Operators for Analyzing and Modifying Probabilistic Data – A Question of Efficiency

Jochen Adamek, Katrin Eisenreich, Volker Markl, Philipp Rösch

j.adamek@tu-berlin.de, katrin.eisenreich@sap.com,
volker.markl@tu-berlin.de, philipp.roesch@sap.com

Abstract: To enable analyses and decision support over historic, forecast, and estimated data, efficient querying and modification of probabilistic data is an important aspect. In earlier work, we proposed a data model and operators for the analysis and the modification of uncertain data in support of what-if scenario analysis. Naturally, and as discussed broadly in previous research, the representation of uncertain data introduces additional complexity to queries over such data. When targeting the interactive creation and evaluation of scenarios, we must be aware of the run-time performance of the provided functionalities in order to better estimate response times and reveal potentials for optimizations to users. The present paper builds on our previous work, addressing both a comprehensive evaluation of the complexity of selected operators as well as an experimental validation. Specifically, we investigate effects of varying operator parameterizations and the underlying data characteristics. We provide examples in the context of a simple analysis process and discuss our findings and possible optimizations.

1 Introduction

In the decision making process, we need to consider risks and chances of future developments. To this end, the derivation and evaluation of scenarios based on different assumptions about the future is a powerful technique. However, as per definition, applying an assumption always introduces uncertainty to the data at hand. This uncertainty must be appropriately represented in the data. Existing uncertainty management approaches mostly address applications in the fields of scientific and sensor data processing, spatial databases, information extraction, or data cleansing. Decision support over large volumes of data including both uncertain data and certain information, e.g., from a data warehouse, has received comparatively little attention so far. A prominent exception is the work presented in [JXW+08], which relies completely on a sample-first approach. In previous work (see [ERM+10]), we in contrast apply a model-extension approach to represent, analyze, and modify uncertain data. Our primary goal is to support users in the flexible creation and evaluation of what-if scenarios over (partially) uncertain data represented through arbitrary distributions. We consider the process of what-if analysis as an iteration of steps of data analysis and scenario creation as described in [ER10]. Apart from its iterative nature, we also point out that we aim to enable users to conduct the analysis process in a highly interactive fashion. In the best case, a user should be able to derive a scenario, analyze it,

change some of its underlying data, and analyze the resulting alternative scenario within seconds. Naturally, the additional complexity implied by the representation of uncertainty poses a major challenge when aiming at low response times. A comprehensive analysis of run-times is therefore a major contribution when assessing the overall performance and general applicability of our approach. Moreover, it can serve us to discover room for improvements.

To exemplify the application scope of our solution, consider the following use cases:

- Use case UC 1: An analyst wants to prospect next year's revenue in a newly developed region R_{new}. He takes the past development of a similar region R_{ref} as reference for his prediction. Additionally, he wants to take into account available forecasts about the general economic development.
- Use case UC 2: A user analyzes the process of delivery and deployment of orders. He wants to investigate possible resource costs caused by deployment personnel during a specific time frame. To this end, he applies different assumptions regarding temporally uncertain delivery times and deployment durations.

As noted above, most of the existing approaches for uncertain data management focus on efficient querying and analyses of (mainly discrete) probabilistic information. The aspect of modifying data to create scenarios, which is a central aspect of our work, is mostly out of their scope. In the remainder of the paper, we will therefore foremost discuss those operators, yet emphasize that the 'traditional' aspect of data analysis is also an integral part of our approach. We briefly describe important aspects of our data model and an operator set for deriving and converting uncertain values, as well as analysis and modification of such values in Section 2. We then evaluate the complexity of the selected set of operators (Section 3) and provide an experimental validation of their costs based on a prototypical implementation (Section 4). We address opportunities for optimizing the computation of steps in an analysis process in Section 5. We present related work in Section 6 and summarize our findings in Section 7.

2 Data Model and Functionalities

What-if analyses and decision support over uncertain data require a flexible data model and powerful operators which both are introduced in our previous work, see [ER10, ERM+10]. Our data model allows for the use of both symbolic and histogram-based representations of uncertainty, similar to [SMM+08]. An uncertain value x_i[1] is associated with a distribution P_i which can be represented symbolically or as a histogram $\bar{P}_i$. A histogram comprises β_i bins $B_i = \{b_1, \ldots, b_{\beta_i}\}$ within a lower support (boundary) l_i and an upper support (boundary) h_i. Each bin $b_j \in B_i$ is associated with a density w_j. Similarly, we use two-dimensional histograms to represent two-dimensional distributions $P_{x,y}$. The discussion in this work is focused on the usage of equi-width histograms (EWH). Alternative

[1] Where possible, we omit the subscript for reasons of readability.

partitioning schemes are conceivable but imply higher costs and are out of the scope of the current work.

By investigating typical questions for what-if analyses and processes implementing such analyses (see [ER10]), we identified a set of operations we deem specifically important in this context as discussed in the following.

Introducing and converting uncertain information First, at the start of an analysis process, no specific knowledge about a value's distribution is given in many cases. Users therefore may want to derive information about an (expected) distribution from historic fact data. For example, in UC 1 the user assumes next year's revenue development in region R_{new} to follow the distribution observed in a similar region R_{ref}. To introduce this information, he derives and stores a histogram over last year's recorded revenue values for cities in R_{ref} by using our operator DRV. Another basic operation (CNV) serves to change the representation of such derived or externally provided uncertain values. This step can be applied explicitly, e.g., when users want to convert a symbolic into a histogram-based representation as basis for flexible modifications. Moreover, it can be adopted implicitly, e.g., when an input is given in symbolic form but the executed operator requires a histogram-based representation. The derivation and conversion of the predicted (relative) economic growth inc_e and the (relative) regional revenue increase inc_{rev} are schematically depicted in the left portion of Figure 1 and further discussed in Sections 3.1 and 3.2.

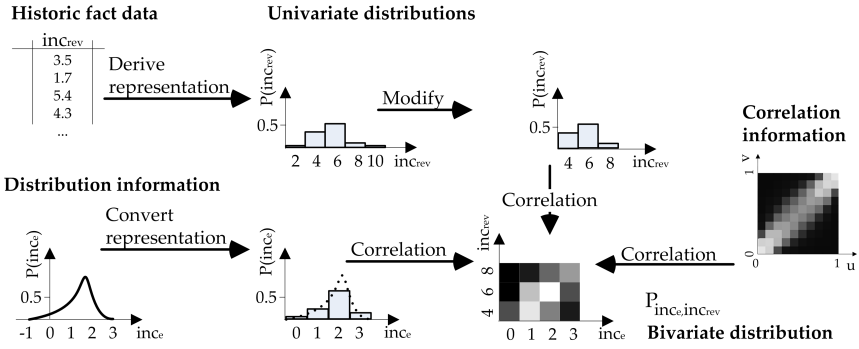


Figure 1: Overview of exemplary representation and processing of distributions

Evaluating the uncertain information Once (uncertain) data is available in appropriate representations, users need analysis capabilities such as computing aggregates, or selecting and viewing values based on a provided threshold. While such data analysis functionality is a natural part of our approach (contributing to the analysis part of the what-if analysis process), it is not the focus of this paper. Rather, we mainly concentrate on (i) the modification of uncertain data, (ii) the handling of dependency in data, and (iii) the issue of temporal indeterminacy, all of which are particularly estimable aspects in the context of scenario-based planning:

Modifying uncertain data: The modification of uncertain data is essential to support the application of assumptions about the development of selected aspects to derive scenarios representing potential future states of the world. For example, considering the inc_{rev} value in UC 1, a user might want to discard "outlier" information or apply the assumption that extremely low revenues will be avoided due to appropriate marketing measures. This is reflected by modifying (MOD) part of the distribution of inc_{rev}, as depicted in the middle portion of Figure 1. In other cases, users want to update an old value altogether to assume a new distribution for a future value. In both cases, we can view the applied modification as a creation of a new scenario. To retain the relation between values in different scenarios, we do not replace the original value but store the modification result together with a reference to the original value. When modifying histograms, we use a delta approach, that is, we keep the (bin-wise) delta of the new value to the old value. Besides enabling users to investigate the lineage of values, the delta approach enables more efficient incorporation of modification results in further processing steps (see Sections 3.3 and 5).

Introducing dependencies: Unlike many other approaches for probabilistic data management, we explicitly target the representation and processing of arbitrary correlation structures between uncertain values. Specifically, we address the case where a user wants to *introduce* an assumed correlation between two values when no dependency information can be faithfully derived. This can be the case, e.g., when underlying fact data is too sparse or when two values are provided as independent data. The latter occurs in UC 1, where we dispose of separate values for the forecast economic development (inc_e) and the derived revenue increase inc_{rev}. To model arbitrary forms of correlation (e.g., linear correlation or high dependencies of extreme values) we apply the approach of copula functions as described in [ER10]. Rather than constructing copulas at run-time (implying calls to statistical functions and corresponding costs) we precompute, store, and process them in the form of a histogram-based approximative correlation representation (ACR). The introduction of correlation is depicted in the right-hand portion of Figure 1, where two univariate distributions ($P_{inc_{rev}}$ and P_{inc_e}) are combined into a joint distribution based on correlation information from an ACR. Details of the ACR approach are discussed further in Section 3.4.

Temporal uncertainty in planning processes: Finally, in addition to representing and processing uncertainty of (measure) values, we also enable the representation of temporal indeterminacy, i.e., uncertainty in the temporal allocation of data [ERM+10]. For example, use case UC 2 rises the necessity to compute costs for the deployment of ordered products following their uncertain delivery times and assuming an uncertain duration of deployment. In this context, we consider the handling of indeterminate "events" occurring during some uncertain time interval. In Section 3.5 we consider the aggregation ($AGG^{\underline{T}}$) over measures associated with such events within a time interval $\underline{T}$.

For a more detailed description of the set of operators and their application in the decision support context, we refer to [ER10, ERM+10].

3 Analytical Evaluation

In this section, we analyze the complexity of the operators lined out above under different parameterizations and varying input data characteristics. In general, except for the case of converting representations, we assume input values to be represented in the histogram-based form; i.e., we do not consider the application of our operators for modification, correlation introduction, and temporal aggregation on symbolic representations. In case a distribution is given in symbolic form, we can use the conversion operator to yield the respective histogram-based representation. Consequently, the number of bins β used to represent univariate distributions is the most relevant factor as regards the complexity of the evaluated operators. Further, for the derivation of distributions, the number of underlying facts (n_F) is crucial. Similarly, where sampling is applied, the number of sample elements (n_S) naturally influences not only the "accuracy" of results but also the implied costs for their derivation. When processing or constructing symbolic representations of distributions, costs for computing specific statistics, such as quantiles, can vary. We do not consider such distribution- and implementation-dependent costs in this section, but rather examine them experimentally in Section 4.

3.1 Derivation of Representations

We support the derivation of distributions over values of fact data, which users can then use as a basic input for their analyses. The intuition is that such a distribution may constitute a proper reference for the development of another value in some similar context. For example, in our use case UC 1, the analyst wants to utilize knowledge about the past revenue increases in region R_{ref} as reference for the prospective revenues in a newly developed region R_{new} with no historic data available. To this end, he needs to construct a distribution from the historic revenues of all stores in R_{ref}.

The $DRV(F, tgt)$ operator serves exactly this purpose, essentially enabling the *introduction* of uncertainty based on fact data. It receives a number n_F of facts from a column in the fact table F of our database. The target distribution P is specified via the tgt parameter, including the representation type and further parameters determining P. In particular, the representation form can be either histogram-based or symbolic. In the former case, a user must further provide the number of bins β and the lower and upper support (l, h) of the desired histogram $\overline{P}$. In the latter case, a user must provide the assumed function of the distribution based on some insight or expectation about the underlying facts.

Histogram Representation When deriving a histogram over the fact values, we assume they are provided in a non-sorted order. In the case of equi-width histograms, which we focus in this paper, we statically compute bin boundaries based on the desired lower and upper support (l, h) and the number of bins β and assign each of the n_F values, resulting in complexity $O(n_F)$. In the general case, for each of the n_F values underlying the histogram to-be derived, we apply a bisection algorithm for sorting it into one of the β bins of the

target histogram, resulting in a complexity of $O(n_F \cdot log_2(\beta))$.

Symbolic Representation In our prototype, we support the derivation of uniform, Gaussian, and Gamma distributions, while further functions could be added. Finding the lower and upper bounds of the distribution support of an assumed uniform distribution requires one scan of the underlying facts to determine their maximum and minimum. For a Gaussian, we similarly need to process each of the n_F values to iteratively compute the mean and variance. We currently estimate the scale and shape parameters of a Gamma distribution from those parameters. Hence, for all considered distribution functions, the computation implies a complexity of $O(n_F)$. The specific costs for a given target function naturally depend on the individual calculations conducted over each fact value.

3.2 Conversion of Representations

The operator $CNV(x, tgt)$ allows users to flexibly change the way of managing the uncertain data, converting the symbolic distribution representation of a value x to a histogram and vice versa, depending on the parameters specifying the target distribution tgt. Further, users can change the resolution of a histogram, e.g., for statistical error analysis, by setting a new number of bins β. Returning to UC 1, a user wants to incorporate information about the economic forecast for the region R_{new}. This forecast is provided by means of an expected value and an associated confidence interval and is represented in the system as a Gaussian with appropriate mean and variance. In order to further process this value, the user converts it to a histogram-based form. Another application of CNV arises when users want to test a (derived) distribution for goodness of fit with actual data; such tests (i.e., the χ^2-test) often rely on binned data.

Similar to the DRV operator, users must provide parameters specifying the desired distribution tgt, including the type of representation and representation-specific parameters. The three potential cases of conversion are as follows:

Symbolic Distribution into Histogram For constructing a histogram from a given distribution function, such as a Gaussian, the user defines the lower and upper support (l,h) of the target histogram and the desired number of bins β or, alternatively, an optimal β can be estimated using a basic heuristic aiming at an optimal approximation of the interval with β bins (e.g., Sturges rule). For each of the β bins, the source distribution is integrated within the lower and upper bin boundary, implying a complexity of $O(\beta)$. In the case of a uniform source distribution, density values for β bins based on l and h equally results in $O(\beta)$.

Histogram into Symbolic Distribution To compute parameters of an assumed uniform distribution from a histogram we need to find its lower and upper bounds by reading β bins or, if available, exploit stored metadata about l and h, inducing a complexity of $O(\beta)$ or a constant access cost $O(1)$, respectively. To estimate parameters of a Gaussian or Gamma

distribution, we compute the required parameters (mean and variance or scale and shape, respectively) through an iterative run over all bins, inducing a complexity of $O(\beta)$.

Histogram into Histogram As a third alternative, we can convert a source histogram $\bar{P}_x$ with β_x bins into a new histogram $\bar{P}_y$ with β_y bins. This conversion serves, e.g., to provide users with a changed granularity of information or to ensure two histograms have the same bin resolution. This might be required for a succeeding operation such as testing the fit of two distributions to each other. The conversion proceeds by finding the bin boundaries of the β_y bins of $\bar{P}_y$ and computing the area covered by the β_x bins in $\bar{P}_x$ within the boundaries of each bin of $\bar{P}_y$. The imposed complexity of the computation is $O(\beta_x + \beta_y)$.

3.3 Modification of Uncertainty

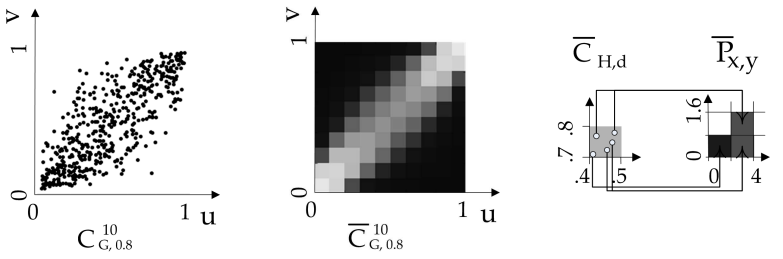
Modification of an uncertain value is necessary to introduce a new assumption about its concrete distribution in a potential scenario or adapt its value otherwise. For example, in UC 1, the derived distribution for the (relative) regional revenue increase inc_{rev} might include large tails due to outlier data. If the user assumes those tails of the distribution irrelevant for his current analysis (or does not want to consider this part of the distribution in his scenario), he can modify $\overline{P}_{inc_{rev}}$ by setting the densities associated with relevant bins of $\overline{P}_{inc_{rev}}$ to 0, as illustrated in Figure 1. In UC 2, the analyst can modify expected deployment start times of selected orders to analyze the potential influence on the resulting deployment costs within the time slot under consideration.

The operator $MOD(x_{old}, x_{new}, cond)$ is applied to histograms $\overline{P}_{x_{old}}$ and $\overline{P}_{x_{new}}$ to change the represented distributions. A modification causes the frequencies associated with selected bins of $\overline{P}_{x_{old}}$ to be changed, optionally depending on a specified condition $cond$. This way, a user can explicitly specify both the affected bins and their target density through x_{new}; alternatively, he can provide a condition for determining the bins whose density shall be changed as well as their new density value. An example is the application of a predicate to modify a certain part (e.g., the tail) of a distribution by specifying a condition on the (new) lower and upper support of x_{new}. To ensure that modified values can be traced back to the original value, we do not replace x_{old} but insert a new value x_{new} with a reference to x_{old}. This way, we can further apply and compare multiple modifications. Physically, the modification is written back as the delta $\overline{P}_{\Delta}$ (bin-wise difference) from $\overline{P}_{x_{old}}$ to $\overline{P}_{x_{new}}$. A worst case complexity of $O(\beta)$ is induced by reading β bins and writing delta density values for all bins. The influence on actual costs depends on the resulting degree of modification (f_m), i.e., the fraction to which a distribution is actually affected by a conditional modification.

3.4 Introduction of Correlation

The possibility to introduce correlation to previously independent (or independently represented) values is a valuable means to investigate effects of dependencies between values in data, e.g., to analyze the probability of extreme values occurring jointly. In UC 1, we assume that the user wants to evaluate a correlation among values that were provided separately; they are represented by the revenue increase distribution $P_{inc_{rev}}$ and the forecast economic growth P_{inc_e}.

We use the operator $COR(x, y, H, d)$ to enable the introduction of a correlation structure determined by H and d between two distributions P_x and P_y. As noted before, the processing of COR is based on the usage of copulas. In brief, a copula C is a distribution function representing the relation between two marginals F and G and their joint distribution H. The formal foundation is Sklar's Theorem [Skl59, MST07], which states that, given H as a bivariate[2] distribution with F and G as univariate marginal distributions, there exists a (copula) function $C : [0, 1]^2 \rightarrow [0, 1]$ so that $H(x, y) = C(F(x), G(y))$. Using the inversion approach (see, e.g., [Nel06]), we can construct a copula $C(u, v) = H(F^{-1}(u), G^{-1}(v))$, u and v being uniforms over $[0, 1]$. We write $C_{H,d}$ to denote a copula where the structure of the represented correlation is determined by the distribution H and the correlation degree d. To correlate two arbitrary distributions P_x and P_y, we then again apply Sklar's Theorem, substituting F and G with the desired marginals P_x and P_y. We investigate both the complexity of the native (sampling-based) approach and our approach based on approximate correlation representations (ACRs).



(a) $n_S = 500$ samples of a copula $C_{G,0.8}$

(b) ACR $\bar{C}_{G,0.8}$ with $\alpha^2 = 10^2$ bins

(c) $n_{S_{ACR}} = \sum w_{i,j}$ samples are drawn from α^2 bins $b_{i,j}$

Figure 2: Factors influencing the performance of correlation introduction

Sampling-based copula approach For the sampling-based approach, we must consider the costs for both constructing and applying the copula. We first draw n_S samples of its underlying bivariate distribution H with correlation d. We transform the samples using the cumulative distribution function of each of F and G to construct the copula as distribution over $[0, 1]^2$. See Figure 2(a) for an example of a Gaussian copula $C_{G,0.8}$ with 500 samples.

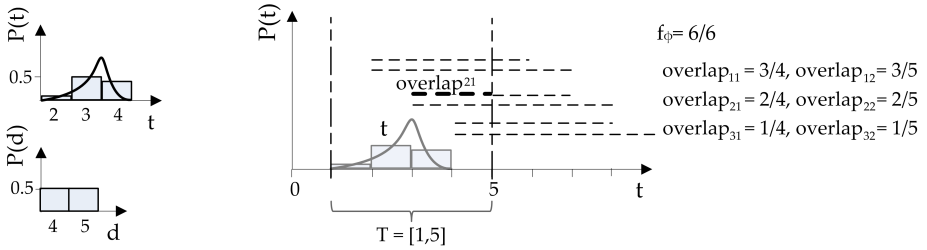
[2]Without loss of generality, we restrict our considerations to the bivariate case, i.e., only consider the correlation amongst two variables.

The copula construction requires $3 \cdot n_S$ computations applying calls to statistical functions, inducing a complexity of $O(n_S)$. Applying the constructed copula implies, for the two coordinates of each of the n_S samples, a computation of the quantiles of the distributions P_x and P_y (provided as histograms $\bar{P}_x$ and $\bar{P}_y$). The computation of quantiles requires a binary search over the frequency values of the β_x, β_y bins of $\bar{P}_x$ and $\bar{P}_y$ succeeded by computing the concrete distribution within the bin based on a uniform spread assumption. In total, the sampling-based approach – including the construction and application of the copula – results in a complexity of $O(n_S + n_S log_2(\beta_x) + n_S log_2(\beta_y))$.

ACR-based approach Aiming at lower response times and an independence from statistical library calls at run-time, we pre-compute copulas $C_{H,d}$ and store them as ACRs with α^2 bins, denoted by $\overline{C}^{\alpha}_{H,d}$ (see Figure 2(b)). Then, to correlate two values, the system chooses and applies an appropriate ACR based on the desired correlation parameters. This way, no costs for copula construction are induced at run-time. Rather, we use the aggregated information stored in the ACR, i.e., the coordinates of the α^2 bins $b_{i,j}$ and their respective weights (densities) $w_{i,j}$. As discussed in [ER10], we decrease the negative influence of discretization by applying inversion based on artificial samples from each bin $b_{i,j}$. This is indicated for an individual bin in Figure 2(c). Those samples are uniformly distributed within each $b_{i,j}$, the sample number per bin being relative to its weight $w_{i,j}$. The correlation introduction then proceeds exactly as for the sampling-based approach. Using a total of $n_{S_{ACR}}$ samples over all bins, this implies a complexity of $O(n_{S_{ACR}} log_2(\beta_x) + n_{S_{ACR}} log_2(\beta_y))$ for computing quantiles of P_x and P_y for each sample. In the usual case, we consider $n_{S_{ACR}}$ to be similar to or higher than n_S to ensure comparable result accuracy. A very basic approach is to invert the coordinates of the α^2 bin *centers* only (instead of inverting $n_{S_{ACR}}$ sample coordinates), resulting in a complexity of $O(\alpha^2 log_2(\beta_x) + \alpha^2 log_2(\beta_y))$. In a usual case, α^2 is a magnitude smaller than $n_{S_{ACR}}$, resulting in, e.g., $\alpha^2 = 40^2 = 1600$ rather than $n_{S_{ACR}} = 40000$ quantile computations for P_x and P_y. However, those savings come at the cost of mostly unacceptable discretization errors.

3.5 Indeterminate Temporal Aggregation

In order to enable the handling of temporal indeterminacy of plans, we consider the analysis over indeterminate events. The indeterminacy of an event e_i is reflected through an uncertain start time t_i and a duration d_i. As an example, to implement UC 2, we need to compute the prospective overall deployment costs implied by a number of indeterminate deployments $e_i \in E_{dep}$ during a specified interval, e.g., $\underline{T} = [1, 5]$. We use the operator $AGG^{\underline{T}}(X, E, \underline{T})$ to compute the aggregate (sum, minimum, or maximum) of values of an attribute $X = \{x_0, \ldots, x_n\}$ associated with temporally indeterminate events $E = \{e_1, \ldots, e_n\}$ within a time interval $\underline{T} = \left[l_{\underline{T}}, h_{\underline{T}}\right]$. To compute the aggregate, we must consider all events that have a potential overlap with $\underline{T}$ (i.e., $l_{t_i} < h_{\underline{T}} \wedge h_{t_i} + h_{d_i} > l_{\underline{T}}$). In the following, we consider the aggregation over a single event $e \in E$ (omitting the subscript for reasons of readability). Essentially, we need to



(a) Start t and duration d of event e

(b) Aggregation over indeterminate events illustrated over the single event e

Figure 3: Representation and processing of temporal indeterminacy associated with indeterminate events

compute the fraction ϕ to yield the contribution $\phi \cdot x$ of the measure value x associated with e. The fraction ϕ depends both on the position of $\underline{T}$ and on the set of all possible intervals $I_{pq} \in \mathcal{I} = \left\{ \left[t_p^{start} = v_p, t_{pq}^{end} = v_p + v_q \right] \right\}, v_p \in I_t, v_q \in I_d$ in which e can occur. To give an example, Figure 3(a) depicts the start time t, $\beta_t = 3$ and duration d, $\beta_d = 2$ of the event e while Figure 3(b) shows how aggregation works over this single event. In Figure 3(b), the fraction ϕ of e results from six possible occurrence intervals $\mathcal{I} = \{I_{11}, I_{12}, \ldots, I_{32}\}$. For each possible interval, we need to compute joint probabilities $P(t = v_p) \cdot P(d = v_q)$ (assuming independence between t and d) and the fractions of overlaps, i.e., the part of I_{pq} that lies within $\underline{T} = [1, 5]$.

We denote the average number of potential occurrence intervals of all considered events $e_i \in E$ as $n_I = (\sum_{i=0\ldots N} \beta_{t_i} \cdot \beta_{d_i})/|E|$ and the fraction of those intervals that actually overlap $\underline{T}$ (and therefore contribute to the aggregate result) as f_ϕ . Temporal aggregation implies $|E| \cdot n_I \cdot f_\phi$ complete computations of overlaps and joint probabilities. The worst case therefore is of complexity $O(n_F \cdot n_I)$.

4 Experimental Evaluation

In this section, we report on experiments evaluating the discussed operators. The goal of those experiments is the validation of our analytical results for the selected operators. Further, based on concrete results, we can quantify the costs for reading and writing data and the specific computations involved in operator processing steps.

4.1 Implementation and Setup

We extended an existing proprietary engine that computes complex analytical queries by means of so-called *Calculation Views* (CV). Those views enable OLAP analysis functionality as well as applying custom operations provided as Python or C++ implementations.

For a more detailed explanation of the underlying architecture and the concept of CVs, see [JLF10]. Further, for the computation of statistic functions (e.g., for copula construction at runtime) we rely on the statistic library IMSL[3]. Physically, the data are stored and accessed per column. Since our operators operate only on one or two columns at most of the time, this setting is beneficial for many operations.

Experimental Setup For the following experiments, we used a dual CPU workstation with 4GB of main memory running Windows Vista 64bit. The goal was to investigate the required run-times for selected operators. To validate the influence of factors identified in Section 3 above on actual run-times, we varied both the parameterization of operators and the scaling factors of the underlying TPC-H[4] data as well as the characteristics of uncertain input data (represented mostly by histograms). Note that, in the general case, intermediate results derived by an operator are not persisted unless stated otherwise.

4.2 Experiments

Loading Histogram Data As a basis for most of the other operators, histogram data needs to be loaded from the database into our internal histogram structure. Figure 4 shows the costs induced by loading histograms with varying numbers β of bins. Note that for the current prototype, those costs are far from optimal due to the fact that we access the internal tables storing our histogram data via SQL statements rather than internal table searches. Clearly, load times increase linearly with the number of fetched histograms. However, repeated access to individual values or small sets of values causes relatively higher costs.

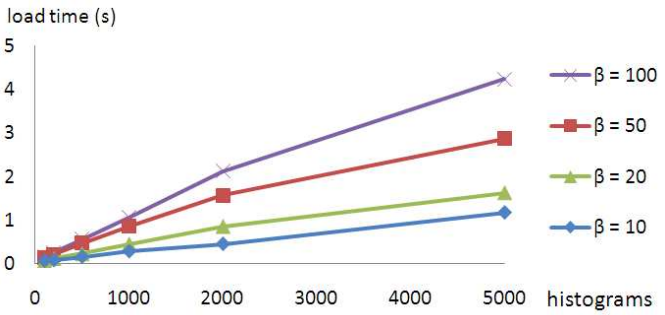


Figure 4: Times for loading histograms for a number of uncertain values

Derivation Subsequently, we use the `lineitem` table from the TPC-H benchmark as a basis for deriving distributions. We derive both histogram-based representations with vary-

[3]http://www.vni.com/products/imsl/
[4]http://tpc.org

ing numbers of bins and symbolic representations over the values of the `extendedprice` attribute. We assume that the considered data follows either one of a uniform, a Gaussian, or a Gamma distribution. Using scaling factors $s = 0.1$ and $s = 1.0$ results in $n_F = 600k$ and $n_F = 6M$ attribute values, respectively. Depending on an optional grouping, we derive a *total* distribution over all values, or $20k$ and $200k$ distributions for each `lineitem.partkey`, respectively. As shown in Table 1, run-times increase almost linearly with the size of n_F when we derive one distribution from all fact values. In the grouped derivation, we observe a slightly stronger increase in the case of histogram derivations, which we attribute to the high memory allocation costs. This factor also causes slightly rising run-times as β increases, even though the bin allocation as such is constant at $O(1)$. For the derivation of symbolic representations, run-times increase perfectly linear with n_F.

Table 1: Run-times (sec) for derivation of different distribution representations over a total of $n_F = 600k$ values ($s = 0.1$) and $n_F = 6M$ values ($s = 1.0$) of attribute `lineitem.extendedprice`.

Scale	Distribution Representation					
	(Equi-width) Histogram			Symbolic		
	$\beta = 10$	$\beta = 20$	$\beta = 100$	Uniform	Gaussian	Gamma
$s = 0.1, total$		0.165		0.156	0.28	0.28
$s = 0.1, grouped$	0.57	0.60	0.61	0.65	0.92	0.92
$s = 1.0, total$		1.62		1.55	2.82	2.82
$s = 1.0, grouped$	6.55	6.87	6.96	6.50	8.95	8.95

Table 2: Run-times (sec) for converting 1000 values from symbolic to histogram representations.

Source	Target Histogram Representation β			
	10	20	50	100
Uniform	0.136	0.19	0.3	0.556
Gaussian	0.115	0.173	0.338	0.619
Gamma	0.136	0.176	0.364	0.692

Table 3: Run-times (sec) for converting 1000 histogram-based representations to symbolic representations.

Source	Target Distribution		
	Uniform	Gaussian	Gamma
$\beta = 10$	0.025	0.06	0.062
$\beta = 50$	0.028	0.067	0.068
$\beta = 100$	0.035	0.080	0.082

Conversion Tables 2 and 3 show results for converting symbolic into histogram representations with varying β and for converting histograms into assumed symbolic representations, respectively. We can see in Table 2 that the observed costs are relatively higher for low values of β, which results from setup and loading costs. Beyond this initial cost, run-times increase almost linear with β, due to the fact that we must compute discrete density values by integration within each of the β bins as discussed in Section 3.2. Note that the concrete cost for potential further distribution functions will vary depending on the concrete implementation (e.g., through a call to an external library) of their integration. Table 3 shows run-times for deriving function parameters of assumed distribution functions from source histograms. Computation costs increase only slightly with the size

of β. The total costs are strongly dominated by loading times, increasing linearly with β as already shown in Figure 4.

Modification We evaluated both the modification of values based on an update value x_{new} provided a priori and applying value-based conditions. Different from the other operators, in the case of MOD, we write back the bin-wise delta from the old to the new value. Figure 5 shows the results of modifying histograms with varying β based on a threshold on the distribution support; that is, all bins with right bounds below a threshold were modified (e.g., set to 0). The threshold was varied so that the modified fraction f_m increased from 0.0 to 1.0. One can see that run-times increase linearly with f_m as only modified bins are written back. The increase of run-times is also linear in β. We observe a stronger increase for $\beta = 100$, which is due to current implementational restrictions of our prototype.

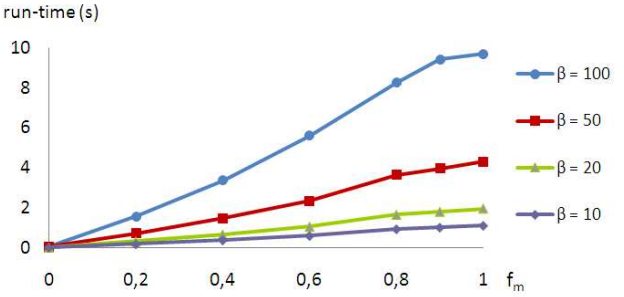


Figure 5: Run-times for modifying 100 histograms, applying (value-based or frequency-based) conditions affecting various fractions of the histograms

Correlation Introduction To evaluate run-times of the COR operator, we varied the characteristics of the copulas underlying the sampling- and ACR-based approaches as well as the distributions to be correlated. First, we evaluated the application of different copulas $C_{H,d}$ built from a bivariate Gaussian and T-student distribution H with correlation degrees $d = 0.4$ and $d = 0.8$, respectively. We further varied the number of samples (n_S) per copula $C_{H,d}$ and the number of bins (α) per ACR $\overline{C}^{\alpha}_{H,d}$. Both $C_{H,d}$ and $\overline{C}^{\alpha}_{H,d}$ were applied to two histograms $\overline{P}^{20}_{x}$ and $\overline{P}^{20}_{y}$ to yield a result histogram $\overline{P}^{20,20}_{x,y}$. Table 4 displays the run-times required for computing $\overline{P}^{20,20}_{x,y}$, averaged over 100 runs each. For the sampling-based approach, we must include the times for copula construction and for deriving $\overline{P}^{20,20}_{x,y}$. In contrast, for the ACR-based approach, we exclude copula construction times since we only need to access the precomputed ACR histograms and compute the quantiles for the $n_{S_{ACR}}$ artificially derived samples. In this case, we fix $n_{S_{ACR}} = 100k$. Table 4 shows that the run-times of the ACR-based cases are almost constant at about the time required for processing the respective copulas $C_{T(1),d}$ and $C_{G,d}$ using $n_S = 5k$ and $n_S = 10k$ samples, respectively. The constant behavior is due to the fact that we keep the number of

$n_{S_{ACR}}$ constant for all applied ACRs, irrespective of the number of α. The displayed run-times for the sampling-based approach increase linearly with the used number of samples n_S. As an indication – although the issue of accuracy is not further discussed in this paper – the result accuracies reached by using copulas of $20k$ and $40k$ samples are comparable to those achieved when using the respective ACR with $\alpha = 40$ bins.

Table 4: Run-times (in ms) for deriving $\overline{P}_{x,y}^{20,20}$ through sampling approach and ACR processing

Copula	n_S			ACR-based (α)		
	$5k$	$10k$	$40k$	10	20	40
$C_{Gauss,0.4}$, $C_{Gauss,0.8}$	123	157	361	150	150	150
$C_{T(1),0.4}$, $C_{T(1),0.8}$	140	189	474	150	150	150

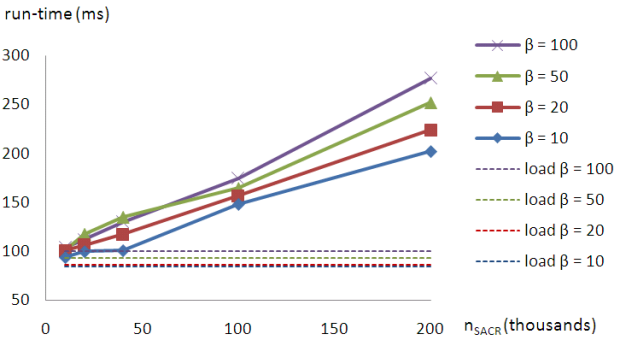
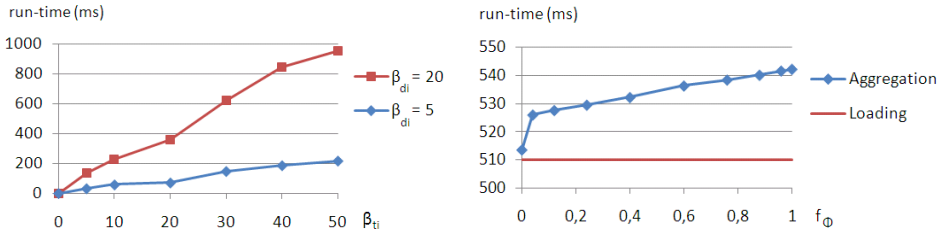


Figure 6: Run-times for ACR-based correlation introduction with varying parameters

We further varied the number of β_x and β_y used to represent P_x and P_y as well as the number of $n_{S_{ACR}}$. The resulting run-times are shown in Figure 6. We can see that there is an initial cost for setting up the operator and loading the data, which clearly dominates run-times especially for small β and $n_{S_{ACR}}$. Beyond this initial cost factor, we see an increase linear in $n_{S_{ACR}}$ due to the cost for each additional sample inversion (quantile computation). With increasing numbers of β_x and β_y (simultaneously set to 10, 20, 50, or 100 bins, respectively) we can see increasing run-times slightly below the assumed logarithmic increase due to the increased cost of each quantile computation, as discussed in Section 3.4.

Temporal Aggregation The efficiency of $AGG^{\underline{T}}$ is subject to many variations as described above. We now evaluate the influence of β_{t_i} and β_{d_i}, as well as the fraction f_ϕ of the potential occurrence intervals I_{ipq} overlapping $\underline{T}$. We apply $SUM^{[10,15]}$ for 1000 artificial events $e_i \in E$. Start times t_i and durations d_i are uniformly distributed over $[0, 5]$, each represented by a corresponding histogram with $\beta_{t_i} = \beta_{d_i} = 5$. The results are displayed in Figures 7(a) and 7(b).

We investigate the variation of β by aggregating over a number of 1000 events associated with varying β_{t_i} and β_{d_i}, respectively. The portion f_ϕ of overlapping occurrence intervals



(a) Varying β_{t_i} from 0 to 50

(b) Varying fraction of overlaps f_ϕ from 0.0 to 1.0

Figure 7: Temporal aggregation over $1k$ events with varying start time and duration characteristics

is kept stable (at 100%) by ensuring that, for every variation, $h_{t_i} < l_{\underline{T}} \wedge l_{t_i} + l_{d_i} > l_{\underline{T}}$. The results are shown in Figure 7(a). Results for varied β_{d_i} are similar given the portion of potential overlaps is similarly kept stable at 100%. To vary the portion f_ϕ of overlapping I_{ipq} from 0.0 to 1.0, we keep t_i and $\underline{T}$ constant and calculate $AGG^{\underline{T}}$ for $l_{d_i} = 0, \ldots, 10$ and $h_{d_i} = l_{d_i} + 5$. The resulting run-times are shown in Figure 7(b). In both experiments, the observed behavior is in line with the results of Section 3.5, reflecting a linear rising in run-times for increased β_{t_i}, β_{d_i}, and f_ϕ, respectively. An exception occurs for $f_\phi = 1/25$, where we observe an initially stronger increase. This is due to the fact that for $f_\phi = 0.0$ all events lie outside $\underline{T}$ and do not induce any costs for testing of overlapping intervals, while for any $f_\phi > 0.0$ those tests imply an initial cost.

5 Efficiency and Optimization of the Analysis Process

So far, we discussed the efficiency of individual operators with respect to their specific characteristics. We now address selected issues of optimization in the face of large amounts of input data and the application of composed operator sequences in order to enable lower response times for their interactive and iterative application in analysis processes.

An example process In Figure 8 we illustrate the subsequent application of selected operators implementing UC 2. Recall that we want to analyze costs associated with a set of indeterminate deployments E_{dep} during a specified time interval, where the deployment of a part follows its uncertain delivery. Consider as the basis for our analysis a data warehouse storing information about line items and associated orders as represented in the `lineitem` and `order` tables. We want to prospect the probable deployment costs for a group of line items during the next weeks. We assume that their times to delivery (`ttd`) (computed from the `order.orderdate` and `lineitem.receiptdate` attributes) will behave similar to the distribution of delivery times observed in the historic data. To reflect this assumption, for each `lineitem.partkey`, we derive a histogram (EWH) $\bar{P}_{t_i}$ over `ttd` for all delivered items. We view P_{t_i} as the distribution of the start time of a prospective deployment event e_i for an item ordered today (viewing "today" as day 0). For

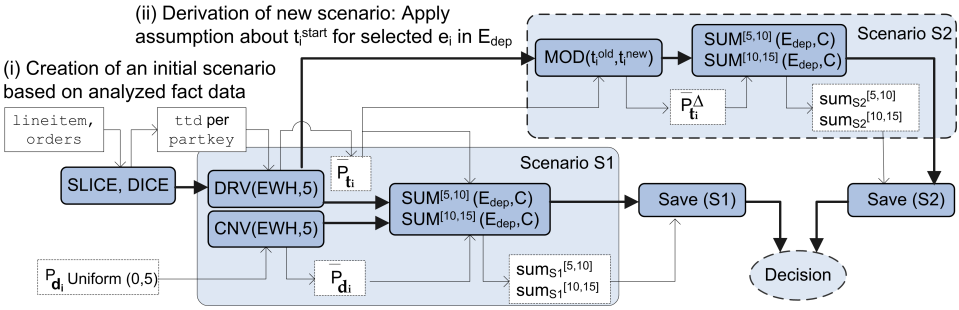


Figure 8: Process illustrating UC 2 and the incremental derivation of scenarios

simplicity, we assume a constant cost $c_i = 100$ and a duration d_i following a uniform distribution in $[0, 5]$ (converted to $\bar{P}_{d_i}$) for all deployment events $e_i \in E_{dep}$. The aim of our analysis is (i) to compute prospective deployment costs induced by selected orders during time intervals $[5, 10]$ and $[10, 15]$ and (ii) to investigate, in the case of an unfavorable cost situation, alternative delivery scenarios. The first step is achieved through an application of $SUM^{[5,10]}(E_{dep}, C)$ and $SUM^{[10,15]}(E_{dep}, C)$. The latter involves the modification of deployment start times t_i based on the user's assumption followed by a second aggregation over E_{dep}, now including the modified temporal information.

5.1 Iterative creation and computation of scenarios

In our exemplary use case UC 2, the user wants to analyze the influence of applying express delivery on the deployment costs induced during a considered time frame. To this end, as shown in Figure 8, he modifies the start times t_i of a selected subset of E_{dep}, creating a new scenario $S2$. He then needs to analyze the modified data underlying $S2$, e.g., by repeating the described aggregation over intervals $[5, 10]$ and $[10, 15]$. The results scenarios can be compared, stored, or further processed. Naturally, we want to reuse as many results as possible throughout this process. To this end, we must provide information about their derivation and evaluate it in the processing of operators. It is important to note that most of the derived results are kept in memory as intermediate results, enabling fast access and iterative application of different operators. Of course, we can persist results to enable their reuse at a later point in time.

Incorporating modifications and insertions The creation of a new scenario virtually always goes along with modifications to some minor part of the underlying data. For example, the scenario described above is derived on the assumption of modified delivery times for a group of items. Note that we can calculate the succeeding aggregation very efficiently given the fraction of modified times is relatively small. In particular, rather than applying $AGG^{\underline{T}}$ to all $e_i \in E_{dep}$, we only need to compute $SUM^{\underline{T}}$ over the affected events, using the delta values t_i^{Δ} that represent the previous modification. We can then

compute sum_{S2} as $sum_{S2} = sum_{S1} + sum_{\Delta}$. Modifications can be incorporated in the described iterative fashion only if we can preserve the semantics of the applied operators. For example, we can apply a similar step to update previously derived histograms or distribution parameters to incorporate modified or new fact data in an iterative fashion, rather than recomputing the complete distribution. Similarly, we can update a bivariate distribution (derived using COR) when one of the marginals is modified. This is because internally, COR essentially relies on summing up the joint densities in the result histogram. Thus, the same iterative approach as above can be applied. Note that we can not use this approach when the analysis process includes operators whose semantics are not preserved under modifications, e.g., for the computation of extrema (both in the sense of standard aggregation and the computation of $MIN^{\underline{T}}$ and $MAX^{\underline{T}}$). A comprehensive consideration of how new data and modifications can be incorporated in the execution of (sequences of) operators is yet outstanding.

5.2 Parallelization

Besides optimizing the calculation of succeeding operators based on the provenance of intermediate results, we also need to address the issue of long response times due to large amounts of data processed by operators such as DRV. To this end, we considered different forms of parallelization. For many of our operators, a large part of their overall costs is determined by loading and processing individual columns. Those can be executed independently, returning relatively small results which are merged in a final step. In previous work, we applied alternative ways of parallel loading and processing of underlying input data. We evaluated parallelization of computations executed within an operator and parallel processing of operators between cores in [ERM+10], where we exemplified our approaches using the data-intensive operator DRV. The reported results show that in cases of large amounts of input data, parallel loading and processing of partitioned data over many cores is beneficial due to dominant loading times. Conversely, when operators process relatively small amounts of data, we can apply threaded execution within a single operator.

6 Related Work

Existing approaches for uncertain data management foremost focus on areas such as the management of sensor data, information extraction results, or scientific data. In this context, those approaches mainly address the representation, indexing, and analysis of data represented through tuple alternatives [HAKO09, SD07, ABS+06] and values distributed over discrete or continuous domains [SMM+08, AW09]. The generally high complexity of queries over uncertain data is a well-known problem and has been discussed – among other issues – with respect to join evaluation [Che06], range predicates [DS07, CXP+04], and exact and approximate aggregate computation [MIW07].

The abovementioned aspects serve as valuable building blocks for the analysis part of the planning processes we envision. However, our work focuses on the specific aspects of derivation and modification of uncertainty and interdependencies in data. To incorporate those aspects, we apply symbolic and equi-width histogram representations of distribution functions. The use of histograms and the performance of different partitioning schemes, such as equi-depth or MaxDiff, have been investigated in depth (see, e.g., [PHIS96]) with respect to both their construction efficiency and accuracy. Likewise, the usage of histogram-based and symbolic representations for uncertain data management has been previously discussed, e.g., in [SMM+08, AW09]. We similarly exploit the histogram model to represent arbitrary distributions generically; in addition, our data model employs uni- and multivariate histograms to represent and efficiently handle modifications (deltas) and correlation information. The aspect of correlations in data was addressed previously primarily for the case of tuple alternatives and discrete value distributions. The approach reported in [SD07] uses graphical models to represent such dependencies in a factored fashion and discusses efficient inference-based query evaluation over the graphs. Although, in general, the graphical representation can be applied in the face of continuous distributions, [SD07] does not address the computation or introduction of correlation information by users. While the authors in [KO08] discuss efficient approaches for the introduction of "conditioning" constraints (e.g., implications and mutual exclusion) and queries over conditioned data, they similarly do not discuss the separate representation and introduction of *arbitrary* correlation to data as we do. Our previous work [ER10, ERM+10] introduced the general ideas of our support for scenario-based planning, but lacked a comprehensive discussion and evaluation of our operators' efficiency, foremost as regards the COR and $AGG^{\underline{T}}$ operators. In this paper, we addressed this open issue as a basic prerequisite to judge their practical applicability. The challenge of efficiently incorporating modified data in the face of scenario creation relates to the topic of data lineage. Lineage handling has been previously discussed in the context of probabilistic data, e.g., in [ABS+06, STW08] and in the broader context of view maintenance in data warehouses [CWW00] and data-centric workflows. Its application for optimizing scenario-based planning process constitutes an interesting new facet complementing previous research.

7 Conclusion and Future Work

In this paper, we extended our previous work on operators for derivation, analysis and modification of uncertain data in the context of scenario-based planning processes. We derived and discussed the complexity of these operators and created a basis for assessing them in different application scenarios. We also validated the analytical results through an experimental evaluation. Generally, we observe a dominating cost factor for loading histogram structures from the database, while the computation routines themselves are highly efficient and introduce only small additional costs with growing data complexity. We further highlighted opportunities for optimization concerning both parallelization and the incremental execution of steps in an analysis process, including the efficient derivation of related scenarios. Finally, we addressed related research topics touching on various

aspects of both the functional and performance-related aspects of the presented work.

Apart from enabling provenance handling in the scenario derivation process, another highly important factor of future work is an assessment of the accuracy of results derived in this process. Naturally, we cannot quantify the "correctness" of a computed result scenario since the future fulfillment of the applied assumptions is unknown. Still, we can measure discretization errors introduced through operator applications and quantify the resulting trade-offs between accuracy and efficiency. In this respect, both varying aspects of the data model and the applied operators can help enable manual and automatic optimization based on users' preferences. For example, applying an alternative histogram partitioning scheme such as equi-depth could decrease approximation errors at the cost of lower construction and update efficiency. Conversely, a user might resort to approximate operators for the benefit of lower run-times. In this context, a complementary track of our work investigates approximate temporal aggregation based on a clustering of events with similar temporal associations. A comprehensive investigation of the exemplified trade-offs is subject to future work.

References

[ABS+06] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha Nabar, Tomoe Sugihara, and Jennifer Widom. Trio: A System for Data, Uncertainty, and Lineage. In *VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 1151–1154. VLDB Endowment, 2006.

[AW09] Parag Agrawal and Jennifer Widom. Continuous Uncertainty in Trio. In *MUD*. Stanford InfoLab, 2009.

[Che06] Reynold Cheng. Efficient join processing over uncertain data. In *In Proceedings of CIKM*, pages 738–747, 2006.

[CWW00] Yingwei Cui, Jennifer Widom, and Janet L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst.*, 25(2):179–227, 2000.

[CXP+04] Reynold Cheng, Yuni Xia, Sunil Prabhakar, Rahul Shah, and Jeffrey Scott Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 876–887. VLDB Endowment, 2004.

[DS07] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007.

[ER10] Katrin Eisenreich and Philipp Rösch. Handling Uncertainty and Correlation in Decision Support. In *Proceedings of 4th Workshop on Management of Uncertain Data at VLDB 2010*, September 2010.

[ERM+10] Katrin Eisenreich, Philipp Rösch, Volker Markl, Gregor Hackenbroich, and Robert Schulze. Handling of Uncertainty and Temporal Indeterminacy for What-if Analysis. In *Proceedings of Workshop on Enabling Real-Time Business Intelligence at VLDB 2010*, September 2010.

[HAKO09] Jiewen Huang, Lyublena Antova, Christoph Koch, and Dan Olteanu. MayBMS: A Probabilistic Database Management System. In *Proceedings of the 35th SIGMOD International Conference on Management of Data*, pages 1071–1074, New York, NY, USA, 2009. ACM.

[JLF10] Bernhard Jäcksch, Wolfgang Lehner, and Franz Faerber. A Plan for OLAP. In *EDBT*, pages 681–686, 2010.

[JXW+08] Ravi Jampani, Fei Xu, Mingxi Wu, Luis L. Perez, Christopher Jermaine, and Peter J. Haas. MCDB: A Monte Carlo Approach to Managing Uncertain Data. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 687–700, New York, NY, USA, 2008. ACM.

[KO08] Christoph Koch and Dan Olteanu. Conditioning probabilistic databases. *Proc. VLDB Endow.*, 1(1):313–325, 2008.

[MIW07] Raghotham Murthy, Robert Ikeda, and Jennifer Widom. Making Aggregation Work in Uncertain and Probabilistic Databases. Technical Report 2007-7, Stanford InfoLab, June 2007.

[MST07] G. Mayor, J. Suner, and J. Torrens. Sklar's Theorem in Finite Settings. *Fuzzy Systems, IEEE Transactions on*, 15(3):410 –416, june 2007.

[Nel06] Roger B. Nelsen. *An Introduction to Copulas*. Springer Series in Statistics. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[PHIS96] Viswanath Poosala, Peter J. Haas, Yannis E. Ioannidis, and Eugene J. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. *SIGMOD Rec.*, 25(2):294–305, 1996.

[SD07] Prithviraj Sen and A. Deshpande. Representing and Querying Correlated Tuples in Probabilistic Databases. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 596–605, 2007.

[Skl59] A. Sklar. Fonctions de repartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de L'Universite de Paris*, 8:229–231, 1959.

[SMM+08] Sarvjeet Singh, Chris Mayfield, Sagar Mittal, Sunil Prabhakar, Susanne Hambrusch, and Rahul Shah. Orion 2.0: Native Support for Uncertain Data. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD*, pages 1239–1242, New York, NY, USA, 2008. ACM.

[STW08] Anish Das Sarma, Martin Theobald, and Jennifer Widom. Exploiting Lineage for Confidence Computation in Uncertain and Probabilistic Databases. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 1023–1032, Washington, DC, USA, 2008. IEEE Computer Society.