PROJECT OVERVIEW

Tools like personal sensing [29, 82], citizen science efforts such as eBird [38] or the Citizen Science Alliance [1], crowdsourcing [52, 116, 13], governmental open-data portals, and social media sources like Twitter's Firehose put into the public's hands a wealth of high-volume — but often low-veracity data. With data enthusiasts, scientists, reporters, and start-up companies all rushing to leverage this data for their own ends, **managing the data's reliability is more important than ever before**.

Historically, the reliability of a dataset would be ensured before it was analyzed, often through complex, carefully developed curation processes that attempt to completely shield the dataset's users from any and all uncertainty. As data sizes and rates have grown, maintaining these processes has become increasingly infeasible. Consequently, a "collect first, clean later" mentality has emerged, resulting in high-volume, high-velocity, and low-veracity data being dumped into so-called data lakes (or "data swamps" [124]). Although this approach seems lighter-weight at first, it simply defers the burden of curation to the analysis step, and in doing so encourages the use of brittle, one-off, or heuristic data cleaning solutions. [124].

Heuristic data cleaning is the closest to being sustainable. With minimal human intervention, good heuristics produce *mostly* reliable, *usually* actionable information from big, messy data. For instance, an analyst might automate their curation workflow by using classifiers to predict missing attribute values, or by using an entity-resolver to find and merge duplicate records. Unfortunately, heuristic solutions are also the most dangerous, as the result of heuristic curation is typically taken as fact. Serious mistakes result when low-confidence, or *uncertain* heuristic inferences are treated as truth [76]. As we will discuss shortly, many principled strategies presently exist for explicitly tracking potential errors in unreliable data. However, using these strategies is hard enough that analysts more often resort to ad-hoc, unreliable approaches like comment fields and README files. Our aim in this proposal is to provide a better option by **enabling data management systems capable of querying and organizing uncertain data, without being hard to use**.

When using a database system, users focus on the questions they want answered, while the system addresses mechanical concerns like which join algorithm is best or how data is stored on disk. Our aim is to do the same for uncertainty management by decoupling the process of asking questions about uncertain data, from mechanical concerns like why the data is uncertain, how the user wants to view uncertainty in query results, or which algorithms should be used. In short, this five-year proposal lays the foundations for research on **declarative uncertainty management**.

Our approach builds on decades of research on managing uncertain data [35, 69, 16, 66, 113, 114, 135, 21, 122, 71, 137, 25, 58, 70, 78, 81, 142] (only a small sample of the field). Although the problem of uncertainty in data is clearly not being ignored, existing models, formal representations, query processing systems, algorithms, tools, and techniques for handling uncertain or probabilistic data create a bewildering array of options for users. Making matters worse, selecting from among these options requires a careful understanding of which statistical guarantees and performance properties are needed, as well as what data models are available. Selecting the right options thus requires a convergence of backgrounds in statistics, database administration, and data modeling that no one individual database user is likely to have. Hence, while efforts to adapt research in this field of probabilistic query processing (PQP) to *specific* real-world challenges exist [45, 42, 48, 57, 70, 80], general-purpose tools for uncertainty-management remain predominantly limited to research prototypes.

Mimir. Our approach builds on preliminary work from PI Kennedy's ODIn lab on a new PQP system called Mimir [138, 141, 109, 123, 88], and leverages it as a way to *virtualize uncertainty*. In classical PQP systems, uncertainty is treated as a property of the data, which leads to uncertainty encodings that store data as a probability distribution over possible values. By contrast, Mimir treats uncertainty as a property of the process that created the data, encoding uncertainty as a form of provenance. Mimir is hardly the first to recognize the link between PQP and provenance [57, 56, 21, 135]. However, to the best of our knowledge we will be the first to leverage the link in order to decouple a user's analytical goals from orthogonal implementation details such as specific algorithms, data representations, or presentation formats.

Mimir takes tasks that are classically performed heuristically (i.e., data curation operations like schema alignment, missing-value imputation, type inference, or outlier removal) and encodes them through high-level abstractions that we call lenses [138, 141]. Lenses act as a form of *uncertainty-capture*, registering alternative outcomes to normally heuristic decisions. Once uncertainty is registered, users in an ideal declarative setting would not need to give any further thought to the fact that their data are uncertain — at least not until the uncertainty becomes relevant. Unfortunately, there currently exists a bewildering array of options available to users for visualizing, summarizing, and analyzing uncertainty, from many different perspectives. As a result, the cognitive burden of explicitly tracking data uncertainty is high, especially when compared to the *apparently* straightforward approach of simply applying heuristic solutions upfront and just querying the data, reliability be damned.

Goals. This proposal aims to realize the potential of declarative uncertainty management systems. By decoupling the process of querying data from the mechanics of managing uncertainty in the data being queried, *we will make it easier for analysts to incorporate uncertainty-awareness into their workflows.* If funded, this proposal will lay the foundations for a lifetime of research on declarative uncertainty management, leading to a new wave of safer, more reliable, and more trustworthy data management systems. As a way to guide our exploration of declarative uncertainty, our initial efforts will be driven by two thrusts:

Thrust 1: Summarizing Uncertainty Different PQP systems offer a range of different perspectives from which to view uncertain query outputs: Probabilities [114, 135, 25, 66], Statistical measures like expectations or variances [71, 78], Histograms [71], Rankings [45, 93], or Provenance [135, 141, 109]. As a consequence, users trying to ask questions about uncertain data also have to explicitly decide upfront how they plan to visualize uncertainty in the answers to those questions. The first thrust of this proposal explores strategies for decoupling these two decisions. Our approach is based on automatically generating summaries of uncertainty in query results by default, while retaining the flexibility to provide additional details if the user desires it. Key challenges that we address in this thrust include algorithms for efficiently constructing summaries, as well as strategies for managing how much detail is shown to the user at any one time.

Thrust 2: Compiler Support for Uncertainty Declarative uncertainty will require many fundamental changes in the design of query compilers. For this five-year proposal, we will address two specific considerations. First, research on PQP has led to a plethora of different algorithms and intermediate representations for probabilistic data, each optimized for specific use cases. Query compilers already automate the rote task of selecting between join algorithms, access paths, join orders, and more, based on the underlying data and the other needs of a query. We need compilers that do the same for uncertainty, selecting between the plethora of available algorithms for PQP. Second, query compilers classically treat schemas as static properties of a database. With uncertainty, this is no longer the case [96, 97, 123]. Although the general challenge of compilers for data with uncertain schemas will not be completely solved through this specific five-year effort, we will take the first steps by creating an *efficient* uncertainty-aware query type-checker.

1 — Background and Related Work

1.1 — Uncertain Query Processing

The majority of recent work on uncertain data management relies on so-called *possible worlds semantics* [17]. An uncertain database $\mathcal{D}$ instantiating schema $sch(\mathcal{D})$ is defined as a set of possible worlds: deterministic database instances $D \in \mathcal{D}$ instantiating schema $sch(D) = sch(\mathcal{D})$. Possible worlds semantics defines queries over uncertain databases in terms of deterministic query semantics. A deterministic query Q applied to an uncertain database defines a set of possible results $Q(\mathcal{D}) = \{\ Q(D) \mid D \in \mathcal{D}\ \}$. Note that these semantics (illustrated in Figure 1) are agnostic to the data representation, query language, and number of possible worlds $|\mathcal{D}|$. A *probabilistic database* $\langle \mathcal{D}, p : \mathcal{D} \to [0,1] \rangle$ is an uncertain database annotated with a probability distribution $p(D)$ with the standard normalization property: $\sum_{D \in \mathcal{D}} p(D) = 1$. Observe that queries induce a marginal distribution over possible result relations ($P[Q(\mathcal{D}) = R] = \sum_{D \in \mathcal{D}\,:\,Q(D)=R}\, p(D)$).

1.2 — Visual Representations of Uncertain Relational Data

The large number of possible worlds that could exist make it impractical to show probabilistic query result relations (i.e., $Q(\mathcal{D})$) directly as a set of possible relation instances. Instead, most work on PQP focuses on reducing this representation into a more easily consumed form, typically summarizing by individual tuples. At one extreme lie what are called *certain query answers*, commonly used in data exchange [50]. For a probabilistic query $Q(\mathcal{D})$ the certain answer (which we write $Q_{certain}(\mathcal{D})$) consists of the set of tuples that are present in the query result in *all* possible worlds (i.e., $\{\ t \mid \forall D \in \mathcal{D} : t \in Q(D)\ \}$). At the other extreme, we can consider the set of *possible query answers* ($Q_{possible}(\mathcal{D})$), which include all tuples that appear in the query result in *any* possible world (i.e., $\{\ t \mid \exists D \in \mathcal{D} : t \in Q(D)\ \}$).

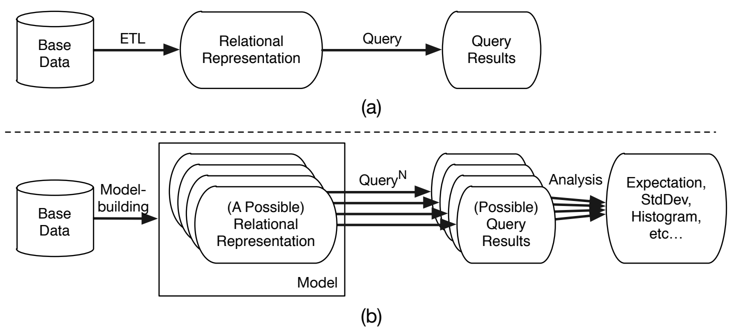


Figure 1: Normal query processing semantics (a) vs query processing under possible worlds semantics (b).

Certain and possible results represent two extremes of a spectrum. The former may omit potentially valuable information, while the latter might produce too many results. The search for an intermediate 'sweet spot' has led to the emergence of a variety of orthogonal summarization strategies that we broadly describe as either *summary-by-filtering* or *summary-by-aggregation*. Summary-by-filtering discards tuples that are likely to be irrelevant to the user — typically based on a metric called the tuple's *confidence*, or the marginal probability of the tuple existing in a randomly sampled query result (i.e., $conf(t) = \sum_{D \in \mathcal{D}\,:\, t \in Q(D)} p(D)$).

Examples of filtering mechanisms include: (1) **threshold** [53], which retains all tuples with a confidence that exceeds a given threshold, (2) **top-k posterior** [45, 92], which retains only the top-k most likely possible tuples in the output relation as ranked by their confidence, (3) **top-k prior** [141], which retains the union of all tuples in the result of the query on the k most likely possible *input* instances — We refer to the special case of the top-1 prior as **best guess** filtering — and (4) **sample** [71, 81], which retains the union of tuples drawn from a fixed-size sample of the possible output relation instances.

Summary-by-aggregation creates groups of different versions of the same tuple from across different possible worlds. An early example is x-tuples/or-set-tuples [21], which display tuples or attribute values in the form of sets of possible alternatives. More recent schemes further summarize attribute alternatives, displaying them as histograms [71], expectations [71, 78], confidence bounds [81], or hard bounds [79].

Example 1 *Consider the following four instances of a relation* $\texttt{Prod}(id, name, brand, type)$.

$\texttt{Prod}_1$	id	name	type	MSRP
	P123	Apple 6s	phone	**$400**
	P125	Note2	**phone**	$300

$\texttt{Prod}_2$	id	name	type	MSRP
	P123	Apple 6s	phone	**$900**
	P125	Note2	**phone**	$300

$\texttt{Prod}_3$	id	name	type	MSRP
	P123	Apple 6s	phone	**$400**
	P125	Note2	**tablet**	$300

$\texttt{Prod}_4$	id	name	type	MSRP
	P123	Apple 6s	phone	**$900**
	P125	Note2	**tablet**	$300

Note the two values that change between each instance (in bold text above). Part P123 has a price of $400 or $900, while part P125 has a type of 'phone' or 'tablet'. Applying summary-by-aggregation, we get a more compact summary representation [21, 66], where discrete values are replaced by sets of possible values.

Prod	id	name	type	MSRP
	P123	Apple 6s	phone	{**$400, $900**}
	P125	Note2	{**phone, tablet**}	$300

1.3 — Evaluating Queries on Uncertain Data

The large number of possible worlds that could exist also make possible worlds semantics equally impractical for query evaluation. Instead of evaluating queries on each individual possible world one at a time, existing systems adopt either a Monte-Carlo- or a Placeholder-based evaluation strategy.

Monte-Carlo. Monte-Carlo evaluation [71, 19, 81] substitutes the set of all possible instances of a relation with a bag of samples drawn from those instances. This is perhaps the most general evaluation strategy, as a naive implementation simply runs the (deterministic) query once on each sampled instance. Monte-Carlo evaluation was first proposed as part of the MCDB system [71], alongside an optimization called tuple-bundles. Tuple-bundles factorize the sampled relation instances by merging similar tuples from different possible worlds together to eliminate redundant computations on values that are consistent across all instances (i.e., determininstic). PI Kennedy's Jigsaw/FuzzyProphet [80, 81] system later used the same strategy for allowing non-statisticians to run what-if?-style queries on predictive analytics models. A key strength of Monte-Carlo evaluation strategies is that they impose no constraints on the query language, and only require that it be possible to draw samples from an uncertain relation. However, this generality comes at the cost of completeness: any statistics obtained about the results are necessarily approximations.

Placeholder. An alternative evaluation strategy relies on placeholders — analogous to promises/futures [95] — to defer realizing concrete instances of uncertain relations until after query evaluation is complete. Instead, query processing manipulates an internal representation where attributes, tuples, and databases are annotated with constraints and transformations that guide how the resulting relation can be instantiated. A common example of a placeholder-based strategy is called c-tables [68, 69], which uses labeled nulls to represent uncertainty. Labeled nulls appearing the relation itself encode missing or unknown attribute values. Each tuple is annotated with a boolean expression over labeled null symbols called a local condition (denoted ϕ), that determines the tuple's presence in any specific world. A boolean expression called the global condition restricts value assignments to labeled nulls. Any valuation v, or assignment of values to labeled nulls that satisfies the global condition defines a specific possible world denoted $\mathcal{D}[v]$.

Example 2 *Returning to Example 1, the results of a query $\sigma_{\texttt{category}='phone'}(\texttt{Prod})$ can be expressed as the C-Table:*

$\sigma_{\texttt{category}='phone'}(\texttt{Prod})$	**id**	**name**	**category**	**MSRP**	ϕ
	P123	Apple 6s	phone	$\langle X \rangle$	$\top$
	P125	Note2	$\langle Y \rangle$	\$300	$\langle Y \rangle = 'phone'$

Under valuation $v_1 = \{X \mapsto \$400, Y \mapsto 'phone'\}$, $\texttt{Prod}[v_1]$ corresponds to $\texttt{Prod}_1$ and the selection predicate passes all rows. Valuation $v_4 = \{X \mapsto \$900, Y \mapsto 'tablet'\}$ corresponds to $\texttt{Prod}_4$ and the relation resulting from the selection contains only one row because $\langle Y \rangle \neq 'phone'$.

Many PQP systems adopt some variation of c-tables for query evaluation, including MayBMS [16, 66], Sprout [113, 114], Trio [21, 135], Orion 2.0 [122], MystiQ [25], Orchestra [58, 70], and Katara [33, 22] as well as PI Kennedy's own system PIP [78].

Most c-tables-based PQP systems also add a probability measure that defines the likelihood of each valuation, and in turn each possible world. Such a pairing of a c-table and a probability measure is commonly referred to as a pc-table [56]. Critically, this distribution is closed under query processing over c-tables [56]. A distribution over valuations of the input instance to a query is valid for the output of the query as well. As a result, the placeholder-style output of a query over c-tables, coupled with the input probability measure, losslessly encode the distribution over possible results.

Unfortunately, computing tuple confidences from this encoding generally reduces to counting satisfying assignments to a boolean formula [40], a #P problem. Consequently, end-to-end PQP systems either resort to approximations outright [66], attempt analytical solutions where possible and fall back to approximation-based techniques otherwise [113, 78, 122], or rely on bounded-error approximations [53, 113] to allow early termination of normally exponential-time algorithms. A notable subclass of PQP systems aim specifically at computing top-k posterior or threshold representations [92, 45, 53]. These systems commonly rely on approximations to quickly assess which results are guaranteed to be in (or not in) the top-k/threshold result. The exponential-time exact marginal probability computations are necessary only for borderline tuples.

1.4 — Uncertainty Capture — Where does uncertainty come from?

PQP Systems. All work on PQP defines a precise model for representing uncertain data, but few address the construction of that model. Orion 2.0 [122] requires users to model their data using a library of constructors for common data distribution types. MCDB's Variable Generating, or VG-Functions [71] generalize this approach by allowing users to define new distributions as randomized procedures that sample from source distributions. PI Kennedy's systems: PIP [78] and Jigsaw [81, 80] take a similar approach. MayBMS [16, 66] constructs models through a non-deterministic repair-key operation that de-duplicates data, and similar approaches are used by Sprout [114, 113], Trio [21, 135], and MystiQ [25].

Model Databases. Several efforts have explored the use of SQL as a language for interacting with probabilistic models. MauveDB [44], BayesStore [132], and FunctionDB [127] allow relations to be defined by continuous functions (representing graphical models in the case of MauveDB and BayesStore). Pulse [12], Lahar [91], and SimSQL [28] focus specifically on time-series models — The latter two systems specifically target Markov processes. QUDE [23] and FIDES [125] explore uncertainty in the model itself, resulting from training errors like accidental p-hacking or biased training data. The idea of models as data also appears under the name Probabilistic Programming [85, 121, 115], which involve domain-specific languages for training and querying a specific classes of models based on the available data.

Data Exchange, Integration, and Cleaning. Data exchange [50, 49] and data integration manage data transformations across different representations and schemas. Overt errors — typically contradictions or missing values — need to be "repaired" before the database can present a consistent answer [49, 18]. Frequently, many such repairs are possible, creating uncertainty as to which repair is the correct one [22, 48]. The Orchestra system [58, 70], Dataspaces [51, 63, 130], and more (e.g., [98, 26, 27, 117]) all use a form of uncertainty capture to facilitate repairs. Data cleaning systems like Katara [33, 32, 22] and others [133, 32, 83, 47, 54, 84] adopt similar techniques, but for repairing self-inconsistent data based on constraints.

Approximate Query Processing. Approximate [10, 111, 112, 64, 60, 61] and online [64, 60, 61] query processing use samples of input data to quickly extrapolate approximations of final query results. The majority of work on AQP is on extrapolating progressively better approximations and error bounds for progressively larger fragments of SQL [9, 136, 31, 62, 118, 65, 11]. Approximations are also beneficial in distributed systems as a response to slow or failed nodes that would otherwise block query execution. Numerous efforts [24, 89, 55, 131, 126] capture uncertainty through constraints on result relations that can allow a blocked computation to be terminated early if the constrained result is actionable or sufficient for the user.

Predictive Interfaces. There has been significant recent interest in interfaces for data entry [30], data cleaning [74], speculative updates [59], and interactive querying [106, 108, 67]. Interface challenges often require aggressive heuristic interpretation, albeit with respect to the query rather than the data.

2 — Preliminaries — Uncertainty Virtualization

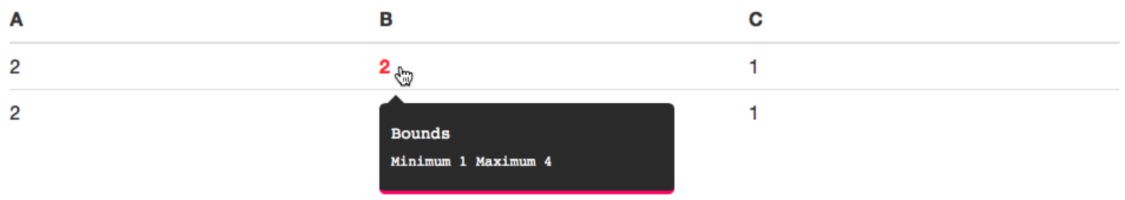
In most existing work on probabilistic databases, uncertainty is treated as a fundamental feature of data. As illustrated in Figure 1, probabilistic data is simply a set of possible versions — or more precisely, an encoding of the set of possibilities. By contrast, Mimir treats uncertainty as a feature of processes: The observable data in a system are fixed, but processes that act on that data may admit different interpretations, which in turn may lead to different outcomes. The immediate consequence of this shift in perspective is that "uncertain data" in Mimir are never materialized — Mimir works strictly with deterministic relations, supplemented with contextual metadata. Although not the original goal of avoiding materialization[1], an important consequence that this proposal aims to leverage is the added flexibility it brings. By not committing to any specific materialization strategy, Mimir is free to select from among a wide variety of visual uncertainty representations, data encodings, and query processing algorithms. We refer to this feature as *virtualized uncertainty*, and we now outline the the basics of how it functions.

Example 3 *Alice is evaluating the effects of a promotional offer for her company. Unfortunately some of her sources use non-standard terminology, while others contain duplicate information.*

`Prod`	**id**	**name**	**type**	**MSRP**
	P123	Apple 6s	phone	\$400
	P123	Apple 6s	phone	\$900
	P125	Note2	phablet	\$300

Primitive-valued expressions (i.e., expressions in selection, projection, or aggregation operators) in Mimir may reference special variable-generating terms (VG-Terms, denoted `Var[·](·)`). Conceptually, a VG-Term behaves like a skolem term, instantiating fresh skolem variables that act as placeholders for decisions that would otherwise need to be made heuristically — The result of any query that uses VG-Terms is effectively a c-table. Additionally, each VG-Term includes a reference to an independently created object called a *model* that manages the domain and probability distribution of the possible value assignments for one or

[1]The decision to avoid materializing uncertainty also facilitates backwards compatibility with classical relational databases [141].



A	B	C
2	2	1
2		1

Bounds
Minimum 1 Maximum 4

Figure 2: An example of Mimir's user interface for displaying relational tables.

more skolem variables — A query and set of models referenced by its VG-Terms together define a pc-table. Typically, VG-Terms are created as part of higher level abstractions called lenses [138, 141]. A lens captures general data-cleaning tasks like schema matching, key-repair, or missing-value imputation. The lens is responsible for instantiating both the view that encodes all possible clean versions of the data (i.e., by encoding heuristic decisions through VG-Terms), as well as any model objects used by those VG-Terms.

Example 4 *Continuing Example 3, Alice applies a type-repair lens to coerce `Prod.type` into a standardized form and a key-repair lens to eliminate duplicates in `Prod.id`. The resulting (lightly abbreviated) view query is:*

```
CREATE VIEW Prod_Clean AS
 SELECT id, name, MSRP, IF type IN ('phone', 'tablet', 'computer')
                        THEN type ELSE Var['TR:type'](id) END AS type FROM
 (SELECT id,
   IF COUNT(DISTINCT name)>1 THEN Var['KR:name'](id) ELSE FIRST(name) END AS name,
   IF COUNT(DISTINCT type)>1 THEN Var['KR:type'](id) ELSE FIRST(type) END AS type,
   IF COUNT(DISTINCT MSRP)>1 THEN Var['KR:MSRP'](id) ELSE FIRST(MSRP) END AS MSRP
  FROM Prod GROUP BY id)
```

In principle, this query would produce the c-table to the right, identical to that of Example 2 before applying selection.

Prod	id	name	type	MSRP
	P123	Apple 6s	phone	$\langle KR : name : P123 \rangle$
	P125	Note2	$\langle TR : type : P125 \rangle$	\$300

Observe that reliable data is preserved, while potentially erroneous data is removed and replaced with skolem symbols. Identifier arguments (e.g., `Prod.id`) deterministically assign names to the generated variables, allowing correlations to be expressed. As the lenses are created, Mimir also instantiates four model objects, named `KR:name`, `KR:type`, `KR:MSRP` (for key-repair), and `TR:type` (for type-repair). The model objects track possible assignments (e.g., $\langle KR : name : P123 \rangle \in \{\$400, \$900\}$), as well as a probability distribution over each (e.g., a uniform distribution in lieu of additional information). Say that Alice queries the data to assess the cost of a 10% discount:

```
SELECT SUM(MSRP * 0.1) FROM Prod_Clean p, Sales s
  WHERE p.id = s.product AND TODAY() - s.date < 1 month
```

The hypothetical result would be encoded as an expression $\$30 + \$30 + 0.1 \cdot \langle KR : name : P123 \rangle + \$30 + \ldots$, *which could be displayed visually (e.g., as a histogram) or as a statistical measure (e.g., an expectation or variance).*

VG-Terms are one of a variety of similar constructs found across databases and programming languages. Most directly related are the VG-Functions of MCDB [71] and PIP [78], which capture the semantics of how uncertainty arises in a dataset through an external "black-box" object analogous to Mimir's models. Green and Tannen's RA-Encodings [56] are also quite similar — In their approach, a single distinguished decision relation ($\mathcal{Z}$) encodes what we refer to as a valuation, while skolem terms are encoded as projections on this relation. Another closely related concept is promises/futures [95] from programming languages, which serve as placeholders for values that are not available yet, while still allowing the value to be manipulated in absentia. The variables generated by a VG-Term behave, in effect, as futures.

The similarity between VG-Terms and futures is critical, as queries containing VG-Terms are never evaluated directly. In previous versions of Mimir [141, 109], queries would be compiled down into a simpler form that could be evaluated on a classical relational DBMS to produce the heuristic or "best-guess" result. To compile the query, VG-Terms were replaced with references to specific values, as in the Green/Tannen RA-Encoding. While only a single relation was shown to users, definitive results would be distinguished from uncertain results through annotations [88] computed alongside the rest of the query [109]. For example, red text [88] as in Figure 2, indicates an unconfirmed heuristic decision "tainting" the marked attribute. As a cost of preserving backwards compatibility with relational DBMSes, Mimir never materializes probabilistic query results. Instead, probabilistic relations are "virtualized" as views [141], with the option of caching specific deterministic realizations of the view query (i.e., best-guess results) for improved performance.

The core of this proposal is the observation that uncertainty virtualization also allows us to de-couple queries from the mechanical concerns of managing uncertain data. Through the proposed work, we aim to demonstrate how uncertainty virtualization can be leveraged to automate important, but rote tasks such as visualizing uncertain query results, selecting algorithms for processing uncertain queries, and managing schema-level uncertainty. The resulting research will form the foundations of a new breed of safer and more reliable data management systems based on *declarative uncertainty management*.

3 — Thrust 1: Summarizing Uncertainty

Classical PQP requires analysts to be aware of uncertainty. Even systems that explicitly define general-purpose operators for analyzing uncertainty in relational data, such as the `conf` aggregate in MayBMS [66, 17] or the expectation aggregates of PIP [78], still require that users explicitly convert probabilistic results into deterministic summaries [15]. Our first thrust aims at decoupling the visualization of uncertain relations, from the process of posing the queries that create those relations in the first place. The goal is to mimic the classical deterministic database experience, while still communicating enough information for users to be able to act on potential errors. In our target application of heuristic data cleaning, the status quo is queries evaluated deterministically on a single realization of uncertain data, to produce a single best-guess (i.e., top-1 prior) result relation. From this starting point, we consider ways of annotating the best guess query results to better communicate uncertainty. For example, our preliminary user study [88] demonstrated that a simple relational table with potentially erroneous data highlighted (e.g., as in Figure 2) is sufficient to discourage users from acting on portions of a query result marked as uncertain.

For this proposal, we will focus on algorithmic concerns associated with two types of annotations: Qualitative Explanations and Correlations. Causal explanations [99, 102, 101, 119, 134, 109, 73] attempt to explain "why" a result appears (resp., does not appear) in a result set by identifying chains of potential database mutations (e.g., `INSERT`, `UPDATE`, or `DELETE` operations) that could lead to a result appearing (resp., not appearing). For *qualitative explanations*, we adopt a similar strategy, but only consider changes to skolem variables as allowable mutations. Our overaching goal for qualitative explanations, broken down into three sub-goals, is to organize these explanations into a hierarchical structure that analysts can use to manage how much or how little detail they see. A similar tradeoff between too much and too little detail arises when trying to present *correlations* in relational data. We propose three metrics — succinctness, precision, and accuracy — for selecting optimal representations from a search space of allowable designs.

Goal 1: Qualitative Uncertainty Analysis. The first goal will be to establish a common framework for qualitative analysis of uncertainty in Mimir. In Mimir, each non-deterministic decision made during heuristic data cleaning is identified by a single skolem variable. Mimir allows variables be translated into specific human-readable descriptions of those decisions [141, 109]. Thus, one approach to causal analysis is to enumerate all variables that affect the result relation. As a simple example, consider the query $\sigma_{\psi(A,B)}(R)$, which filters R based on a predicate ψ over attributes A and B. If R is deterministic, then uncertainty in the result can only arise through VG-Terms in ψ. With $f[X\backslash x]$ denoting formula f with X replaced by x:

R	A	B
t_1	`'foo'`	5
t_2	`'foo'`	6
t_3	`'bar'`	6
t_4	`'baz'`	7

Base Data

$\sigma_{\psi(A,B)}(R)$	A	B	ϕ
t_1	`'foo'`	5	$\psi[A\backslash\texttt{'foo'},B\backslash 5]$
t_2	`'foo'`	6	$\psi[A\backslash\texttt{'foo'},B\backslash 6]$
t_3	`'bar'`	6	$\psi[A\backslash\texttt{'bar'},B\backslash 6]$
t_4	`'baz'`	7	$\psi[A\backslash\texttt{'baz'},B\backslash 7]$

Generic C-Table

$\sigma_{\texttt{Var[]}(A)}(R)$	A	B	ϕ
t_1	`'foo'`	5	$\langle foo \rangle$
t_2	`'foo'`	6	$\langle foo \rangle$
t_3	`'bar'`	6	$\langle bar \rangle$
t_4	`'baz'`	7	$\langle baz \rangle$

With $\psi = \texttt{Var(A)}$

Continuing the example, assume that R is deterministic and that $\psi = \texttt{Var[](A)}$ — that is, that each result row is filtered based on a skolem variable (as before, denoted $\langle\cdot\rangle$) assigned based on the row's value for

A. Every row in the result is non-deterministic, and the corresponding explanation is given by a variable (either $\langle foo \rangle$, $\langle bar \rangle$, or $\langle baz \rangle$). An explanation for the full query includes all three variables.

Using this outline as a model, we can propose a naive strategy for finding all variables that affect the output of a query: (1) Identify every operator that uses a VG-Term, (2) Construct the operator's input relation(s), and (3) For each VG-Term `Var[](A,B,...)` compute all distinct values of $A, B, \ldots$ from the input relation(s).

This naive approach has three weaknesses that we will need to address. First, the approach does not produce minimal explanations. For example, tuples that are removed by a later selection operator will still contribute to the explanation. Similarly, conditional expressions may make an explanation data dependent, as in the above query, but with $\psi =$ `CASE WHEN A > 6 THEN Var[](A) ELSE TRUE END`. In this case, tuples t_1, t_2, and t_3 take the deterministic branch and uncertainty in the result is entirely due to $\langle baz \rangle$.

Second, variables can affect query results through uncertainty in attribute values, a tuple's presence, or sort orders. A framework for causal analysis will have to cope with all three forms of uncertainty.

Finally, and perhaps most importantly, this approach will typically produce very large numbers of explanations, making summarization essential. Our first summarization strategy leverages the existing association between VG-Terms and model objects. The model's identifier becomes part of the unique identity of each skolem variable constructed by the VG-Term. For example, a per-row missing-value imputation query might be encoded via the expression `CASE WHEN A IS NULL THEN Var['Impute'](ROWID) ELSE A END`. Here, 'Impute' becomes part of any variables instantiated by the VG-Term (e.g., $\langle Impute : 23 \rangle$ for row number 23). This shared identity drives the first level of our explanation hierarchy. We group variables — or more precisely their corresponding explanations — into *explanation families*.

The full set of all explanation families appearing in a query can be derived statically. However, it may be helpful to users if each family is communicated alongside a summary of its members, such as the number of distinct variables belonging to the family, or example explanations for specific variables belonging to the family. Furthermore, simply appearing in a query does not guarantee that the family has any members that contribute to uncertainty in the query's output. We will need to devise strategies for efficiently computing these summaries in order to achieve our first goal. For example, we may need to use approximations like sketching [37] the size of a family. We also expect that it will be more efficient to compute this statistic for all families together, as one large batch operation, a process made challenging by the fact that VG-Terms can appear anywhere in a relational algebra tree.

End Goal	Establish a framework for reasoning about causal explanations of uncertainty.
Deliverable	A scalable algorithm for summarizing explanation families.

Goal 2: Goal-Based Ranking. The second goal will be to filter explanations based on their impact on a query's results — a measure analogous to tuple influence [73] or responsibility [100]. Informally, influence can be thought of as the derivative of a value of interest in the query result (e.g., an aggregate value or confidence) with respect to changes in the probabilities of input tuples. Though related, our interest is on the impact of *skolem variables* on the *uncertainty in the results*. Hence, it will be necessary to extend existing work, like that of Kanagal and Deshpande [73] to measure the effect of *skolem variables* on *measures of uncertainty* such as entropy or standard deviation. Although we expect this extension to be straightforward, there are several opportunities for adjacent novelty. For example, assessing the joint contribution of an entire explanation family as a group is likely to present opportunities for batch evaluation.

It may be useful to also incorporate other factors into the ranking based on the user's goals. For example, a user doing data cleaning may be focused on low-hanging fruit: easy-to-clean sources of uncertainty that will have a high impact on her query results [141]. Conversely, a user trying to debug a query may have context-specific knowledge that causes her to quickly disregard certain types of uncertainty as culprits. In other words, if given weights on variables, explanation families, and/or general categories of model (e.g., type-repair or key-repair), we can better prioritize explanations that the user is likely to be interested in.

It is not realistic to expect the user to provide these weights directly, but it may be possible to infer them. For example, consider a data cleaning interface that shows users a list of explanations and creates the opportunity to either acknowledge or correct the heuristic decisions. A user skipping over highly ranked explanations sends a strong signal that these explanations are ranked too highly, and we can re-weight them. We plan to explore several approaches, including linear optimization and bayesian feedback to reweight them [105]. Time permitting, we will also explore how to incorporate feedback incrementally [110].

End Goal	Explore relevance metrics for summarizing explanations by filtering.
Deliverable	Algorithms for influence-based relevance and goal-based relevance.

Goal 3: Relaxed Dependency Models. The dependency model outlined in Goal 1 is strict: Any variable that could possibly affect the derivation of a given tuple or result value creates another explanation. In some cases however, the inclusion of one variable as an explanation may depend on the value assigned to another. For example, consider the expression `CASE WHEN Var(A) > 5 THEN Var(B) ELSE FALSE END`. Any given variable $\langle B\rangle$ is only a dependency when a corresponding variable $\langle A\rangle$ is assigned a value no greater than 5. In other words, each possible explanation $\langle B\rangle$ is *conditionally dependent* on the corresponding $\langle A\rangle$s.

Conditional dependencies create new opportunities for summarizing explanations by filtering out explanations that may not be immediately critical. A naive approach is to use the best-guess (top-1 prior) value of each variable to decide whether a conditional dependency is relevant. In the example above, if the best guess value for a variable $\langle foo\rangle$ was 2, then the expression evaluated on the row $\langle A:\texttt{'foo'}, B:\texttt{'bar'}\rangle$ would contribute $\langle foo\rangle$ as an explanation, but not $\langle bar\rangle$, since $2 \not> 5$. Extending the hierarchy from goal 1, we could nest hidden variables underneath the variables doing the hiding, For example we might indicate that *"this heuristic choice is preventing 10 other heuristic choices from being relevant"*. Like explanation families, nesting explanations requires us to efficiently generate summary statistics for sets of hidden variables.

Alternatively, we can use the probability of a conditionally dependent variable being hidden to impose a threshold- or top-k-style limit on the set of explanations shown. These computations are effectively probabilistic, existential queries. Although #P in general [41], there is substantial work on how to efficiently approximate the resulting probabilities [45, 113, 114, 53, 43] that we expect to be able to build on.

End Goal	Explore techniques for computing summaries from causal dependencies.
Deliverable	Scalable algorithms for summarizing explanations based on conditional dependence.

Goal 4: Correlations. The final goal of this thrust is to develop a process for effectively communicating correlations between attributes, tuples, and explanations in a displayed relation. Few PQP systems consider correlations in query outputs, and as a result their output may hide critical information. For example, a subtle schema matching error might lead to an imputation model being trained with a slight, but highly targeted bias. Knowing that a specific set of cells exhibit this unexpected correlation (i.e., the cells that the imputation model was biased against), an analyst might examine them more closely.

We will start with a simplified form of the problem: detecting correlations among sets of skolem variables. Recall that each skolem variable is drawn from a distribution defined by its corresponding model object. Hence, we can treat skolem variables backed by the same model object as probabilistic random variables X_i drawn from a joint distribution $\mathcal{P}(X_1, \ldots, X_N)$. Our preliminary goal then is to find a way to summarize the distribution $\mathcal{P}$. A solution developed for this base case could then be extended to the more general cases of correlations between cells (expressions over skolem variables), row-presence (boolean formulas over skolem variables), and sort orders (determined by expressions over skolem variables).

We could treat the problem as a form of structure discovery in graphical models [90], which aims to determine *all* correlated, or conditionally independent variables in a joint distribution. However, the goal of structure learning is to generate a complete representation, and not a succinct, representative summary intended for human consumption. As a result, it may be possible to devise a structure summarization strategy that draws on structure learning but avoids its typically high cost.

A second approach focuses on the information content of data representations for communicating correlations. Without focusing on any specific visual encoding of a relation's correlations (see Figure 3 for examples), we treat summaries as communicating *fragments of marginal distributions*. We will initially assume that all variables are binary (i.e., true or false), and define the fundamental unit of information conveyed by a representation to be a feature $F \subseteq [1, N]$, a set of variable identifiers. We treat a summary representation S as a *partial* mapping from features F to the marginal probability of the feature's variables all being true:

$$S(F) \equiv p\left(\bigwedge_{i\in F} X_i\right) = 1 - p\left(\bigvee_{i\in F} \neg X_i\right)$$

R	A	B
t_1	1	5
t_2	1	6
t_3	2	6
t_4	3	7

$p(t_1 \wedge t_2) = 0.4$
$p(t_3 \wedge t_4) = 0.15$
$p(t_2 \wedge t_4) = 0.6$

R	A	B	p
	1	5	0.5
	1	6	$\}$ 0.6
	3	7	
	2	6	0.2

Figure 3: Example representations of correlated features: Annotations (left) or grouped tuples (right).

We define three measures applicable to any visual encoding of correlated data that fits into this framework: (1) Succinctness, (2) Precision, and (3) Accuracy. A baseline measure for succinctness is simply the number of features given by the summary (i.e., $|S|$). To measure precision and accuracy, we consider each feature to be a constraint on the user's perception of the joint distribution $\mathcal{P}_S$. Taking a uniform prior, we can model the user's perception as a distribution $p(P = \mathcal{P}_S \mid S)$ over the space of distributions allowed by the marginal probabilities given by S. This model allows us to define precision as the entropy of the distribution of $\mathcal{P}_S$. Similarly, accuracy can be modeled in terms of the expectation of the distribution, for example by taking the K-L divergence between $E[\mathcal{P}_S]$ and $\mathcal{P}$.

This model provides a framework for trading off between succinctness, precision, and accuracy. When combined with constraints from specific families of visual representations (e.g., the grouped-tuples representation in Figure 3 assigns each variable to exactly one feature) any representation on the optimal tradeoff curve can be obtained, for example with a linear solver. Once we have an algorithm for the basic model, we will continue by generalizing away assumptions like the uniform prior or variables being boolean.

End Goal	A framework for reasoning about representations of correlated tabular data.
Deliverable	An algorithm for selecting between correlation-aware representations of c-tables.

4 — Thrust 2: Compiler Support for Uncertainty

Our second thrust explores the implications of declarative uncertainty on compiler design. Both goals show how virtualized uncertainty can be leveraged during the query compilation process, first to evaluate queries faster, and second to make authoring queries easier in the presence of uncertainty.

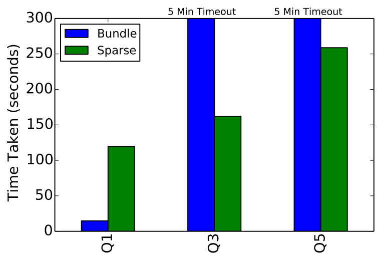
Goal 1: Query Optimization. Probabilistic queries introduce new degrees of freedom to query optimization. For instance, the background section introduced a wide range of sample- and placeholder-based query evaluation techniques, while Thrust 1 introduced a "best guess" evaluation strategy. Each of these strategies has strengths and weaknesses — placeholder-based evaluation becomes expensive on aggregate queries, best-guess evaluation has limited representational utility, and sampling has its own challenges that we will turn to shortly. The most efficient strategy for any given query varies with respect to the operators it uses, which specific inputs are uncertain, and even properties of the uncertainty (e.g., whether a variable is drawn from a finite and discrete distribution). The first goal of this thrust is to design a query optimizer capable of efficiently selecting an appropriate evaluation strategy for a given query and data.

To make the problem concrete, our initial focus will be on a surprising tradeoff in sampling-based query evaluation. Assume we are given a query $\mathcal{Q}(D)$ with schema A, and that we would like to generate N samples of $\mathcal{Q}(D)$ (denoted $\mathcal{Q}_1(D), \ldots, \mathcal{Q}_N(D)$). Concretely, we would like to construct a version of the query $\mathcal{Q}^N(D)$ with schema $\langle A_1, \ldots, A_N, \mathbb{W}_1, \ldots, \mathbb{W}_N \rangle$, where $\forall i < N \;:\; \mathcal{Q}_i(D) \equiv \pi_{A_i}(\sigma_{\mathbb{W}_i}(\mathcal{Q}^N(D)))$. That is, each tuple contains N copies of schema A, with each copy A_i mapping to the value of A in world i sampled from the original query. The boolean attributes $\mathbb{W}_i$ indicate whether a tuple is present in each sampled world. These semantics mirror those of MCDB's tuple-bundles [71].

Clearly, one way to achieve this result is to use tuple-bundles directly. Figure 4a summarizes the general process of rewriting an SPJU bag-relational algebra query into the desired form (i.e., such that $[\![Q]\!]_{TB}(D) \equiv Q^N(D)$). Unfortunately, this is not always the right thing to do. Figure 4b uses the benchmark from [109] to compare the performance of tuple bundles with an alternative, *sparse* encoding which simply maintains one entire copy of each tuple in each possible world, annotated with the index of the world it belongs to. Results shown in the figure include the cost of constructing a result in the format described above. In spite of the output format being explicitly tailored for tuple-bundles, *tuple bundles were slower on complex queries*.

$$\begin{aligned}
[\![R(A)]\!]_{TB} &\equiv \pi_{A_i \leftarrow A, \mathbb{W}_i \leftarrow \mathbf{T}}(R(A)) \\
\left[\!\!\left[\pi_{B \leftarrow f(A)} R(A)\right]\!\!\right]_{TB} &\equiv \pi_{B_i \leftarrow f_i(A_i)} [\![R(A)]\!]_{TB} \\
\left[\!\!\left[\sigma_{\phi(A)} R(A)\right]\!\!\right]_{TB} &\equiv \sigma_{\bigvee_i (\mathbb{W}_i)} \left(\pi_{A_i \leftarrow A_i, \mathbb{W}_i \leftarrow \phi_i(A_i) \wedge \mathbb{W}_i}([\![R(A)]\!]_{TB})\right) \\
[\![R(A) \times S(B)]\!]_{TB} &\equiv \sigma_{\bigvee_i (\mathbb{W}_i)} \left(\pi_{A_i \leftarrow A_i, B_i \leftarrow B_i, \mathbb{W}_i \leftarrow R.\mathbb{W}_i \wedge S.\mathbb{W}_i}([\![R(A)]\!]_{TB} \times [\![S(B)]\!]_{TB})\right) \\
[\![R(A) \uplus S(A)]\!]_{TB} &\equiv [\![R(A)]\!]_{TB} \uplus [\![S(A)]\!]_{TB}
\end{aligned}$$

(a) A simplified tuple-bundle rewrite $[\![\cdot]\!]_{TB}$ that assumes all attributes are non-deterministic. f_i and ϕ_i denote the result of evaluation in the specified world.



(b) Comparison of Sampling on TPC-H Queries Q1, Q3, and Q5.

Figure 4: Tuple Bundles

To understand why this is, note the predicates $\bigvee_i (\mathbb{W}_i)$ in the rewrites for selection and cross-product, which filter out tuple-bundles that have already been eliminated from all sampled worlds. This check makes joins incredibly expensive. Classical hash-based join algorithms are not as effective at pre-filtering potential joins, as tuples join if the condition $\phi_i(A_i, B_i)$ is satisfied in *any* sampled world. Clearly, if the condition depends only on deterministic attributes, this is not a problem. However, non-deterministic equi-joins effectively reduce to performing N independent equi-joins, one for each version of the tuple. The overhead resulting from needing to re-assemble the tuple bundle after the join, outweighs the benefits of the tuple bundle's reduced redundancy.

In summary, we have two representations: tuple-bundles and sparse tuples. The two representations encode identical information, and it is possible to transform either representation into the other. Furthermore, certain relational operators naturally inline with the shift between representations. For example, a probabilistic equi-join could be implemented without the cost of re-assembling tuple bundles by emitting a sparse representation instead. Similarly, aggregates can inexpensively link sparse tuples back into tuple-bundles. The goal then, is to select an optimal query plan accounting for the different representational options, transformations between the representations, and the costs of algorithms for each.

Optimizing the evaluation of sampling queries is only one step in the process of building an uncertainty-aware query optimizer. For example, rather than sampling, we could simply enumerate all options of a finite, discrete distribution, (e.g., as in the world set decompositions of MayBMS [16]), or use placeholder-style evaluation. Another factor to consider when optimizing probabilistic queries is how the results will be used. For example, the added overhead of deferring sampling through placeholder-style evaluation may be worth it if the number of samples needed is not known ahead of time [78].

End Goal	Design, implement, and evaluate an uncertainty-aware query optimizer.
Deliverable	An optimizer for plans that hybridize tuple-bundle and sparse-tuple encodings.

Goal 2: Schema-Level Uncertainty. Schema evolution [39], semistructured data [46, 123, 20, 72], and natural-language interfaces [94], all create situations where precise relational schemas are not available. Although it may still be possible to enumerate a set of *possible* schemas [123, 72], classical data management systems are ill equipped to handle any more than just one (often *heuristically* selected) "optimal" schema. If chosen poorly, this schema may lead to errors that can be be hard to diagnose and even harder to fix.

As the second goal for this thrust, we aim to help users to diagnose and fix errors caused by uncertainty at the schema-level. Our approach re-visits the type-checker, a part of query compilers typically responsible for annotating queries with schema information. Type-checkers also serve to sanity check the query, for example by identifying nonsensical operations (e.g., `(3 / 'foo')`) or references to non-existent attributes. Thus, any unexpected consequences of a schema selection mistake will first appear in the type-checker.

As in the prior goals, our aim with the type-checker is to preserve the existing deterministic database user experience, while also making it easier to detect and recover from errors. We will first need a representation of uncertain schemas that preserves backwards compatibility with classical type-checking (i.e., based on

$$\begin{aligned}
[\![R(A)]\!]_{TC} &\equiv \pi_{Name,Type}\left(\sigma_{Table='R'}(\mathbf{attr\_catalog})\right) && (1)\\
[\![\pi_{B\leftarrow A}(R(A))]\!]_{TC} &\equiv \pi_{Name\leftarrow'B',Type\leftarrow Type}(\sigma_{Name='A'}([\![R(A)]\!]_{TC}) && (2)\\
[\![\pi_{C\leftarrow A+B}(R(A))]\!]_{TC} &\equiv \pi_{Name\leftarrow'C',Type\leftarrow\mathbf{escalate}_{+}(1.Type,2.Type)}\left(\sigma_{Name='A'}([\![R(A)]\!]_{TC}) \times \sigma_{Name='B'}([\![R(A)]\!]_{TC})\right) && (3)\\
[\![\sigma_{\phi}(R(A))]\!]_{TC} &\equiv [\![R(A)]\!]_{TC} && (4)\\
[\![R(A) \times S(B)]\!]_{TC} &\equiv [\![R(A)]\!]_{TC} \uplus [\![S(B)]\!]_{TC} && (5)\\
[\![R(A) \uplus S(B)]\!]_{TC} &\equiv [\![R(A)]\!]_{TC} \cap [\![S(B)]\!]_{TC} && (6)
\end{aligned}$$

Figure 5: Parts of a simplified type-checker query mapping $[\![\cdot]\!]_{TC}$ for SPJU bag-relational algebra.

heuristically selected schemas). While backwards-compatible, this representation should encode sufficient information to allow a more advanced type-checker to quickly diagnose schema errors and suggest fixes.

Our initial approach will be to re-cast type-checking as a form of relational query. We will devise a scheme for rewriting normal relational queries $Q(D)$ into corresponding *type-inference queries* $[\![Q]\!]_{TC}\,(sch(D))$. Given the relational encoding of a database's schema, the type-inference query constructs a relational encoding of the corresponding query's schema. Examples of this rewriting process are given in Figure 5.

It is possible to use lenses to generate relational encodings of uncertain schemas [123]. For example, consider a scheme developed by DiScala and Abadi [46] for shredding de-normalized data into multiple normal-form relations. This scheme relies on a heuristically selected spanning tree over a graph of functional dependencies. We can encode the scheme in Mimir by using a repair-key lens [123] to create a probabilistic, relational encoding of all possible spanning trees, in turn encoding all possible output schemas. Thus, by creating a type-checker that operates over relational encodings, it becomes possible to leverage results of the work outlined previously in this proposal to realize uncertainty-aware type-checking.

Our primary goal is to be able to mimic the deterministic database experience: Computing the *best-guess results* for a type-inference query achieves this end[2]. If this mode detects an error (e.g., a missing attribute or a type mismatch), we can instead compute all *possible results* of the type-inference query to determine whether there exists any schema configuration under which the query makes sense. We can then leverage qualitative explanations or correlation summaries from Thrust 1 to better explain what schema changes need to happen, or why the result is an error. The user can then apply any appropriate remedial actions.

Re-casting the type-checker in terms of relational queries accomplishes three immediate goals. First, relational queries provide us with precise, intuitive semantics for propagating uncertain schemas. Second, a relational type-checker can leverage, for free, the mountain of existing work on probabilistic query processing. Finally, a relational type-checker, when combined with existing work on triggers and incremental view maintenance could, in principle, be used to implement guards for schema requirements, preventing after-the-fact schema adjustments from breaking existing view definitions or database applications. In spite of these benefits, it may not be possible to define a clean relational encoding for some parts of the type-checker (e.g., type escalation — Equation 3 in Figure 5). As a result, we suspect that creating an efficient relational type-checker will require either special-purpose extensions to relational algebra, or that we simply use the relational encoding as a template for an implementation in a lower-level language.

End Goal	Realize and evaluate a query compiler capable of managing schema-level uncertainty.
Deliverable	A relational-algebra type-checker for non-deterministic schemas.

5 — Evaluation Plan

5.1 — Benchmarks, Metrics and Deliverables:

Both thrusts will be evaluated on a mix of datasets and workloads that include elements of uncertainty. For example, MayBMS [14] and MCDB [71] were both evaluated on probabilistic adaptations of the popular TPC-H benchmark [128]. Further datasets and query workloads will be drawn from resources like the MIT Patient Monitoring Project [120], the UCI Machine Learning Repository [8], as well as governmental open data portals, including the upcoming City of Buffalo portal [34].

[2]In practice, we will use a classical type-checker for normal type-checking, as type-checking queries will also need type-checking.

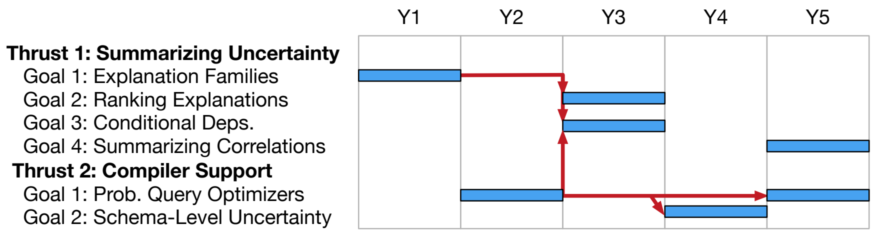


Figure 6: Anticipated timeline and dependencies for proposed goals

The research goals of this proposal are purely algorithmic. However, it may be desirable to obtain feedback about user-facing components of the proposed work. Here, we expect to be able to leverage PI Kennedy's involvement in NSF ACI Award #1640864 (CIF21 DIBBs: EI: Vizier, Streamlined Data Curation), which aims to implement a system for simultaneous data exploration and curation. As Mimir is a core component of this project, we expect that our partners — Oracle, Airbus, the NYU Center for Urban Science and Progress, and others — will be able to provide us with insights that can guide our algorithmic efforts. We do not expect to need user studies for the proposed work[3].

Qualitative Explanations. For goal 1.1, our primary metric for success will be to demonstrate that our framework can construct complete and minimal qualitative explanations of uncertainty. Performance, in particular of algorithms for summarizing explanation families, will be a secondary measure of success. Goals 1.2 – 1.3 both propose algorithmic contributions, so demonstrating their efficiency and scalability will be our primary measure.

Correlations. Goal 1.4 outlines a wide design space. One measure of success would simply be a demonstrably correct, efficient optimizer for pre-defined tradeoffs between succinctness, precision, and accuracy. Secondary measures of success would be defined classes of visualizations that can be generated quickly, as well as methods for solving the problem that scale beyond the limits of typical linear solvers.

Uncertainty-Aware Optimizers. A clear measure of success for goal 2.1 is whether optimized plans match or beat both fixed-format sampling algorithms, in terms of performance and scalability. A secondary measure of quality will be time required to select an optimal plan.

Type-checker for Uncertain Schemas. If we are able to create a probabilistic type-checker, goal 2.1 will have succeeded. However, there are numerous secondary metrics such as the cost of type-checking in both modes, as well as the type-checker's scalability on complex schemas with many possible options.

5.2 — Timeline

Figure 6 outlines the anticipated timeline for the project. Explanation families serve as a lightweight introduction to potential students for the project, and motivate work on compiler optimization as a means of efficiently computing probabilities for conditional dependencies. Schema level uncertainty will require the same optimizations. Time permitting, we will consider other PQP algorithms in the optimizer in Year 5.

Risk Analysis and Mitigation. The most significant risk in this project is that one or more algorithmic challenges proposed do not admit an efficient solution. If this happens, we will focus on approximation algorithms, while attempting to identify classes of problem where exact solutions are feasible. A second risk is that the family of interface designs that we are targeting turns out to be not useful. To mitigate this risk we will emphasize rapid prototyping of demonstrable interfaces that we can use to solicit feedback early on. If significant changes are required, we will pivot as needed.

6 — Integrating Education and Research

PI Kennedy is an avid supporter of educational outreach and has been involved in K-12 STEM programs in the Buffalo area, including Science is Elementary [7], Coder Dojo [2], Liberty Partnerships [5], the Interdisciplinary Science and Engineering Partnership [3], and the ACM's Computer Science Teachers Association

[3]If the situation changes, we will notify the cognizant NSF official and obtain approval from UB's IRB before proceeding.

(CSTA). PI Kennedy's outreach efforts presently include running summer programs through the Liberty Partnerships, and periodic data science workshops for students at local area high-schools.

Uncertainty-awareness as a form of pedagogy. Many of the techniques proposed, when realized, serve as a form of pedagogy. Through NSF ACI Award #1640864 (CIF21 DIBBs: EI: Vizier, Streamlined Data Curation), we hope to release Mimir as a tool to the public, where it can serve as a form of tangential learning. Declarative uncertainty makes it much easier for data hobbyists — potential users without a formal background in data science or statistics — to more safely jump into data science. By giving data hobbyists insight into errors, for example through qualitative explanations, Mimir helps them to learn potential types of errors that can occur in the data analysis process. In effect, declarative uncertainty can lead to systems that provide "training wheels" for data science.

Data4Good Workshops. Through his contacts with the ACM CSTA, PI Kennedy has been running periodic workshops on data science at Williamsville North High School under the monicker Data4Good. To date, the intent of these workshops has been simple awareness and exposure: The workshops introduce students to using open-data portals, and expose them to the tools that they will need for analyzing the data. As an example, a recent after-school workshop conducted in May illustrated the use of JuPyTer notebook to analyze crime data from the Detroit open data portal. The workshop guided students through the process of finding the data, loading it into JuPyTer online, and detecting outliers through visualizations.

If this proposal is funded, PI Kennedy will be able to devote more time to the Data4Good workshops, expanding them into a week-long program (e.g., a 'camp') to be conducted during the summer or winter break. Key material that he hopes to cover during this time includes: (1) Data and where to find it, (2) Rudimentary data models: Spreadsheets and DataFrames (e.g., in NumPy/Pandas), (3) Transforming and visualizing data with DataFrames and JuPyTer notebook, and (4) Basics of Geospatial data. PI Kennedy intends to have four days of intensive teaching and hands-on-lab components, followed by a one-day hackathon, where participants join into teams to construct interesting visualizations. In addition to providing a valuable service to the community, PI Kennedy hopes to be able to use this as an opportunity to derive new insights about how people approach uncertainty, in turn driving the core goals of this proposal.

Advancing UB's Graduate Databases Course. PI Kennedy is lead instructor for two database courses at UB: CSE-4/562 (Database Systems), and CSE-662 (Languages and Runtimes for Big Data). Kennedy plans to integrate material from the proposed work into both courses. CSE-4/562 already regularly includes segments on probabilistic data and representations. CSE-662 is a project-based course where students are offered a range of seed projects, of which they pick one to implement as part of a team. PI Kennedy has a history of integrating his research efforts into this course. In the two years that the course has existed, seed projects completed by students have led to insights related to IIS Awards #1617586 (III: Small: Just in Time Datastructures) and #1629791 (CI-P: Supporting Pocket Scale Data Management Research).

6.1 — Prior and Ongoing Educational and Outreach Efforts:

K-12 STEM Education. For two years PI Kennedy volunteered as a classroom leader with Buffalo's chapter of *Science is Elementary*, a program that works with local elementary schools in **predominantly-minority neighborhoods** to excite children about the scientific method and encourage them to ask questions about the world they live in. PI Kennedy is now in his fifth summer of volunteering with *Liberty Partnerships*, a program that prepares high- and middle-school students from **underprivileged neighborhoods** to attend college. In his first two years with Liberty Partnerships, PI Kennedy participated in a program of short-term internships where students shadow mentors in science and engineering fields. Since then, Liberty started a program of summer camps for middle-school students from underprivileged schools As part of this program PI Kennedy has been regularly teaching "sections" of a simulated introductory CS course, where students gain confidence by discovering that they are capable of the material that they will be learning in these courses. Finally, PI Kennedy participated in a short-lived program called *CoderDojo-Buffalo*, which paired **middle-school students with mentors** as they learn coding skills. Although CoderDojo-Buffalo no longer exists, PI Kennedy models aspects of his Data4Good workshops on it.

Graduate and Undergraduate Education. PI Kennedy considers equalizing gender ratios in computer science to be one of his top priorities. He presently advises 4 PhD students (1 female) and co-advises 4 PhD students. He has recently completed independent studies or similar research projects with 3 MS students (1 female). PI Kennedy is an avid supporter of undergraduate research and is presently mentoring 3 BS students (2 female), in addition to mentoring 5 BS students who have already graduated. PI Kennedy teaches UB's Graduate Database Systems course (CSE-562), one of UB's most popular CSE courses. Starting in Fall 2017, this course will be available to undergraduates (as CSE-462). PI Kennedy also co-teaches a project based course entitled Languages and Runtimes for Database Systems (CSE-662), which he and his co-instructor Lukasz Ziarek have used to get graduate students involved in their research activities.

Small Data. At ICDE 2017, PI Kennedy organized a panel on "Small Data" [77], which emphasized the need for data management systems that work with people, rather than working to replace them. The work outlined in this proposal represents one step in this direction. If funded, PI Kennedy will be able to continue his efforts on small data management systems like Mimir through workshops, tutorials, and editorials.

Community Organization. In addition to PI Kennedy's educational outreach efforts, he co-leads the Buffalo Database Seminar Meetup along with Charlie Wertz of Buffalo State University, and Tyler Poland of local startup EngineYard. The meetup is a monthly meeting of database enthusiasts from industry and local educational institutions, featuring regular talks about database techniques and technology.

7 — Broader Impacts of the Proposed Work

The current status quo of treating heuristic inferences as fact is unacceptable Already, we are seeing significant impacts on people's lives: Numerous high profile instances of misuse of heuristics has led to everything from children being mistaken for terrorists [6], to people being denied access to critical financial services due to someone else's bad credit [4]. Manual curation would be ideal, but is unsustainable due to the sheer volume and velocities of data that modern analysts need to work with. What remains is a need for easy to use tools for working with uncertain data that help users to differentiate between trustworthy and non-trustworthy inference. If fully realized and developed, declarative uncertainty has the potential to transform analytics as we know it by making it easier to work with unreliable data, spawning a new wave of safer, more reliable tools for uncertainty-aware analytics.

PI Kennedy has a history of deploying his research efforts into practice through the release of code artifacts, including PIP [75] and DBToaster [36]. Likewise, Mimir has been released as a well documented prototype [129] with 11 watchers, 8 stars, and 8 forks on GitHub. The proposed work promises to lead to numerous improvements in Mimir, making it both faster and easier to use. These improvements will be incorporated into Mimir for public release. As already mentioned, PI Kennedy is collaborating with NYU and the Illinois Institute of Technology on an NSF-supported effort to build a user interface called Vizier for exploring messy datasets while curating them. Vizier also provides us with a deployment path for many of the ideas outlined in this proposal. As a core part of Vizier, any functionality added to Mimir can be directly translated into improvements to Vizier.

Our partners on Vizier, including the NYC Taxi and Limousine Commission, as well as the NYU Center for Urban Science and Progress, demonstrate that there is keen interest in tools for uncertainty management. There is also significant interest from industry for uncertainty-aware query processing. Many of PI Kennedy's efforts on PQP have resulted from collaborations with industry including Jigsaw (Microsoft) [80, 81] and Mimir (Oracle, Airbus) [79, 141, 109, 123]. Ying Yang, the first student to work on Mimir was recently hired by Oracle for a related project.

8 — Results from Prior NSF Support

PI Kennedy has been tenure-track faculty since Fall of 2012 and has been a PI/CoPI on 4 NSF awards. **Intellectual Merit:** Since receiving his first award 2.5 years ago, PI Kennedy's NSF funding has facilitated 3 conference papers [105, 140, 123], 2 workshop papers [88, 86], 3 technical reports [109, 87, 104], 1 masters thesis [107], and 2 PhD theses [103, 139]. **Broader Impacts:** NSF: ACI-1640864 (2017-2019; $2.7m; "CIF21 DIBBs: EI: Vizier, Streamlined Data Curation") is funding the unification of Mimir with provenance systems GProM and VisTrails, as the basis for a new tool for exploring and curating messy data. Since being funded half a year ago, the project has already linked the three systems, and is exploring user interface designs and prototype deployments. PI Kennedy's NSF funding has supported the professional development of 6 PhD students (2 female), and through REU supplements, independent studies, and coursework have led to research opportunities for 3 MS students (1 female) and 8 BS students (3 female). Dr Kennedy is currently advising 4 PhD students (including 1 female student) and co-advising 4 PhD students.

REFERENCES

[1] The citizen science alliance. http://www.citizensciencealliance.org.

[2] Coder dojo. http://coderdojo.com/.

[3] The interdisciplinary science and engineering partnership. http://isep.buffalo.edu.

[4] John Oliver rips into the credit reporting industry. http://money.cnn.com/2016/04/11/pf/john-oliver-credit-reports/index.html.

[5] Liberty partnerships. http://www.libertypartnerships.com.

[6] Meet Mikey, 8: U.S. Has Him on Watch List. http://www.nytimes.com/2010/01/14/nyregion/14watchlist.html?

[7] Science is elementary. http://elementaryschoolscience.org.

[8] The UC Irvine machine learning repository. http://archive.ics.uci.edu/ml/.

[9] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. *SIGMOD Rec.*, 28(2):275–286, June 1999.

[10] S. Agarwal, A. P. Iyer, A. Panda, S. Madden, B. Mozafari, and I. Stoica. Blink and it's done: Interactive queries on very large data. *Proc. VLDB Endow.*, 5(12):1902–1905, Aug. 2012.

[11] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. Jordan, S. Madden, B. Mozafari, and I. Stoica. Knowing when you're wrong: Building fast and reliable approximate query processing systems. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 481–492, New York, NY, USA, 2014. ACM.

[12] Y. Ahmad, O. Papaemmanouil, U. Cetintemel, and J. Rogers. Simultaneous equation systems for query processing on continuous-time data streams. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 666–675, April 2008.

[13] Y. Amsterdamer, S. B. Davidson, T. Milo, S. Novgorodov, and A. Somech. Oassis: Query driven crowd mining. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 589–600, New York, NY, USA, 2014. ACM.

[14] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *ICDE*, pages 983–992. IEEE Computer Society, 2008.

[15] L. Antova and C. Koch. On apis for probabilistic databases. In *QDB/MUD*, pages 41–56, 2008.

[16] L. Antova, C. Koch, and D. Olteanu. MayBMS: Managing incomplete information with probabilistic world-set decompositions. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 1479–1480, April 2007.

[17] L. Antova, C. Koch, and D. Olteanu. $10^{(10^6)}$ worlds and beyond: Efficient representation and processing of incomplete information. *The VLDB Journal*, 18(5):1021–1040, Oct. 2009.

[18] M. Arenas and L. Libkin. Xml data exchange: Consistency and query answering. *J. ACM*, 55(2):7:1–7:72, May 2008.

[19] S. Arumugam, F. Xu, R. Jampani, C. Jermaine, L. L. Perez, and P. J. Haas. Mcdb-r: Risk analysis in the database. *Proc. VLDB Endow.*, 3(1-2):782–793, Sept. 2010.

[20] M. A. Baazizi, H. B. Lahmar, D. Colazzo, G. Ghelli, and C. Sartiani. Schema inference for massive JSON datasets. In *EDBT*, pages 222–233. OpenProceedings.org, 2017.

[21] O. Benjelloun, A. D. Sarma, C. Hayworth, and J. Widom. An introduction to uldbs and the trio system. *IEEE Data Eng. Bull.*, 29(1):5–16, 2006.

[22] G. Beskales, I. F. Ilyas, L. Golab, and A. Galiullin. Sampling from repairs of conditional functional dependency violations. *The VLDB Journal*, 23(1):103–128, Feb. 2014.

[23] C. Binnig, L. D. Stefani, T. Kraska, E. Upfal, E. Zgraggen, and Z. Zhao. Toward sustainable insights, or why polygamy is bad for you. In *CIDR*. www.cidrdb.org, 2017.

[24] P. Bonnet and A. Tomasic. Partial answers for unavailable data sources. In T. Andreasen, H. Christiansen, and H. L. Larsen, editors, *Flexible Query Answering Systems, Third International Conference, FQAS'98, Roskilde, Denmark, May 13-15, 1998, Proceedings*, volume 1495 of *Lecture Notes in Computer Science*, pages 43–54. Springer, 1998.

[25] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. Mystiq: A system for finding more answers by using probabilities. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 891–893, New York, NY, USA, 2005. ACM.

[26] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. Olap over uncertain and imprecise data. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 970–981. VLDB Endowment, 2005.

[27] D. Burdick, A. Doan, R. Ramakrishnan, and S. Vaithyanathan. Olap over imprecise data with domain constraints. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 39–50. VLDB Endowment, 2007.

[28] Z. Cai, Z. Vagena, L. Perez, S. Arumugam, P. J. Haas, and C. Jermaine. Simulation of database-valued markov chains using simsql. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 637–648, New York, NY, USA, 2013. ACM.

[29] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn. The rise of people-centric sensing. *Internet Computing, IEEE*, 12(4):12–21, July 2008.

[30] K. Chen, H. Chen, N. Conway, J. M. Hellerstein, and T. S. Parikh. USHER: improving data quality with dynamic forms. In F. Li, M. M. Moro, S. Ghandeharizadeh, J. R. Haritsa, G. Weikum, M. J. Carey, F. Casati, E. Y. Chang, I. Manolescu, S. Mehrotra, U. Dayal, and V. J. Tsotras, editors, *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, pages 321–332. IEEE, 2010.

[31] S. Chen, P. B. Gibbons, and S. Nath. Pr-join: A non-blocking join achieving higher early result rate with statistical guarantees. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 147–158, New York, NY, USA, 2010. ACM.

[32] X. Chu, I. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 458–469, April 2013.

[33] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD Conference*, pages 1247–1261. ACM, 2015.

[34] City of Buffalo Department of Management Information Systems. Request for proposals open data portal and content management system. https://www.ci.buffalo.ny.us/files/1_2_1/city_departments/Purchasing/OpenData.pdf, February 2017.

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

[36] T. D. Consortium. DBToaster. http://www.dbtoaster.org.

[37] G. Cormode. *Count-Min Sketch*, pages 511–516. Springer US, Boston, MA, 2009.

[38] Cornell Lab of Ornothology. ebird. http://ebird.org/.

[39] C. A. Curino, H. J. Moon, and C. Zaniolo. Graceful database schema evolution: The prism workbench. *Proc. VLDB Endow.*, 1(1):761–772, Aug. 2008.

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

[41] N. Dalvi and D. Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6):30:1–30:87, Jan. 2013.

[42] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang. The data civilizer system. In *CIDR*. www.cidrdb.org, 2017.

[43] A. Deshpande, L. Hellerstein, and D. Kletenik. Approximation algorithms for stochastic boolean function evaluation and stochastic submodular set cover. In *SODA*, pages 1453–1467. SIAM, 2014.

[44] A. Deshpande and S. Madden. Mauvedb: Supporting model-based user views in database systems. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, pages 73–84, New York, NY, USA, 2006. ACM.

[45] L. Detwiler, W. Gatterbauer, B. Louie, D. Suciu, and P. Tarczy-Hornoch. Integrating and ranking uncertain scientific data. In *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, pages 1235–1238, March 2009.

[46] M. DiScala and D. J. Abadi. Automatic generation of normalized relational schemas from nested key-value data. In *SIGMOD Conference*, pages 295–310. ACM, 2016.

[47] A. Ebaid, A. Elmagarmid, I. F. Ilyas, M. Ouzzani, J.-A. Quiane-Ruiz, N. Tang, and S. Yin. Nadeef: A generalized data cleaning system. *Proc. VLDB Endow.*, 6(12):1218–1221, Aug. 2013.

[48] R. Fagin, B. Kimelfeld, and P. G. Kolaitis. Probabilistic data exchange. *J. ACM*, 58(4):15:1–15:55, July 2011.

[49] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89 – 124, 2005. Database Theory.

[50] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, Mar. 2005.

[51] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: A new abstraction for information management. *SIGMOD Rec.*, 34(4):27–33, Dec. 2005.

[52] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: Answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 61–72, New York, NY, USA, 2011. ACM.

[53] W. Gatterbauer and D. Suciu. Dissociation and propagation for approximate lifted inference with standard relational database management systems. *VLDB J.*, 26(1):5–30, 2017.

[54] F. Geerts, G. Mecca, P. Papotti, and D. Santoro. The llunatic data-cleaning framework. *Proc. VLDB Endow.*, 6(9):625–636, July 2013.

[55] A. Gheerbrant, L. Libkin, and C. Sirangelo. When is naive evaluation possible? In R. Hull and W. Fan, editors, *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 75–86. ACM, 2013.

[56] T. Green and V. Tannen. Models for incomplete and probabilistic information. In T. Grust, H. Höpfner, A. Illarramendi, S. Jablonski, M. Mesiti, S. Müller, P.-L. Patranjan, K.-U. Sattler, M. Spiliopoulou, and J. Wijsen, editors, *Current Trends in Database Technology – EDBT 2006*, volume 4254 of *Lecture Notes in Computer Science*, pages 278–296. Springer Berlin Heidelberg, 2006.

[57] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, pages 31–40, New York, NY, USA, 2007. ACM.

[58] T. J. Green, G. Karvounarakis, N. E. Taylor, O. Biton, Z. G. Ives, and V. Tannen. Orchestra: Facilitating collaborative data sharing. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 1131–1133, New York, NY, USA, 2007. ACM.

[59] N. Gupta, A. J. Demers, J. Gehrke, P. Unterbrunner, and W. M. White. Scalability for virtual worlds. In Y. E. Ioannidis, D. L. Lee, and R. T. Ng, editors, *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pages 1311–1314. IEEE, 2009.

[60] P. Haas. Large-sample and deterministic confidence intervals for online aggregation. In *Scientific and Statistical Database Management, 1997. Proceedings., Ninth International Conference on*, pages 51–62, Aug 1997.

[61] P. J. Haas and J. M. Hellerstein. Ripple joins for online aggregation. *SIGMOD Rec.*, 28(2):287–298, June 1999.

[62] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *Proceedings of the 21th International Conference on Very Large Data Bases*, VLDB '95, pages 311–322, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[63] A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems. In *Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '06, pages 1–9, New York, NY, USA, 2006. ACM.

[64] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, SIGMOD '97, pages 171–182, New York, NY, USA, 1997. ACM.

[65] W.-C. Hou and G. Ozsoyoglu. Statistical estimators for aggregate relational algebra queries. *ACM Trans. Database Syst.*, 16(4):600–654, Dec. 1991.

[66] J. Huang, L. Antova, C. Koch, and D. Olteanu. MayBMS: A probabilistic database management system. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 1071–1074, New York, NY, USA, 2009. ACM.

[67] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Tutorial*, Melbourne, Australia, 2015.

[68] T. Imieliński and W. Lipski, Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, Sept. 1984.

[69] T. Imielinski and W. Lipski Jr. On representing incomplete information in a relational data base. In *Proceedings of the seventh international conference on Very Large Data Bases-Volume 7*, pages 388–397. VLDB Endowment, 1981.

[70] Z. G. Ives, T. J. Green, G. Karvounarakis, N. E. Taylor, V. Tannen, P. P. Talukdar, M. Jacob, and F. Pereira. The orchestra collaborative data sharing system. *SIGMOD Rec.*, 37(3):26–32, Sept. 2008.

[71] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. Jermaine, and P. J. Haas. Mcdb: A monte carlo approach to managing uncertain data. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 687–700, New York, NY, USA, 2008. ACM.

[72] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy. Pay-as-you-go user feedback for dataspace systems. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 847–860, New York, NY, USA, 2008. ACM.

[73] B. Kanagal, J. Li, and A. Deshpande. Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In *SIGMOD Conference*, pages 841–852. ACM, 2011.

[74] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Wrangler: interactive visual specification of data transformation scripts. In D. S. Tan, S. Amershi, B. Begole, W. A. Kellogg, and M. Tungare, editors, *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, pages 3363–3372. ACM, 2011.

[75] O. Kennedy. The PIP MayBMS plugin. http://maybms.sourceforge.net.

[76] O. Kennedy. Stop the truthiness and just be wrong (abstract). CIDR-Abstract, 2017.

[77] O. Kennedy, D. R. Hipp, S. Idreos, A. Marian, A. Nandi, C. Troncoso, and E. Wu. Small data (panel discussion), 2017.

[78] O. Kennedy and C. Koch. Pip: A database system for great and small expectations. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 157–168, March 2010.

[79] O. Kennedy, Y. Yang, J. Chomicki, R. Fehling, Z. H. Liu, and D. Gawlick. Detecting the temporal context of queries. In *International Workshop on Business Intelligence for the Real-Time Enterprise (BIRTE)*, 2014.

[80] O. A. Kennedy, S. Lee, C. Loboz, S. Smyl, and S. Nath. Fuzzy prophet: Parameter exploration in uncertain enterprise scenarios. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 1303–1306, New York, NY, USA, 2011. ACM.

[81] O. A. Kennedy and S. Nath. Jigsaw: Efficient optimization over uncertain enterprise data. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 829–840, New York, NY, USA, 2011. ACM.

[82] P. Klasnja, S. Consolvo, D. W. McDonald, J. A. Landay, and W. Pratt. Using mobile & personal sensing technologies to support health behavior change in everyday life: lessons learned. In *AMIA*, 2009.

[83] S. Kolahi and L. V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *Proceedings of the 12th International Conference on Database Theory*, ICDT '09, pages 53–62, New York, NY, USA, 2009. ACM.

[84] L. Kot and C. Koch. Cooperative update exchange in the youtopia system. *Proc. VLDB Endow.*, 2(1):193–204, Aug. 2009.

[85] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan. Mlbase: A distributed machine-learning system. In *CIDR*. www.cidrdb.org, 2013.

[86] G. Kul, D. T. Luong, T. Xie, P. Coonan, V. Chandola, O. Kennedy, and S. Upadhyaya. Ettu: Analyzing query intents in corporate databases. In *ERMIS*, 2016.

[87] G. Kul, D. T. Luong, T. Xie, P. Coonan, V. Chandola, O. Kennedy, and S. Upadhyaya. Summarizing large query logs in ettu. Technical report, ArXiv, 2016.

[88] P. Kumari, S. Achmiz, and O. Kennedy. Communicating data quality in on-demand curation. In *QDB*, 2016.

[89] W. Lang, R. V. Nehme, E. Robinson, and J. F. Naughton. Partial results in database systems. In C. E. Dyreson, F. Li, and M. T. Özsu, editors, *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 1275–1286. ACM, 2014.

[90] P. Larrañaga, M. Poza, Y. Yurramendi, R. H. Murga, and C. M. H. Kuijpers. Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(9):912–926, 1996.

[91] J. Letchner, C. Re, M. Balazinska, and M. Philipose. Access methods for markovian streams. In *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, pages 246–257, March 2009.

[92] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *Proc. VLDB Endow.*, 2(1):502–513, Aug. 2009.

[93] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *VLDB J.*, 20(2):249–275, 2011.

[94] Y. Li, H. Yang, and H. V. Jagadish. Nalix: A generic natural language search environment for XML data. *ACM Trans. Database Syst.*, 32(4):30, 2007.

[95] B. Liskov and L. Shrira. Promises: Linguistic support for efficient asynchronous procedure calls in distributed systems. In *PLDI*, pages 260–267. ACM, 1988.

[96] Z. H. Liu, A. Behrend, E. Chan, D. Gawlick, and A. Ghoneimy. Kids-a model for developing evolutionary database applications. In *DATA*, pages 129–134, 2012.

[97] Z. H. Liu and D. Gawlick. Management of flexible schema data in rdbmss - opportunities and limitations for nosql -. In *CIDR*. www.cidrdb.org, 2015.

[98] B. Louie, L. Detwiler, N. Dalvi, R. Shaker, P. Tarczy-Hornoch, and D. Suciu. Incorporating uncertainty metrics into a general-purpose data integration system. In *Scientific and Statistical Database Management, 2007. SSBDM '07. 19th International Conference on*, pages 19–19, July 2007.

[99] A. Meliou, W. Gatterbauer, K. Moore, and D. Suciu. Why so? or why no? functional causality for explaining query answers. *MUD*, 2010.

[100] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *PVLDB*, 4(1):34–45, 2010.

[101] A. Meliou, W. Gatterbauer, and D. Suciu. Reverse data management. *Proceedings of the VLDB Endowment*, 4(12), 2011.

[102] A. Meliou and D. Suciu. Tiresias: The database oracle for how-to queries. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 337–348, New York, NY, USA, 2012. ACM.

[103] N. Meneghetti. *Exploiting Qualitative User Feedback in Deterministic and Probabilistic Databases*. PhD thesis, 2016.

[104] N. Meneghetti, O. Kennedy, and W. Gatterbauer. Beta probabilistic databases: A scalable approach to belief updating and parameter learning. Technical report, UB CSE, 2016.

[105] N. Meneghetti, O. Kennedy, and W. Gatterbauer. Beta probabilistic databases: A scalable approach to belief updating and parameter learning. In *SIGMOD*, 2017.

[106] A. Nandi. Querying without keyboards. In *CIDR 2013, Sixth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings*. www.cidrdb.org, 2013.

[107] A. Nandi. Mimir: Bringing ctables into practice, 2016.

[108] A. Nandi, L. Jiang, and M. Mandel. Gestural query specification. *PVLDB*, 7(4):289–300, 2013.

[109] A. Nandi, Y. Yang, O. Kennedy, B. Glavic, R. Fehling, Z. H. Liu, and D. Gawlick. Mimir: Bringing ctables into practice. Technical report, ArXiv, 2016.

[110] M. Nikolic, M. ElSeidy, and C. Koch. Linview: Incremental view maintenance for complex analytical queries. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 253–264, New York, NY, USA, 2014. ACM.

[111] F. Olken and D. Rotem. Simple random sampling from relational databases. In *Proceedings of the 12th International Conference on Very Large Data Bases*, VLDB '86, pages 160–169, San Francisco, CA, USA, 1986. Morgan Kaufmann Publishers Inc.

[112] F. Olken and D. Rotem. Random sampling from b+ trees. In *Proceedings of the 15th International Conference on Very Large Data Bases*, VLDB '89, pages 269–277, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[113] D. Olteanu, J. Huang, and C. Koch. Sprout: Lazy vs. eager query plans for tuple-independent probabilistic databases. In *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, pages 640–651, March 2009.

[114] D. Olteanu, J. Huang, and C. Koch. Approximate confidence computation in probabilistic databases. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 145–156, March 2010.

[115] D. Olteanu and S. J. van Shaik. Probabilistic data programming with enframe. *IEEE Data Engineering Bulletin*, 37(3):18–25, September 2014.

[116] H. Park and J. Widom. Crowdfill: A system for collecting structured data from the crowd. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 87–90, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.

[117] R. Pochampally, A. Das Sarma, X. L. Dong, A. Meliou, and D. Srivastava. Fusing data with correlations. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 433–444, New York, NY, USA, 2014. ACM.

[118] C. Qin and F. Rusu. Sampling estimators for parallel online aggregation. In G. Gottlob, G. Grasso, D. Olteanu, and C. Schallhart, editors, *Big Data*, volume 7968 of *Lecture Notes in Computer Science*, pages 204–217. Springer Berlin Heidelberg, 2013.

[119] S. Roy and D. Suciu. A formal approach to finding explanations for database queries. In *SIGMOD Conference*, pages 1579–1590. ACM, 2014.

[120] M. Saeed, C. Lieu, G. Raber, and R. Mark. Mimic ii: a massive temporal icu patient database to support research in intelligent patient monitoring. In *Computers in Cardiology, 2002*, pages 641–644, Sept 2002.

[121] A. Sampson, P. Panchekha, T. Mytkowicz, K. S. McKinley, D. Grossman, and L. Ceze. Expressing and verifying probabilistic assertions. *SIGPLAN Not.*, 49(6):112–122, June 2014.

[122] S. Singh, C. Mayfield, S. Mittal, S. Prabhakar, S. Hambrusch, and R. Shah. Orion 2.0: Native support for uncertain data. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1239–1242, New York, NY, USA, 2008. ACM.

[123] W. Spoth, B. S. Arab, E. S. Chan, D. Gawlick, A. Ghoneimy, B. Glavic, B. Hammerschmidt, O. Kennedy, S. Lee, Z. H. Liu, X. Niu, and Y. Yang. Adaptive schema databases. In *CIDR*, 2017.

[124] M. Stonebraker. Why the 'data lake' is really a 'data swamp'. http://cacm.acm.org/blogs/blog-cacm/181547-why-the-data-lake-is-really-a-data-swamp.

[125] J. Stoyanovich, B. Howe, S. Abiteboul, G. Miklau, A. Sahuguet, and G. Weikum. Fides: Towards a platform for responsible data science. In *SSDBM*, pages 26:1–26:6. ACM, 2017.

[126] B. Sundarmurthy, P. Koutris, W. Lang, J. F. Naughton, and V. Tannen. m-tables: Representing missing data. In *ICDT*, volume 68 of *LIPIcs*, pages 21:1–21:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[127] A. Thiagarajan and S. Madden. Querying continuous functions in a database system. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 791–804, New York, NY, USA, 2008. ACM.

[128] Transaction Processing Performance Council. The tpc-h benchmark. http://www.tpc.org/tpch/default.asp.

[129] T. O. L. . UB-CSE. The mimir data-ish exploration middleware. http://www.mimirdb.info.

[130] M. A. Vaz Salles, J.-P. Dittrich, S. K. Karakashian, O. R. Girard, and L. Blunschi. itrails: Pay-as-you-go information integration in dataspaces. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 663–674. VLDB Endowment, 2007.

[131] S. V. Vrbsky and J. W. Liu. APPROXIMATE - A query processor that produces monotonically improving approximate answers. *IEEE Trans. Knowl. Data Eng.*, 5(6):1056–1068, 1993.

[132] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. M. Hellerstein. Bayesstore: Managing large, uncertain data repositories with probabilistic graphical models. *Proc. VLDB Endow.*, 1(1):340–351, Aug. 2008.

[133] J. Wang and N. Tang. Towards dependable data repairing with fixing rules. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 457–468, New York, NY, USA, 2014. ACM.

[134] X. Wang, A. Meliou, and E. Wu. Qfix: Diagnosing errors through query histories. In *SIGMOD Conference*, pages 1369–1384. ACM, 2017.

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

[136] S. Wu, S. Jiang, B. C. Ooi, and K.-L. Tan. Distributed online aggregations. *Proc. VLDB Endow.*, 2(1):443–454, Aug. 2009.

[137] F. Xu, K. Beyer, V. Ercegovac, P. J. Haas, and E. J. Shekita. E = mc3: Managing uncertain enterprise data in a cluster-computing environment. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 441–454, New York, NY, USA, 2009. ACM.

[138] Y. Yang. On-demand query result cleaning. In *VLDB PhD Workshop*, 2014.

[139] Y. Yang. *Interactive Data Management and Data Analysis*. PhD thesis, 2017.

[140] Y. Yang and O. Kennedy. Convergent inference with leaky joins. In *EDBT*, 2017.

[141] Y. Yang, N. Meneghetti, R. Fehling, Z. H. Liu, and O. Kennedy. Lenses: An on-demand approach to etl. *Proc. VLDB Endow.*, 8(12), 2015.

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