \to [0, 1]$. Given a probabilistic database, one may define $D_{\mathcal{G}} = \text{argmax}_{D \in \mathcal{D}} P(D)$. We also note that although we refer to it as a "best-guess" world, any distinguished world — even an arbitrary one — will still be able to provide some utility, as any possible world can serve as an over-approximation of certain answers.

2.3 $\mathcal{K}$-relations

The UA-DB model that we propose to study is based on the **$\mathcal{K}$-relation/$\mathcal{K}$-database** [29] framework that defines relations annotated with elements from the domain K of a commutative semiring $\mathcal{K}$. A commutative semiring is an algebraic structure based on two binary operators $(+_{\mathcal{K}}, \cdot_{\mathcal{K}})$ with identities $(0_{\mathcal{K}}, 1_{\mathcal{K}})$ that satisfy specific algebraic laws like commutativity, associativity, and distributivity. 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. The $\mathcal{K}$-relation framework defines a non-standard semantics for positive relational algebra that computes the annotation of a query result by combing the annotations of input tuples using the addition $(+_{\mathcal{K}})$ and multiplication $(\cdot_{\mathcal{K}})$ operations of the semiring. When using the boolean semiring, i.e., the semiring over the boolean constants $\mathbb{B} = \{T, F\}$ with disjunction $(\vee)$ as addition and conjunction $(\wedge)$ as multiplication, the resulting $\mathbb{B}$-relations exactly mirror set-semantics (a tuple is annotated with T if and

only if it appears in a relation). Similarly, the semiring $(\mathbb{N}, +, \times, 0, 1)$ of natural numbers $\mathbb{N}$ with standard addition and multiplication over natural numbers mirrors bag semantics (each tuple is annotated with its multiplicity). Other semirings can be used to model, for example, security policies and provenance.

Example 2 *Figure 3 shows an example of how to encode a bag semantics database as an $\mathbb{N}$-database by annotating each tuple t with its multiplicity (the number of copies of t that exist in the relation). Annotations are shown to the right of each tuple. Query Q_a, shown below, returns states.*

$$Q_a = \pi_{state}\left(\mathsf{Address} \bowtie \mathsf{Neighborhood}\right)$$

In the input database every tuple appears once (is annotated with 1). The annotation of an output tuple is computed by multiplying annotations of joined tuples, and summing up annotations projected onto the same result tuple. For instance, 2 NY addresses are returned.

3 Uncertainty-Annotated Databases

Before discussing our proposed contributions, we first introduce the core concepts behind them. We begin with incomplete $\mathcal{K}$-relations, an extension of incomplete databases to $\mathcal{K}$-relations. Recall that $\mathcal{K}$-relations generalize set semantics, bag semantics, provenance, and many other extension of the relational model. The same holds for incomplete $\mathcal{K}$-relations. We introduce certain annotations as a sensible generalization of certain answers to incomplete $\mathcal{K}$-relations. Exploiting the natural order, an order relation over the domain of a semiring that is defined for many semirings, we define the certain annotation of a tuple as a lower bound of its annotation across all possible worlds. Importantly, this generalization coincides with the standard definition of certain answers for set semantics and the definition for bag semantics proposed by Guagliardo et al. [34]. We then introduce Uncertainty-Annotated Databases (UA-DBs), which are databases where each tuple is annotated with an under-approximation as well as an over-approximation of its certain annotation. By choosing the annotation of a tuple in the selected-guess world as the over-approximation, UA-DBs are a strict improvement over selected-guess query processing. Figure 4 shows the relationship of certain answers, selected-guess answers, and UA-DBs for set semantics. Recall that set semantics corresponds to $\mathbb{B}$-relations: tuples are either annotated with T (true) to indicate that they exist or F (false) to indicate that they do not exist. A set semantics UA-DB annotates each tuple with a pair $[b_1, b_2]$ where b_2 encodes the over-approximation of certain answers (it is T if the tuple exists in the selected-guess world) and b_1 encodes an under-approximation of certain answer (if it is T then the tuple is surely a certain answer). Hence, a UA-DB sandwiches the certain answers from below and above. Queries over UA-DBs treat the two annotations of a tuple independently and, thus, preserve the encoded selected-guess world (over-approximation). An important result we have proven [23] is that queries over UA-DBs also preserve the under-approximation of certain answers, which is a generalization of an earlier result for set semantics from Reiter [61].

Figure 4: A set UA-DB stores best-guess tuples and labels some subset of those as certain. The certain answers of a query are sandwiched between the results a UA-DB labels as certain (under-approximation) and the best-guess answers (over-approximation). The set of `POSSIBLE` tuples [5] is the union of all possible worlds, and `IMPOSSIBLE` tuples are tuples from the active domain ($\mathbb{D}^{arity(\mathcal{R})}$) not in any possible world.

3.1 Incomplete $\mathcal{K}$-Relations

An *incomplete $\mathcal{K}$-database* $\mathcal{D}$ is a set of possible worlds $\mathcal{D} = \{D_1, \ldots, D_n\}$ where each possible world D_i is a $\mathcal{K}$-database. Incomplete $\mathcal{K}$-relations are defined analogously. Queries over incomplete $\mathcal{K}$-databases are evaluated using possible worlds semantics, i.e., evaluating the query over each possible world using $\mathcal{K}$-relational semantics. That is, the result of a query is a set of $\mathcal{K}$-relations (possible worlds). Incomplete $\mathcal{K}$-databases can be trivially extended to probabilistic $\mathcal{K}$-databases by defining a distribution $P : \mathcal{D} \mapsto [0, 1]$ such that $\sum_{D \in \mathcal{D}} P(D) = 1$.

Certain and Possible $\mathcal{K}$-Annotations. We specifically consider semirings with a natural order $\preceq_{\mathcal{K}}$[1] that defines a lattice (also called l-semirings [45]). Assuming that the natural order is a total (resp., partial) order, we define the **certain** annotation of a tuple to be the minimum (resp., greatest lower bound) of the tuple's annotations across all possible worlds. Similarly, we define the **possible** annotation of a tuple to be the maximum (resp., least upper bound) across all possible worlds. We use $\sqcap_{\mathcal{K}}$ and $\sqcup_{\mathcal{K}}$ to denote the greatest lower bound (e.g., min in a total order) and lowest upper bound (e.g., max) operations for a l-semiring $\mathcal{K}$.

Under set semantics (where $F < T$ is the natural order), this gives us 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 case, certainty (possibility) is a lower (upper) bound on the multiplicity of the tuple across all possible worlds. As mentioned above, this semantics coincides with a proposal for certain and possible answers in bag-relational databases by Guagliardo et. al. [33]. For a set S of annotations from a semiring $\mathcal{K}$, these operations are defined as follows:

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

Example 3 *Consider the $\mathbb{N}$-database $\mathcal{D}$ (bag semantics) containing a single relation* `loc` *with two attributes* `locale` *and* `state` *and two possible worlds (denoted D_1, D_2) as shown below:*

D_1:

locale	*state*	$\mathbb{N}$
Lasalle	*NY*	***3***
Tucson	*AZ*	***2***

D_2:

locale	*state*	$\mathbb{N}$
Lasalle	*NY*	***2***
Tucson	*AZ*	***1***
Greenville	*IN*	***5***

The certain multiplicity of the tuple $\langle$ Lasalle, NY $\rangle$ *is* $\text{CERT}_{\mathbb{N}}(\{3, 2\}) = min(3, 2) = 2$. *Similarly, for the tuple* $\langle$ Greenville, IN $\rangle$ *(only present in possible world D_2) we get* $\text{CERT}_{\mathbb{N}}(\{0, 3\}) = 0$, *i.e., the tuple has certain multiplicity* 0. *Reinterpreting the example in set semantics, all tuples that exist (multiplicity larger than* 0*) are annotated with true (T) and all other tuples with false (F). Then for the first tuple we get,* $\text{CERT}_{\mathbb{B}}(\{T, T\}) = T \wedge T = T$, *i.e., the tuple is certain. Conversely for the third tuple we get* $\text{CERT}_{\mathbb{B}}(\{F, T\}) = F \wedge T = F$, *i.e., the tuple is not certain.*

Just like classical incomplete databases, incomplete $\mathcal{K}$-databases are used only as an abstract model to define clear semantics. More compact representations are needed. For example, existing incomplete data models like c-tables can be used to concisely encode incomplete $\mathbb{B}$-databases.

3.2 UA-Databases

Figure 5 provides an overview of the UA-DBs framework. In a typical scenario, a user would start from a compact encoding, e.g., c-tables, which represents an incomplete $\mathcal{K}$-database. Following [36] we use $Rep(\mathcal{D})$ to denote the incomplete database encoded by a model $\mathcal{D}$. Evaluating a query over this model using possible world semantics yields a result that encodes the query result over the incomplete $\mathcal{K}$-database. For both the input and the output we can determine the certain answers (assuming the existence of an algorithm *Cert* that realizes this step). Note that it is not possible to evaluate the query directly over the certain answers from the input since

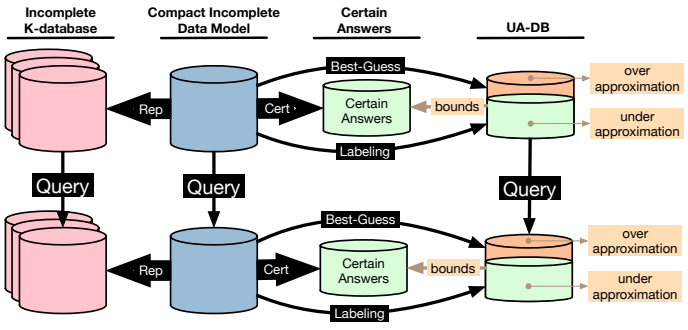

Figure 5: Representations of incomplete data and relationships between them.

[1]The natural order for a semiring $\mathcal{K}$ is defined as $a \preceq_{\mathcal{K}} b \Leftrightarrow \exists c : a +_{\mathcal{K}} c = b$.

certain answers are not closed under queries. A UA-DB can be derived from the incomplete data model by extracting the selected-guess world $D_\mathcal{G}$ (the over-approximation) and an under-approximation of certain answers $\mathcal{L}$ that we refer to as a *labeling*. We will discuss briefly below how compute such labelings. We have proven in [23] that positive queries over UA-DBs preserve these approximations. Hence, the result of the query evaluated over the UA-DB bounds ("sandwiches") the certain answers from below and above.

Consider $\mathcal{D}$, an encoding of an incomplete $\mathcal{K}$-database using some incomplete data model, e.g., V-tables [3]. Furthermore, let $D_\mathcal{G}$ be a distinguished possible world of $\mathcal{D}$ (selected-guess world) and $\mathcal{L}$ be a labeling for $\mathcal{D}$. An UA-DB for $D_\mathcal{G}$ and $\mathcal{L}$ is a $\mathcal{K}^2$-database: a database that labels each of its tuples with a pair of elements from $\mathcal{K}$ computed as follows:

$$D(t) = [D_\mathcal{G}(t), \mathcal{L}(t)] \qquad\qquad Q(D)(t) = [Q(D_\mathcal{G})(t), Q(\mathcal{L})(t)]$$

That is, the UA-DB D annotates each tuple with (1) the annotation from $D_\mathcal{G}$, and (2) the annotation from $\mathcal{L}$. Similarly, the result of query $Q(D)$ is a relation that labels its tuples with the annotations obtained by querying $D_\mathcal{G}$ and $\mathcal{L}$ independently. Since $D_\mathcal{G}$ and $\mathcal{L}$ are $\mathcal{K}$-databases this amounts to standard $\mathcal{K}$ query evaluation. Observe that this trivially preserves the over-approximation of certain answers, because by definition the certain answers are a lower bound on the result of a query over one possible world. The fact that the under-approximation is preserved by this approach is less obvious. In [23] we have proven that positive relational algebra over $\mathcal{K}$-relations preserves under-approximations. For specific incomplete data models, e.g., tuple-independent databases, UA-DBs compute precisely the certain answers. The same holds for certain classes of queries.

To demonstrate the backward-compatibility of UA-DBs with existing incomplete and probabilistic data models we have developed methods [23] for computing best-guess worlds and labelings (under-approximations of certain answers) for these models. We refer to the later as *labeling schemes*.

Example 4 *Consider a probabilistic version of the incomplete $\mathbb{B}$-relation from Example 3 where $P(D_1) = 0.4$ and $P(D_2) = 0.6$. One way to encode this database more compactly is to use a tuple-independent probabilistic database [67] (TIP-DB) as shown below. In a TIP-DB, tuples are assumed to be independent probabilistic events. The marginal probability $P(t)$ of each tuple t is stored in an attribute p. A world with the highest probability for a TIP-DB can be computed as the set of tuples that have probability larger than or equal to 0.5.[2] For TIP-DBs, certain answers can be computed efficiently by returning all tuples that have a probability of 1 and we can use this method as a labeling scheme. For instance, storing the annotations of a tuple in attributes `isBG` and `isCert`[3], we can compute a UA-DB from the TIP table `loc` using the SQL query:*

```
SELECT locale, state, 'T' AS isBG, CASE WHEN p>=0.5 THEN 'T' ELSE 'F' END AS isCert
FROM R WHERE p = 1
```

Applying this technique, we get the UA-DB shown below on the right. In this particular instance, the labeling $\mathcal{L}$ is exact. The first two tuples are both certain and labeled as certain (i.e., $D_\mathcal{G}(t) = \mathcal{L}(t) = T$), and so is annotated with $[T, T]$. The last tuple is possible but not certain, and so is annotated with $[T, F]$.

TIP-DB:

locale	state	p
Lasalle	NY	1
Tucson	AZ	1
Greenville	IN	0.6

UA-DB:

locale	state	$\mathbb{B}^2$
Lasalle	NY	**[T,T]**
Tucson	AZ	**[T,T]**
Greenville	IN	**[T,F]**

4 Research Thrust I - Formal Foundations of UA-DBs

In preliminary work on UA-DBs [23], we extended the classical notion of certain and possible answers for incomplete databases to support any type of semiring-annotated relation (generalizing bag semantics,

[2]Each subset of a TIP-DB is a possible world and the probability of a possible world is computed by multiplying the probabilities of tuples that are in the possible world and $1 - p(t)$ for tuples that are not in the possible world. The probability can be maximized by including tuples if $p(t) \geq 1 - p(t)$ which is the case if $p(t) \geq 0.5$.

[3]Note that the `isBG` attribute is redundant since it will be `'T'` for every tuple in the result of the query. Our implementations omits this attribute.

various types of provenance, access control, ...). We established UA-DBs as a light-weight model that labels selected-guess answers according to an approximation of certain answers. The selected-guess answers provide an over-approximation of certain answers, while the labeled tuples provide an under-approximation.

The work conducted as part of this proposal will significantly extend the formal foundation of UA-DBs by studying under which conditions (data, query, or semiring properties) the approximations of certain answers provided by UA-DBs are bounded (or exact). Furthermore, we will extend UA-DBs to support attribute-level annotations that bound the values of attributes across possible worlds. Finally, using attribute-level bounds and extending ideas from functional aggregate queries [40, 43, 44] to concisely encode possible answers, we will approximate certain answers for non-monotone queries such as queries with aggregation and negation.

4.1 I-a: Studying the Approximation in UA-DBs

Starting with set- and bag-semantics UA-DBs, we will study which properties of incomplete databases and queries cause UA-DBs to precisely identify certain answers. Furthermore, we will investigate whether we can guarantee any bounds on the approximation provided by UA-DBs.

In preliminary work we have demonstrated that positive queries (SPJ-U) over an UA-DB preserve the under-approximation and over-approximation of certain answers. However, we observe that under certain circumstances UA-DBs exactly encode the certain answers of a query. For instance, evaluating a positive query over any UA-DB generated from a TIP-DB returns a labeling that is precisely the certain answers. Even when this is not the case, initial experiments [23] suggest that in many real-world settings, labelings are close approximations of (or exactly equal to) certain answers. In [23] we did evaluate a large number of randomly generated projection queries over the result of missing value imputation for 9 real-world datasets and found that the percentage of certain answers mis-classified by our approach as uncertain is typically less than 5%. When using real-world queries, the error rate is even lower (less than 1% for the example queries we have tested). We propose to study how characteristics of the input *data* (e.g. the incomplete or probabilistic data model a UA-DBs is derived from) and of the *query* (e.g., structural parameters such as whether the query is hierarchical [19]) relate to the tightness of the approximation provided by a UA-DB. This information is useful for keeping the user aware of the degree of uncertainty in a UA-DB's query results For example, if we can establish that the error rate of a labeling is no more than 1% then it is reasonable to trust a UA-DB if it labels a result as uncertain.

4.2 I-b: Incomplete Databases and Certain Answers Beyond Sets and Bags

Although UA-DBs admit a natural extension to incomplete databases and certain answers beyond sets and bags, some proofs (e.g., conditions when approximations are exact) may not translate directly to arbitrary semirings. We will study the properties of these non-traditional cases and will investigate how approximation of certain answers is affected by the choice of annotation domain.

Our incomplete $\mathcal{K}$-relations and UA-DBs 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 and which are uncertain 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. Extending the result of research thrust I-a, we will study how the choice of semiring affects the precision of our approximation of certain answers. Furthermore, we will investigate novel applications enabled by incomplete databases beyond set semantics. For instance, the aforementioned incomplete databases with access control annotations would enable a rigorous treatment of access control for applications that analyze data that is the inherently uncertain result of information extraction, data cleaning, or data wrangling.

4.3 I-c: Bounding Values for Attribute-Level Uncertainty

We will extend the UA-DB model with attribute-level annotations that bound an attribute's values across all possible worlds, extend the definition of certain answers accordingly, study how these annotations propagate through queries, and investigate what certainty guarantees are provided.

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, 38, 41, 53, 63, 68], tracking uncertainty at the attribute-level can lead to more concise and precise representations of uncertainty.

Example 5 *Consider an employee table with two attributes: name and salary. Assume we know with certainty that Peter is an employee, but only know that his salary is either \$25,000, \$30,000, or \$40,000. This information can be modeled as an incomplete database with 3 possible worlds:* $D_1 = \{(Peter, 25000)\}$, $D_2 = \{(Peter, 30000)\}$, *and* $D_3 = \{(Peter, 40000)\}$. *Let us assume that we choose* $D_1 = \{(Peter, 25000)\}$ *as the selected-guess world. Even though part of the information in the single tuple of* D_1 *is certain (the name), it would be labeled as uncertain in a UA-DB because the tuple itself is not certain.*

We propose to enhance the UA-DB model with attribute-level annotations that encode lower and upper bounds on the values of attributes. The concept of over- and under-approximations of certain answers extends naturally to attribute-level UA-DBs, but requires us to now reason about the certainty of tuples that do not have the same values in every possible world.

Example 6 *Using attribute annotations, the salary attribute of this tuple from the example above would be annotated with a bound* $[25000, 40000]$. *Because this tuple matches exactly one tuple in each possible world that has a name of Peter and a salary falling within the bounds, we can label it as certain.*

Of course the same would be true if we choose a strictly wider bound for the salary (e.g., $[1000, 70000]$). Intuitively, the first bound is more precise and thus preferable. We plan to formalize this concept into a measure for how precisely an UA-DB with attribute-level annotations 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. Furthermore, we will investigate algorithms for computing queries over attribute-level UA-DBs and for translating incomplete data models into our model, and study the computational complexity of these problems. Finally, we will investigate alternatives to bounds for data types that do not have a meaningful ordering (e.g., name in the example): For example, as with rows, we might label attributes as certain (i.e., taking exactly one value across all possible worlds where the tuple appears) or uncertain (taking any value).[4]

4.4 I-d: Non-Monotone Queries with Compact Encodings of Possible Answers

Approximating certain answers for non-monotone queries (e.g., negation and aggregation) is challenging since non-monotonicity may turn an under-approximation into an over-approximation. We propose to employ and extend techniques for computing functional aggregate queries to compactly bound possible answer and in turn to efficiently compute under-approximations of certain answers.

The problem of computing an approximation of the certain answers to a query in the presence of (some) aggregate functions that "play nice" with a semiring $\mathcal{K}$ can be represented as a functional aggregate query, for which recent work presents algorithms with provable runtime guarantees [44]. We note that these algorithm hold for arbitrary functional aggregate queries over arbitrary semirings. One intriguing aspect of these algorithms is that they artificially impose a total ordering on the underlying semiring(s). One question we will explore in this proposal is whether we can exploit the fact that a semiring $\mathcal{K}$ comes equipped with a natural order to perhaps speedup the algorithms from [40, 43] specialized to our context. Secondly, we will consider the challenging task of handling negation. The results in [40, 43] cannot handle negations. For these queries we will turn our attention to a join query algorithm that is based on the following geometric view — instead of trying to figure out which potential tuples are part of the output, the algorithm in [42] tries to "rule out" tuples that cannot be part of the output. Since the set of non-answers is typically much larger than the set of answers, this algorithm relies on a geometric encoding of a relation and represents sets of non-answers compactly as boxes in such a space. We will extend these results for our settings. We note that there are implementation challenges in that the algorithms presented in [42] ignore (large) poly-log factors that impact real-life performance. However, we have been able to implement this algorithm in the context of a model-counter (i.e., a #-SAT solver) [21].

[4] This would be closely related to models which allow variables as values such as Codd-tables or V-tables.

5 Research Thrust II - Design and Implementation of U4U

Thrust II addresses the challenge of translating the principles developed in Thrust I into practice through a prototype UA-DB called *U4U* (*Uncertainty for You*). We will address logistical challenges involved in managing labeled data and then realize U4U in three stages. In stage 1, we will realize a purely rewrite-based implementation based on our preliminary work [23] for incomplete bag semantics databases. Then in stage 2 we will explore how augmenting the database through new data structures, algorithms, and cost-based optimization strategies can improve performance and accuracy. Finally, in stage 3 we will incorporate elements of existing incomplete (resp., probabilistic) database systems, enabling new forms of incremental approximation. We will assume for the sake of discussion that, except where noted, our source data consists of a selected-guess world that has already been labeled, for example using one of several existing data cleaning systems that label data as certain or possible [39, 53, 70] or one of our existing labeling adapters [23].

5.1 II-a: Management of Uncertainty-Labeled Data

We will extend SQL with tools for creating and updating uncertainty-labeled data.

Classical relational databases are deterministic: Data is provided as-is. For U4U to be useful — whether as a query rewriting middleware or a stand-alone database system — it needs uncertainty-labeled data. PIs Kennedy and Glavic have already explored labeling data in the context of the Mimir system [53, 70] where non-deterministic data cleaning operators called lenses label their outputs. We will develop, formalize, and realize two additional strategies for generating uncertainty-labels for relational data: (1) Adapting existing models for incomplete data, and (2) Incorporating uncertainty directly into SQL DML and DDL.

Adapters for Incomplete and Probabilistic Data. Numerous existing techniques have been developed for representing incomplete [36], probabilistic [31, 67], and fuzzy [15, 71] data. As part of this goal, we will implement labeling schemes such as those we have proposed [23] for TIP-DBs, block-independent relations, and c-tables. These adapters will be implemented as relational operators that operate on existing data. For example, consider a table of (potentially redundant) RSVPs and a user trying to convert this into a list of participants for each meeting.

```
CREATE TABLE RSVPS(email string, name string, event_id int, guests int);
CREATE TABLE PARTICIPANTS AS BLOCK_IND(RSVPS WITH (email,event_id) AS KEY);
```

The `BLOCK_IND` adapter operator acts as the MayBMS probabilistic repair-key [6], selecting a single selected-guess repair for user-specified key attributes (optionally weighted by a user-provided weight attribute or expression), and labeling the resulting rows and attributes with certainty and bounds respectively. We expect a wide range of adapters to be implementable through a combination of query rewriting and user-defined functions.

DML and DDL for Uncertainty-Labeled Data. Another practically relevant alternative for labeling is to incorporate it into SQL's data modeling (DML) and definition (DDL) languages. This requires us to develop a consistent semantics for manipulating uncertain data under possible-worlds semantics. The update semantics defined by Abiteboul et al. [2] could be employed here. We will investigate how this semantics translates to UA-DBs. For example, if the user learns that John Doe is likely to want to bring 3 guests in place of the number previously declared, they can update the RSVPs table as:

```
PROBABLY UPDATE PARTICIPANTS SET guests = 3 WHERE email = 'john@doe.org';
```

This update splits every existing possible world into two: one where the update occurs and one where it does not. Since there can only be one selected-guess world, prefixing the `UPDATE` with `PROBABLY` (resp., `POSSIBLY`) indicates that the update should be applied (resp., not applied) in the selected-guess world. We will first formalize semantics for creating and propagating labeled data through updates and schema operations. Then, we will implement these semantics, first through a rewrite-based approach based on PI Glavic's work on updates and transactions over annotated databases through reenactment [8–10] (a declarative replay technique for updates with annotated semantics), and later as a native version built into the database engine.

5.2 II-b: Query Rewriting for UA-DBs

We will develop techniques for rewriting queries with uncertainty annotations, ensuring that these queries run efficiently on existing relational databases, that the rewriting middleware supports tuple- and attribute-level uncertainty, and if time permits that they support uncertain provenance.

In [23] and [53], we developed a query rewriting middleware that implemented a bag semantics UA-DB with tuple-level uncertainty using a classical relational database. As the first stage of realizing U4U, we will extend this middleware with support for: (1) Attribute-level uncertainty, (2) Aggregation, and if time permits (3) semirings other than bags. The key challenge we will focus on at this stage is ensuring performance, while retaining backwards compatibility by not requiring any fundamental changes to the underlying database.

Efficiently Supporting Attribute-Level Uncertainty. We will translate the theory developed in goal I-c into practice by realizing it within U4U. A naive implementation of attribute bounds would extend each relation's schema with two additional attributes for the upper and lower bounds, respectively. For example, the schema of the rewritten `PARTICIPANTS` would add an $attribute^+$ and $attribute^-$ for each existing *attribute*, as well as a row-level uncertainty annotation `phi`[5]:

```
( email string, event_id int, guests int, email⁺ string, event_id⁺ int,
  guests⁺ int, email⁻ string, event_id⁻ int, guests⁻ int, phi boolean )
```

This naive approach roughly triples the size of all relations produced by a rewritten query. We will identify, implement, and evaluate more efficient strategies for storing these bounds. For example, storing deltas (relative or absolute) from the best guess value, rather than the actual bounds may be more compact. For example, consider that an attribute's value in the selected-guess world is 150 and the bounds for this value are $[140, 160]$. Given the best-guess value 150, this bound can be delta-encoded as $[-10, +10]$. This encoding is quite effective in applications like sensor data where (with high probability) the real value is close to the sensor reading. Similarly, schema-level rules (e.g., the upper bound is always infinite) preclude the need to include one or both bounds. It will also be important to consider that exact bounds may not be available (e.g., if an uncertain value is transformed by a non-monotone user-defined function). For such attributes, we will explore gracefully degrading to simpler boolean (i.e., certain vs uncertain) attribute-level annotations.

Supporting Aggregation. Aggregate queries introduce an additional layer of complexity. For example take the query: `SELECT SUM(guests) AS total FROM PARTICIPANTS`. The aggregation function result (the value of `total`) can be uncertain in multiple ways: (1) if the existence of at least one input tuple is uncertain, then the aggregation function result cannot be the same in worlds that include this tuple and worlds that do not include this tuple (unless the `guests` value of this tuple is 0 in every possible world); and (2) even if the existence of all input tuples is certain, the aggregation function result may still be uncertain if the `guests` attribute of one of these tuples differs across possible worlds (is uncertain). The following simplified query illustrates one naive (and incorrect) way to test for these conditions:

```
SELECT SUM(guests) AS total, TRUE AS phi, SUM(guests⁺) AS total⁺,
       SUM(CASE WHEN PHI THEN guests⁻ ELSE 0 END) AS total⁻ FROM PARTICIPANTS
```

Non-group by queries produce exactly one tuple in all possible worlds, so we consider the result tuple to certainly exist (`phi = TRUE`). The bounds on `total` take the best and worst cases assuming that attribute `guests` is non-negative. However, evaluated over a UA-DB, this query only considers results in the selected-guess world of `PARTICIPANTS`. If there exists a possible tuple not in the selected-guess world, then the computed upper bound is incorrect. In other words, to avoid having to mark all aggregated values as uncertain, we need to be able to efficiently reason about possible tuples. The problem becomes harder for group-by aggregate queries like `SELECT event_id, SUM(guests) AS total FROM PARTICIPANTS GROUP BY event_id`. In addition to the above challenges, the existence of 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 of aggregate values.

We will initially explore two approaches to this problem. First, we will consider tracking simple schema-level properties like whether a best-guess relation is "complete" (i.e., whether it is exactly the possible

[5]Following [23], `phi` is true for tuples that are certain.

tuples of the relation), or complete for specific groups. Second, we will consider evaluating aggregates in multiple passes (e.g., by combining multiple aggregate queries), for example by adapting a rewriting scheme by Guagliardo and Libkin [33], which over-approximates the impossible answers of a query. We will return to labeling aggregate values in goal II-c, when we consider new algorithms and data structures for UA-DBs.

Stretch Aim: Implementing Non-Bag Semantics. Time permitting, we will also explore implementing a UA-DB that supports annotations other than $\mathbb{N}$ or $\mathbb{B}$. One particularly interesting use case is uncertain provenance. For example, lineage [68] associates each query result tuple with the set of source tuples on which it depends. Applied to lineage, UA-DBs differentiate between tuples on which an output *may* depend, and tuples on which it *certainly* depends.[6] In settings like data cleaning, this distinction helps to prioritize work [17, 39] and provides more intuitive *qualitative* summaries of uncertainty [70].

The primary challenge we anticipate for non-bag semantics is supporting semirings with unbounded storage requirements. For example, the domain of provenance polynomials is the set of all polynomials. We will begin by adapting existing approaches for *deterministic* $\mathcal{K}$-relations. For example Orchestra [37] uses stores provenance polynomials referentially using temporary tables. ProvSQL [62] uses boolean circuits to store the full provenance of a tuple in a non-1NF relation. Conversely, PI Glavic's GProM [9] encodes provenance using multiple database rows (each storing a bounded-size fragment), and relying on extremely aggressive query optimization [59, 60] to limit the performance impact of this representation.

5.3 II-c: Database Engine Specializations for UA-DBs

We will identify, realize, and evaluate new data-structures, algorithms, optimization techniques, and other internal improvements that will make UA-DBs more efficient and expressive than they could be made through query rewriting alone.

Our second goal aims at supporting uncertainty-aware data management within existing relational databases. However, ensuring full backwards-compatibility precludes many opportunities for optimization. The third goal of this thrust is to consider architectural changes including new algorithms, data structures, and optimization techniques for improving UA-DB performance and utility.

Tetris for Over-Approximating Possible Answers. As noted above, support for uncertainty-labeled aggregate queries requires being able to determine the existence of tuples that are possible (but that may not be selected-guess answers). One approach to this problem is to use dual queries [33] to approximate the set of tuples that can not possibly be in the result (The impossible tuples of Figure 4). However, the intermediate results produced by this approach are large, as the set of impossible answers typically consists of most of the active domain [50]. Goal I-d explores the applicability of the Tetris join algorithm [21, 42], which avoids large intermediate states by incrementally ruling out progressively larger regions of the potential space of output tuples. In addition to implementing an adaptation of the algorithm specialized for UA-DBs, we will explore how the algorithms may be tuned for possible queries.

A (usually) Single-pass UA-DB Aggregation Operator. In general, aggregate queries over UA-DBs require multiple passes over their source data to effectively mark results as certain or not: (1) A pass to compute the selected-guess answers as normal, including determining the certainty of each tuple; (2) A pass to determine the existence of any possible rows with an uncertain group-by attribute value, which would make all group's aggregate values uncertain; and (3) A pass to determine whether there exist any possible rows with certain group-by attribute values, which would make one group's aggregate values uncertain. As a second part of this goal, we will explore optimization rules and runtime optimization techniques that can collapse passes, or eliminate the need for some of these passes. For example, if we can establish that the selected-guess answers includes all possible rows, we could execute the aggregate with one relational operator that performs all three passes in a single scan. Similarly, if the first pass identifies any selected-guess answers with uncertain group-by attributes, the latter two passes are redundant.

[6] Lineage in this sense corresponds to the semiring $\text{Which}(X)$ [30].

5.4 II-d: Online Refinement of UA-Rels

We will extend U4U to support exact certain (and probabilistic) queries over data by using UA-DBs as a pre-processing step to prune out certain (resp., $P = 1$) and uncertain (resp., $P = 0$) tuples before computing exact certainty (or probabilities).

The heavyweight machinery of computing certain answers using incomplete, probabilistic, and fuzzy databases (e.g., C-Tables [31, 36], U-Relations [7], VG-Functions [38], or VC-Tables [70]) is wasteful when most tuples (and/or attributes) in a dataset are certain. We will explore a novel approach to incomplete and probabilistic query processing that works in three stages: (1) Compute a preliminary result using UA-DBs, (2) Compute exact certainty, probabilities, expectations, and other relevant measures for tuples and attributes that the UA-DB result labels as uncertain (as having a range), and (3) Compute counts, marginal distributions, and other relevant measures over tuples not in the selected-guess answers. This three stage approach significantly benefits interactive analytics: (1) Based on preliminary experimental experience with real-world data [23], UA-DBs mis-classify comparatively few tuples, so the need for heavyweight computations is limited and (2) Results are produced incrementally as in online aggregation [35] and and approximate query processing [4][7]. We will implement and evaluate exact incomplete query processing in U4U. If time permits, we will then add support for probabilistic queries.

6 Intellectual Merit

The proposed research generalizes incomplete databases, probabilistic databases, and certain answers beyond set semantics. 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 UA-DBs, a new type of uncertain database and their approximation. 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 for attribute-level uncertainty. All of these expected results are also interesting research contributions in their own right. The inherent low complexity of query evaluation for UA-DB 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. 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 UA-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 uncertainty management. Finally, even though updates to incomplete data have been studied formally [2]), to the best of our knowledge no practical solution exists.

7 Broader Impacts

The state of the art in uncertainty management is what we call selected-guess query processing: An analyst or data ingest process selects one possible world and proceeds with analyses as if that world was true. In other words, uncertainty is dealt with by ignoring it. As we showed in [70] this model is applied in ETL (Extract-Transform-Load) pipelines. For instance, when imputing missing values as part of a ETL workflow, NULL values in a column are often replaced using a classifier to predict the "correct" value. However, the heuristic steps in a data curation workflow frequently admit alternative repairs of the data — they are uncertain, even if that uncertainty is presently discarded. UA-DBs have the potential for great practical impact since they combine the practicality and performance of selected-guess query processing with the rigor of certain answers. Our proposed techniques can significantly improve many real world use cases which currently make decisions based on uncertain data with severe negative impact. In addition to the potential of the proposed research itself, this grant will support three Ph.D. students and one postdoctoral researcher.

[7] Our proposed approach compliments anytime probabilities [24], which targets existential queries and individual tuples.

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 UA-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 effect analysis results should be taught to students starting from an early stage. The PIs will incorporate this topic into 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 about new releases and features.

Minority and Undergraduate Involvement. We are committed to recruiting and mentoring minorities. All the PIs have been involved in different efforts to foster minority involvement and we will use this proposal as a vehicle to further these efforts. A detailed description of efforts related to this proposal can be found in the *Broadening participation in computation plan* of the proposal.

Technology Transfer and Software Tools. Our team has an strong record of translating research results into practice. The software we will develop will be released as open source. Value-added open data derived by our methods will also will 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 UA-DBs has a small overhead compared to deterministic (best guess) query processing; (ii) the approximation accuracy of certain answers provided by UA-DBs is high for most real world use cases; (iii) attribute-level uncertainty encoded as bounds on attribute values increases the accuracy of approximations encoded by UA-DBs at a low additional performance penalty; (iv) 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 UA-DBs.

We will evaluate the performance of our implementation of UA-DBs in U4U over a wide range of real world datasets and synthetic benchmarks from the literature (e.g., PDBench [5]). We plan to use uncertain data where directly available: (1) the output of some information integration tools is probabilistic in nature [14,22], (2) Using probabilistic data cleaning operators called Lenses [70], we can create uncertain databases that are cleaned versions of real world datasets, but with explicitly modeled uncertainty. This enables us to evaluate our techniques over a range of publicly available (messy) open government datasets. We will experimentally compare the performance and approximation accuracy of our approach against other methods for approximating certain answers, against exact methods for computing certain answers, and with selected-guess query processing. Furthermore, we will compare UA-DBs with tuple-level uncertainty with the novel attribute-bound 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.

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 SUNY Buffalo and IIT (travel funds are requested as part of the budget). These day-long meetings will include progress reports and 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 the master and undergraduate level in projects related to the proposed research. The yearly meetings will also serve as a platform for these students to

meet in person and present their research results. 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 - Formal Foundations	Thrust II - Algorithms and Their Implementation in U4U
1	•**I-a** - study when UA-DBs are exact •**I-c** - extend UA-DB model for attribute-level uncertainty •**I-d** - investigate the applicability of FAQ and Tetris for over-approximating possible answers	•**II-a** - develop adapters for incomplete and probabilistic data models •**II-b** - develop rewrites for queries over set and bag semantics UA-DBs
2	•**I-a** - study bounds for certain answer approximations based on data and query characteristics •**I-c** - generalize the concept of certain answers to attribute-level UA-DBs	•**II-a** - investigate methods for preserving approximations of certain answers under updates •**II-b** - develop rewrites for attribute-level uncertainty UA-DBs
3	•**I-b** - study approximation bounds for additional semirings •**I-c** - develop query execution algorithms for attribute-level UA-DBs •**I-d** - extend FAQ and Tetris [42, 43] to support negation	•**II-a** - design and implement DDL and DML extensions for various tasks that generate uncertainty •**II-b** - extend rewrites for non-monotone queries (aggregation and negation) •**II-c** - adapt the Tetris algorithm [42] to efficiently compute over-approximations of possible answers
4	•**I-c** - study approximation bounds guaranteed by attribute-level UA-DBs •**I-d** - integrate the extended FAQ representation of possible answers to support aggregation and negation (set difference) queries	•**II-b** - develop rewrite techniques for additional semirings (e.g., provenance) •**II-d** - study online refinement methods for computing certain answers starting from a UA-DB query result •**II-c** - develop specialized single pass aggregation algorithms for UA-DBs •**II-d** - extend online refinement to support probabilistic databases and attribute-level uncertainty

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 [9, 11, 49, 50, 59, 65]. **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 7 Ph.D. students (minority female) and 3 undergraduate students (minorities).

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 [23, 52, 64, 66, 69], inter-modal provenance [27, 57], and interfaces for the above [46, 47], through a practical system called Vizier. **Broader Impact**: The Vizier system [32] 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 6 Ph.D. students (2 female) and one undergraduate student (minority 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 [54], (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 [42]; (iii) extending the join results to a more general framework [43], 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 [56]. 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] City of Chicago Data Portal - Chicago Crime Data. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2.

[2] Serge Abiteboul and Gösta Grahne. Update semantics for incomplete databases. In *VLDB'85, Proceedings of 11th International Conference on Very Large Data Bases, August 21-23, 1985, Stockholm, Sweden.*, pages 1–12, 1985.

[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. ACM Press, 1987.

[4] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. The aqua approximate query answering system. *SIGMOD Rec.*, 28(2):574–576, jun, 1999.

[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. IEEE Computer Society, 2007.

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

[8] Bahareh Arab, Dieter Gawlick, Vasudha Krishnaswamy, Venkatesh Radhakrishnan, and **Boris Glavic**. Using reenactment to retroactively capture provenance for transactions. *IEEE Transactions on Knowledge and Data Engineering*, 30(3):599–612, 2018.

[9] 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.

[10] Bahareh Sadat Arab, Dieter Gawlick, Vasudha Krishnaswamy, Venkatesh Radhakrishnan, and **Boris Glavic**. Reenactment for read-committed snapshot isolation. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 841–850, 2016.

[11] 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.

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

[15] 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. Springer US, 1999.

[16] 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. http://doi.acm.org/10.1145/1066157.1066277

[17] Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*, pages 1247–1261, 2015.

[18] E. F. Codd. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst.*, 4(4):397–434, dec, 1979.

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

[20] 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.

[21] Jimmy Dobler and **Atri Rudra**. Implementation of tetris as a model counter. *CoRR*, abs/1701.07473, 2017.
[22] Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. Probabilistic data exchange. *J. ACM*, 58(4):15:1–15:55, jul, 2011.
[23] 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.
[24] Robert Fink, Jiewen Huang, and Dan Olteanu. Anytime approximation in probabilistic databases. *VLDB J.*, 22(6):823–848, 2013.
[25] 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.
[26] Juliana Freire, **Boris Glavic**, **Oliver Kennedy**, and Heiko Mueller. The exception that improves the rule. *CoRR*, abs/1606.00046, 2016.
[27] Juliana Freire, **Boris Glavic**, **Oliver Kennedy**, and Heiko Mueller. The exception that improves the rule. In *HILDA*, 2016.
[28] **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.
[29] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.
http://doi.acm.org/10.1145/1265530.1265535
[30] Todd J. Green and Val Tannen. The semiring framework for database provenance. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 93–99, 2017.
[31] 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, *Current Trends in Database Technology – EDBT 2006*, volume 4254 of *Lecture Notes in Computer Science*, pages 278–296. Springer Berlin Heidelberg, 2006.
[32] The VizierDB Group. Vizier: Built for data exploration. http://vizierdb.info.
[33] Paolo Guagliardo and Leonid Libkin. Making SQL queries correct on incomplete databases: A feasibility study. In *PODS*, pages 211–223. ACM, 2016.
[34] Paolo Guagliardo and Leonid Libkin. Correctness of SQL queries on databases with nulls. *SIGMOD Record*, 46(3):5–16, 2017.
[35] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online aggregation. In *SIGMOD*, pages 171–182, 1997.
http://doi.acm.org/10.1145/253260.253291
[36] Tomasz Imieliński and Witold Lipski, Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, sep, 1984.
[37] Zachary G. Ives, Todd J. Green, Grigoris Karvounarakis, Nicholas E. Taylor, Val Tannen, Partha Pratim Talukdar, Marie Jacob, and Fernando Pereira. The orchestra collaborative data sharing system. *SIGMOD Rec.*, 37(3):26–32, sep, 2008.
[38] 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.
http://doi.acm.org/10.1145/1376616.1376686
[39] Shawn R. Jeffery, Michael J. Franklin, and Alon Y. Halevy. Pay-as-you-go user feedback for dataspace systems. In *SIGMOD Conference*, pages 847–860. ACM, 2008.
[40] Manas R. Joglekar, Rohan Puttagunta, and Christopher Ré. AJAR: aggregations and joins over annotated relations. 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 91–106, 2016.

[41] **Oliver Kennedy** and Christoph Koch. PIP: A database system for great and small expectations. In *ICDE*, pages 157–168. IEEE Computer Society, 2010.

[42] 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.

[43] 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.

[44] Mahmoud Abo Khamis, Hung Q. Ngo, and **Atri Rudra**. Juggling functions inside a database. *SIGMOD Record*, 46(1):6–13, 2017.

[45] Egor V. Kostylev and Peter Buneman. Combining dependent annotations for relational algebra. In *ICDT*, pages 196–207. ACM, 2012.

[46] Poonam Kumari, Said Achmiz, and **Oliver Kennedy**. Communicating data quality in on-demand curation. In *QDB*, 2016.

[47] Poonam Kumari and **Oliver Kennedy**. The good and bad data. In *NEDB*, 2018.

[48] 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.

[49] Seokki Lee, Bertram Ludäscher, and **Boris Glavic**. Provenance summaries for answers and non-answers. *PVLDB*, 11(12):1954–1957, 2018.

[50] Seokki Lee, Bertram Ludäscher, and **Boris Glavic**. Pug: a framework and practical implementation for why and why-not provenance. *The VLDB Journal*, August 2018.

[51] Leonid Libkin. Sql's three-valued logic and certain answers. *ACM Trans. Database Syst.*, 41(1):1:1–1:28, 2016.

[52] Niccolò Meneghetti, **Oliver Kennedy**, and Wolfgang Gatterbauer. Beta probabilistic databases: A scalable approach to belief updating and parameter learning. In *SIGMOD*, 2017.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] Xing Niu, Bahareh Arab, Dieter Gawlick, Zhen Hua Liu, Vasudha Krishnaswamy, **Oliver Kennedy**, and **Boris Glavic**. Provenance-aware versioned dataworkspaces. In *TaPP*, 2016.

[58] 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.

[59] 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.

[60] 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.

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

[62] Pierre Senellart, Louis Jachiet, Silviu Maniu, and Yann Ramusat. Provsql: Provenance and probability management in postgresql. *PVLDB*, 11(12):2034–2037, 2018.

[63] 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. http://doi.acm.org/10.1145/1376616.1376744

[64] 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.

[65] 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.

[66] 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.

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

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

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

[70] 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.

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